Documentazione progetto Chat-APP

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

2.3 Use case 6

2.4 Pianificazione 7

2.5 Analisi dei mezzi 7

2.5.1 Software 7

2.5.2 Hardware 7

3 Progettazione 9

3.1 Design dell’architettura del sistema 9

3.2 Design dei dati e database 9

3.3 Design delle interfacce 10

4 Implementazione 11

4.1 Importare Firebase 11

4.2 Auth 11

4.2.1 Login con Google 11

4.2.2 Login con email e password 11

4.3 Index 12

4.3.1 onAuthStateChanged 12

4.3.2 InvioMessaggio 12

4.3.3 convertToBase64 12

4.3.4 onChildAdded 12

4.3.5 Esportazione (CSV) 12

4.3.6 Esportazione (PDF) 13

4.4 CSS 13

5 Test 14

5.1 Protocollo di test 14

5.2 Risultati test 18

5.3 Mancanze/limitazioni conosciute 21

6 Consuntivo 22

7 Conclusioni 23

7.1 Sviluppi futuri 23

7.2 Considerazioni personali 23

8 Glossario 23

9 Sitografia 24

10 Allegati 25

# 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 non saranno più visibili.

# 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\*

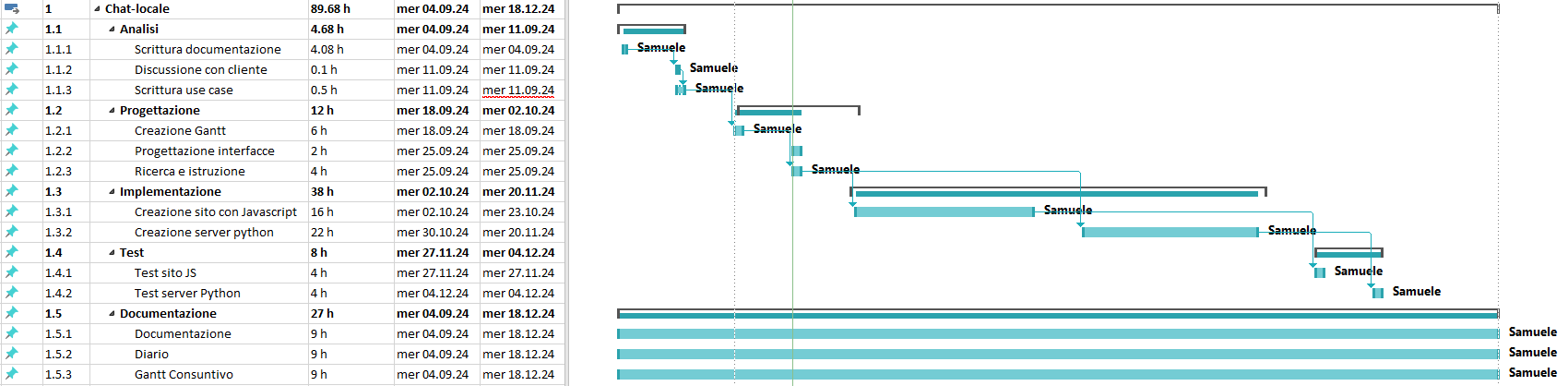
## Analisi dei mezzi

### 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

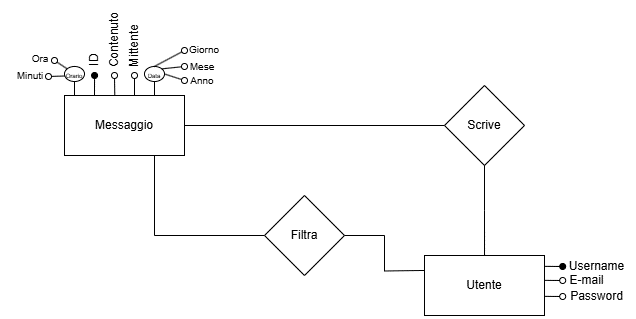


# Progettazione

## 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.

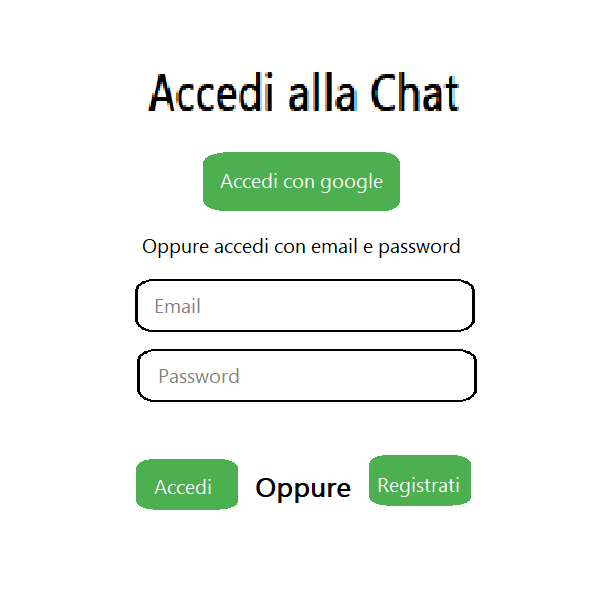
## Design dei dati e database



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

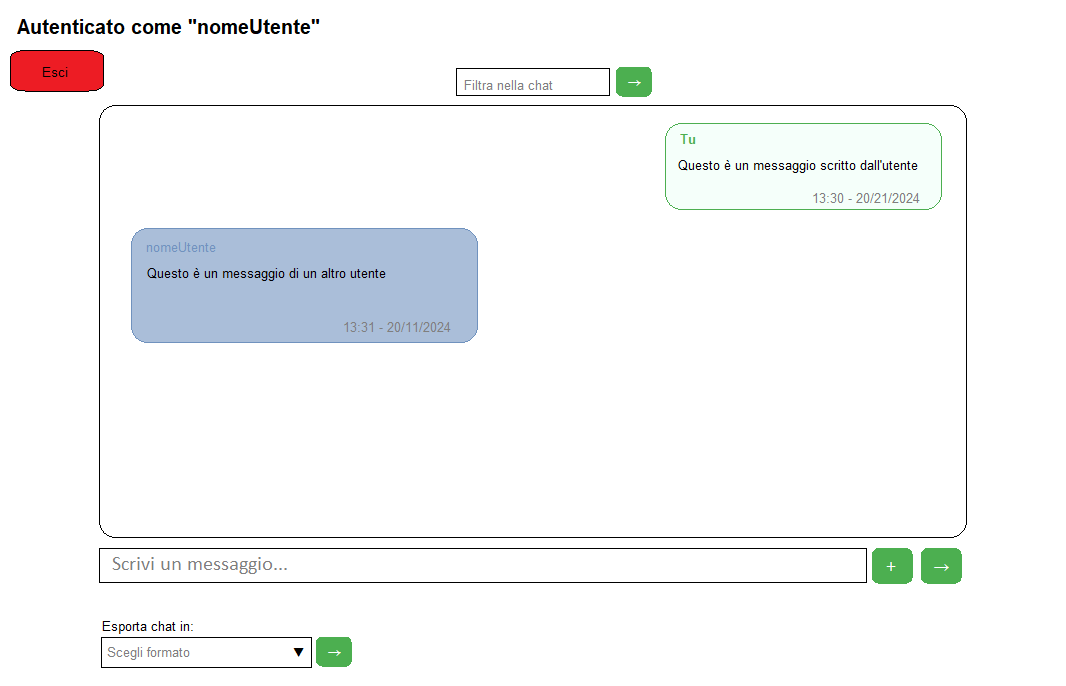
con un account Google oppure fornendo una email

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.



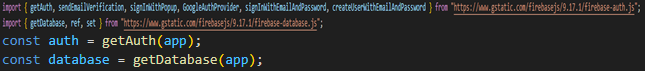
L’import serve a importare tramite CDN la funzione per inizializzare Firebase. “firebaseConfig” è un oggetto che contiene informazioni utili ad indentificare il progetto aperto nella console di Firebase. “app” è la variabile dove si salva l’inizializzazione di Firebase.

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 sono salvati i messaggi). Snapshot contiene le informazioni di un solo messaggio. Per recuperare gli attributi bisogna fare “snapshot.val().\*nomeAttributo\*.

### Esportazione (CSV)

**const blob = new Blob([contenuto], { type: 'text/csv' });**

Creazione di un blob specificandone il tipo "testo/csv".

**const url = URL.createObjectURL(blob);**

Creazione dell'url per il download del blob (posizione del blob contenente il csv)

**const a = document.createElement('a');**

Generazione di un elemento "<a>" per il download dell'elemento.

**a.href = url;**

Impostazione dell'attributo "href" dell'elemento anchor nel url del blob.

**a.download = 'chat.csv';**

Definizione del nome del file.

**a.click();**

.click() è un metodo per scaricare automaticamente un file, simula la pressione di un utente.

### Esportazione (PDF)

**Importazione dei moduli necessari (il primo importa jsPDF, mentre il secondo il supporto per le tabelle)**

<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-autotable/3.5.21/jspdf.plugin.autotable.min.js"></script>

**Inizializzazione degli oggetti**

const { jsPDF } = window.jspdf;

const doc = new jsPDF();

**doc.text("Chat esportata", 10, 10);**

Aggiunge del testo a delle coordinate x e y definite.

**doc.autoTable({});**

Funzione per aggiungere una tabella al documento pdf.

**Head: [ ]**

È un attributo della funzione all’interno di “autoTable” che definisce un header. Accetta un array di parole.

**Body: dati.map(messaggio => [ ]);**

Tramite il metodo map() viene restituito un array di oggetti da dare all’attributo body.

**Margin:** e **Theme:**

Sono attributi di estetica, in particolare margin serve a distanziare gli elementi, mentre theme applica uno stile predefinito alla tabella.

**doc.save(“chat.pdf”);**

save() è un metodo che permette di salvare l’oggetto jsPDF con un nome specificato (in questo caso chat.pdf).

## 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

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-01 | **Nome:** | Ricerca e filtri |
| **Descrizione:** | Bisogna impedire agli utenti di filtrare la chat per valori vuoti o nulli. | | |
| **Prerequisiti:** | Bisogna aver eseguito l’accesso alla chat | | |
| **Procedura:** | 1. Eseguire l’accesso alla chat in: <http://chat-samuele.labosam.cpt.local/auth.html> 2. premere il bottone accanto alla textbox (con placeholder “Inserire parola chiave…”) senza immettere una chiave e verificare se l’utente viene avvisato. | | |
| **Risultati attesi:** | Viene mostrato un alert all’utente per informarlo che manca la parola chiave | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-01 | **Nome:** | Ricerca e filtri |
| **Descrizione:** | Verifica che la chat venga filtrate per una parola chiave | | |
| **Prerequisiti:** | Bisogna aver eseguito l’accesso alla chat | | |
| **Procedura:** | 1. Eseguire l’accesso alla chat 2. Immettere una parola chiave nel textbox e premere il bottone per applicare il filtro | | |
| **Risultati attesi:** | Vengono mostrati tutti i messaggi che contengono la chiave inserita | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-01 | **Nome:** | Ricerca e filtri |
| **Descrizione:** | Verifica del reset del filtro | | |
| **Prerequisiti:** | Bisogna aver eseguito l’accesso alla chat | | |
| **Procedura:** | 1. Eseguire l’accesso alla chat 2. Premere sul pulsante “X” vicino alla casella di testo | | |
| **Risultati attesi:** | Vengono rimostrati tutti i messaggi | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-02 | **Nome:** | Controllo di accesso (registrazione con Google) |
| **Descrizione:** | Verifica del funzionamento della registrazione usando Google. | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account google | | |
| **Procedura:** | 1. Nella pagina di login premere “Accedi con google” 2. Eseguire l’accesso nel pop-up inserendo le credenziali di Google (se si esegue l’accesso a Google con un solo account nel browser potrebbe non essere richiesto l’inserimento delle credenziali) 3. Attendere il reindirizzamento alla chat | | |
| **Risultati attesi:** | L’utente deve essere reindirizzato alla pagina della chat | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-02 | **Nome:** | Controllo di accesso (login con Google) |
| **Descrizione:** | Verifica del funzionamento del login usando Google. | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account Google registrato | | |
| **Procedura:** | 1. Nella pagina di login premere “Accedi con Google” 2. Attendere il reindirizzamento alla chat | | |
| **Risultati attesi:** | L’utente deve essere reindirizzato alla pagina della chat | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-02 | **Nome:** | Controllo di accesso (registrazione con email) |
| **Descrizione:** | Verifica del funzionamento della registrazione usando una email. | | |
| **Prerequisiti:** | Bisogna aver accesso ad una email | | |
| **Procedura:** | 1. Nella pagina di login immettere l’email e scegliere una password 2. Se l’email è già in uso, non valida o la password troppo corta restituisce un errore 3. Verificare l’account dal link ricevuto via mail 4. Attendere il reindirizzamento alla chat | | |
| **Risultati attesi:** | L’utente deve essere reindirizzato alla pagina della chat | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-02 | **Nome:** | Controllo di accesso (login con email) |
| **Descrizione:** | Verifica del funzionamento del login usando una email. | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account (email) registrato | | |
| **Procedura:** | 1. Nella pagina di login immettere l’email e la password 2. Se la password è errata apparirà una scritta in rosso “Password dimenticata?” che reindirizza ad una pagina per il reset della password 3. Attendere il reindirizzamento alla chat | | |
| **Risultati attesi:** | L’utente deve essere reindirizzato alla pagina della chat | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-03 | **Nome:** | Storico chat (CSV) |
| **Descrizione:** | Un utente deve poter scaricare la chat coi messaggi disponibili in un file CSV | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account registrato nella piattaforma | | |
| **Procedura:** | 1. Accedere alla chat 2. Premere sul menu a tendina e selezionare CSV 3. Premere su esporta 4. Aprire il file scaricato e verificare se i messaggi scaricati siano identici a quelli nella chat | | |
| **Risultati attesi:** | L’utente deve avere un file CSV con i messaggi disponibili nella chat | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-03 | **Nome:** | Storico chat (PDF) |
| **Descrizione:** | Un utente deve poter scaricare la chat coi messaggi disponibili in un file PDF | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account registrato nella piattaforma | | |
| **Procedura:** | 1. Accedere alla chat 2. Premere sul menu a tendina e selezionare PDF 3. Premere su esporta 4. Aprire il file scaricato e verificare se i messaggi scaricati siano identici a quelli nella chat | | |
| **Risultati attesi:** | L’utente deve avere un file PDF con i messaggi disponibili nella chat | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-04 | **Nome:** | Analisi delle conversazioni (data messaggio) |
| **Descrizione:** | L’utente deve poter visualizzare l’orario e la data nella quale viene inviato il messaggio | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account registrato nella piattaforma | | |
| **Procedura:** | 1. Accedere alla chat 2. Scrivere un messaggio 3. Controllare se l’orario e la data del messaggio siano corretti | | |
| **Risultati attesi:** | L’utente deve vedere l’orario e la data del messaggio | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-04 | **Nome:** | Analisi delle conversazioni (volume del messaggio) |
| **Descrizione:** | L’utente deve poter visualizzare il conteggio dei caratteri dei messaggi | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account registrato nella piattaforma | | |
| **Procedura:** | 1. Accedere alla chat 2. Scrivere un messaggio 3. Controllare se il conteggio dei caratteri sia corretto (aggiungendo spazi o caratteri speciali) 4. Se il messaggio è vuoto (come per una immagine) allora viene visualizzato “Nessun carattere”, altrimenti “x caratteri/e” | | |
| **Risultati attesi:** | L’utente deve poter vedere il numero di caratteri del messaggio | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-04 | **Nome:** | Analisi delle conversazioni (messaggi / giorno) |
| **Descrizione:** | L’utente deve poter visualizzare una statistica dei messaggi al giorno (negli ultimi 30 giorni) | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account registrato nella piattaforma ed eventualmente accesso alla console di Firebase | | |
| **Procedura:** | 1. Accedere alla chat 2. Annotarsi il rapporto dei messaggi al giorno 3. Scrivere dei messaggi per alzare la media 4. Ricaricare la pagina e controllare se la media dei messaggi è cambiata 5. Eventualmente accedere alla console di Firebase e cambiare manualmente la data dei messaggi rendendoli più vecchi di 30 giorni | | |
| **Risultati attesi:** | Visualizzare una media dei messaggi ogni 30 giorni funzionante | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-05 | **Nome:** | Visibilità messaggi |
| **Descrizione:** | L’utente deve poter visualizzare solamente i messaggi più recenti (non più vecchi di 30 giorni) | | |
| **Prerequisiti:** | Bisogna aver accesso ad un account registrato nella piattaforma e l’accesso alla console di Firebase | | |
| **Procedura:** | 1. Accedere alla chat 2. Scrivere un messaggio 3. Accedere alla console di Firebase e cambiare la data a 30 giorni fa 4. Controllare la chat | | |
| **Risultati attesi:** | Il messaggio non deve essere visibile | | |

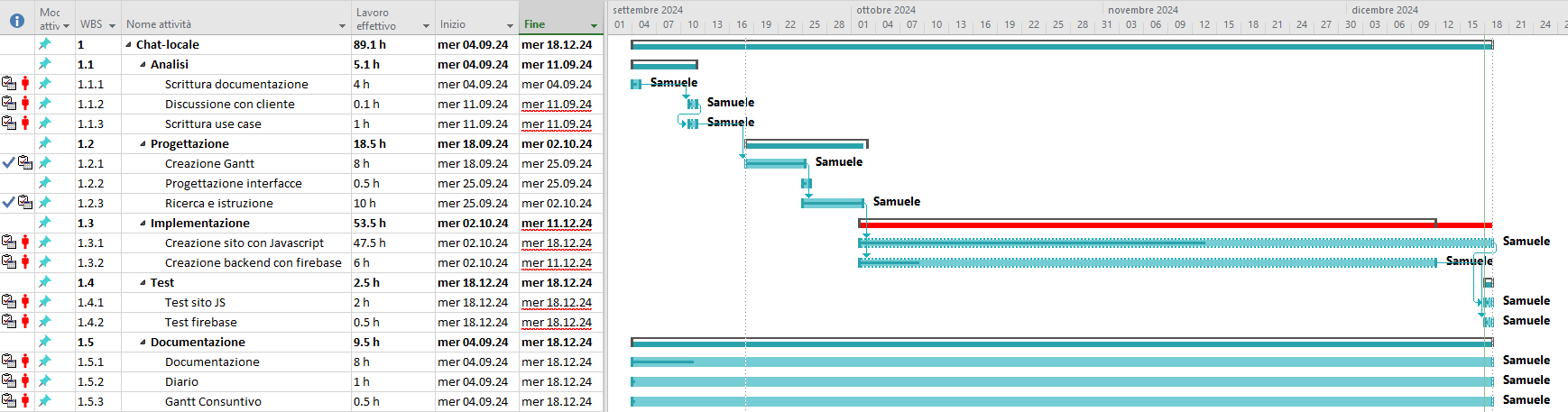
## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Risultato** | **Descrizione** | **Data del test** |
| 001 | Passato | **Risultato atteso:**  Viene mostrato un alert all’utente per informarlo che manca la parola chiave  **Risultato effettivo:**  Il test ha avuto un esito positivo, quando viene premuto il pulsante l’alert viene mostrato correttamente. | 18.12.2024 |
| 002 | Passato | **Risultato atteso:**  Vengono mostrati tutti i messaggi che contengono la chiave inserita  **Risultato effettivo:**  Il test ha avuto un esito positivo, quando viene applicato il filtro, vengono mostrati solamente i messaggi che contengono la parola chiave. Nell’immagine viene filtrato per la parola “ciao”. | 18.12.2024 |
|  |
| 003 | Passato | **Risultato atteso:**  Vengono rimostrati tutti i messaggi  **Risultato effettivo:**  Il test ha avuto un esito positivo, quando un utente preme il pulsante per resettare il filtro i messaggi vengono mostrati correttamente. | 18.12.2024 |
| 004 | Passato | **Risultato atteso:**  L’utente deve essere reindirizzato alla pagina della chat  **Risultato effettivo:**  Il test ha avuto un esito positivo, quando viene premuto il pulsante per la registrazione tramite Google, viene aperto il pop-up dove inserire le credenziali, e l’utente viene reindirizzato alla chat. | 18.12.2024 |
| 005 | Passato | **Risultato atteso:**  L’utente deve essere reindirizzato alla pagina della chat  **Risultato effettivo:**  Il test ha avuto un esito positivo, quando viene premuto il pulsante per l’accesso tramite Google, viene aperto un pop-up (dove non va inserito nulla), e l’utente viene reindirizzato alla chat. | 18.12.2024 |
| 006 | Passato | **Risultato atteso:**  L’utente deve essere reindirizzato alla pagina della chat  **Risultato effettivo:**  Il test ha avuto un esito positivo, dopo aver verificato l’email, l’utente viene reindirizzato alla chat. | 18.12.2024 |
| 007 | Passato | **Risultato atteso:**  L’utente deve essere reindirizzato alla pagina della chat  **Risultato effettivo:**  Il test ha avuto un esito positivo, dopo aver inserito le credenziali di accesso di un account registrato, l’utente viene reindirizzato alla chat. | 18.12.2024 |
| 008 | Passato | **Risultato atteso:**  L’utente deve avere un file CSV con i messaggi disponibili nella chat  **Risultato effettivo:**  Il test ha avuto un esito positivo, quando viene premuto il pulsante per esportare (e selezionato nel menu a tendina “CSV”), viene scaricato un file CSV contenente i messaggi più recenti (non più vecchi di 30 giorni). | 18.12.2024 |
| 009 | Passato | **Risultato atteso:**  L’utente deve avere un file PDF con i messaggi disponibili nella chat  **Risultato effettivo:**  Il test ha avuto un esito positivo, dopo aver verificato l’email, l’utente viene reindirizzato alla chat. | 18.12.2024 |
|  |  |  |  |
| 010 | Passato | **Risultato atteso:**  L’utente deve vedere l’orario e la data del messaggio  **Risultato effettivo:**  Il test ha avuto un esito positivo, l’orario e la data vengono mostrati correttamente. | 18.12.2024 |
| 011 | Passato | **Risultato atteso:**  L’utente deve poter vedere il numero di caratteri del messaggio  **Risultato effettivo:**  Il test ha avuto un esito positivo, dopo aver scritto un messaggio, sotto vengono mostrati il giusto numero di caratteri (spazi esclusi). Nell’immagine sottostante viene mostrata una parola con 6 caratteri e 2 spazi all’interno. | 18.12.2024 |
| 012 | Passato | **Risultato atteso:**  Visualizzare una media dei messaggi ogni 30 giorni funzionante  **Risultato effettivo:**  Il test ha avuto un esito positivo, dopo aver inviato numerosi messaggi, la è cambiata. Avendo modificato tramite Firebase la data manualmente si è notato che la media dei messaggi si è abbassata. | 18.12.2024 |
|  |  |  |  |
| 013 | Passato | **Risultato atteso:**  Il messaggio non deve essere visibile  **Risultato effettivo:**  Il test ha avuto un esito positivo, dopo aver cambiato manualmente la data del messaggio tramite la console di Firebase, il messaggio non veniva caricato. | 18.12.2024 |

## Mancanze/limitazioni conosciute

Potevano essere precisate le cause di errore dopo un tentativo fallito di accesso, siccome viene semplicemente comunicato all’utente un errore generico.

# Consuntivo



# Conclusioni

Considerando la concorrenza, questo progetto potrebbe non avere un impatto significativo su larga scala, dato che esistono alternative con un maggior numero di funzionalità. Tuttavia, rappresenta per me un piccolo successo personale; nonostante le difficoltà incontrate, sono riuscito a superare le mie aspettative. La soluzione realizzata è usufruibile per chi cerca principalmente la funzionalità. Il progetto è stato completato entro la scadenza concordata, rispettando i requisiti stabiliti con il cliente. Purtroppo i tempi per le singole attività sono stati calcolati male.

## Sviluppi futuri

In futuro si potrebbe aggiungere la possibilità di discutere con persone singole, salvando molteplici chat con due utenti. Altre piccole migliorie potrebbero essere l’aggiunta di username, l’aggiunta di foto profilo e la possibilità di eliminare il proprio account.

## Considerazioni personali

Lo scopo principale di questo progetto non era sviluppare un’applicazione di messaggistica, ma introdurre le dinamiche di lavoro. Questa esperienza mi ha permesso di acquisire competenze e mi ha allenato a ricercare autonomamente informazioni. Ho compreso quanto la mia percezione iniziale del lavoro fosse diversa rispetto al risultato effettivo. Grazie a questa esperienza, in futuro saprò gestire un progetto con maggiore consapevolezza.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| Database | Una banca dati dove archiviare e gestire dati |
| CSS | Linguaggio che permette di definire il layout e la grafica di una pagina web. |
| Gantt | Diagramma utilizzato per la gestione dei progetti che mostra le attività pianificate ed effettive e il tempo previsto ed effettivo |
| HTML | Linguaggio per strutturare il contenuto di un sito internet |
| JavaScript | Linguaggio che aggiunge logica e interattività nei siti |
| Firebase | Servizio online per lo sviluppo di applicazioni, offre vari tool come il database, autenticazione, hosting… |
| Tag html | Elemento usato in una pagina internet che definisce una parte della struttura |
| Textbox | Casella di testo |
| Placeholder | Testo mostrato in una textbox vuota  (Es: “*Inserire la email…*”) |
| alert() | Metodo di JavaScript che apre una finestra pop-up con un messaggio personalizzato |
| URL | Stringa che definisce la posizione di una risorsa |

# Sitografia

* <https://chatgpt.com/>
* <https://firebase.google.com/docs>
* <https://www.geeksforgeeks.org/>
* <https://www.w3schools.com/>
* <https://youtube.com/>

# Allegati

* Diari di lavoro
* Quaderno dei compiti (QdC)
* Abstract
* Applicativo

