Chat con Firebase

1 Introduzione 4

1.1 Informazioni sul progetto 4

1.2 Abstract 4

1.2.1 Situazione iniziale 4

1.2.2 Attuazione 4

1.2.3 Risultati 4

1.3 Scopo 4

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 5

2.3 Use case 6

2.4 Pianificazione 1

2.5 Analisi dei mezzi 1

2.5.1 Software 1

2.5.2 Hardware 1

3 Progettazione 1

3.1 Design dell’architettura del sistema 1

3.2 Design dei dati e database 1

3.3 Design delle interfacce 2

3.4 Design procedurale 3

4 Implementazione 3

4.1 Creazione di un progetto con Firebase 3

4.2 Creazione del progetto 3

4.3 Index.html 4

4.4 Chat.html 4

4.5 Index.js 4

4.5.1 registerNewUser 4

4.5.2 loginUser 4

4.5.3 logoutUser 5

4.5.4 sendMessage 5

4.5.5 viewAllUsers 5

4.5.6 addToChannel 5

4.5.7 createChannel 5

4.5.8 channelExists 5

4.5.9 deleteOldMessages (da finire) 6

4.5.10 loadChannels 6

4.5.11 getChatBox 6

4.5.12 changeChannel 6

4.5.13 showNickname 6

4.5.14 reloadMessages 6

4.5.15 deleteChannel 6

4.5.16 openModifyChannel 6

4.5.17 modifyChannel (da finire) 7

4.5.18 addToBan 7

4.5.19 banUser 7

4.5.20 loadAdminOption 7

4.5.21 resizeMessagesDiv 7

4.5.22 onEnterSendMessage 7

4.6 Sidebars.js 7

4.7 Bootstrap.bundle.min.js 7

4.8 Bootstrap.css 7

4.9 Sidebars.css 7

5 Test 8

5.1 Protocollo di test 8

5.2 Risultati test 10

5.3 Mancanze/limitazioni conosciute 10

6 Consuntivo 10

7 Conclusioni 10

7.1 Sviluppi futuri 10

7.2 Considerazioni personali 10

Sitografia 10

8 Allegati 10

# Introduzione

## Informazioni sul progetto

Sono un’allieva del terzo anno di informatica alla Scuola d’Arti e Mestieri a Trevano. Ho realizzato questo progetto durante il modulo 306 con la supervisione del docente Geo Petrini. Il progetto è iniziato il 09.09.2021 è si è concluso il 23.12.2021.

## Abstract

### Situazione iniziale

Lo scopo di questo progetto è quello di realizzare una chat multiutente gestita per canali. L’applicazione permette a qualsiasi utente di registrarsi tramite un indirizzo e-mail. Sarà poi possibile creare nuovi canali specificando dopo quanto tempo i messaggi saranno cancellati ed eventualmente bannare gli utenti. Per gli utenti con ruolo admin è anche possibile inviare messaggi in broadcast su tutti i canali.

### Attuazione

Nei requisiti non c’erano limitazioni per il linguaggio da utilizzare quindi ho deciso di sviluppare un’applicazione web utilizzando HTML, Javascript e CSS e l’ausilio di bootstrap. Per le interazioni con il database di firebase ho utilizzato la documentazione fornita sul sito ufficiale e altre guide trovate online.

### Risultati

I requisiti del progetto sono stati soddisfatti però avrei potuto gestire meglio la struttura del database ad oggetti che non avevo mai utilizzato e non l’ho saputa gestire al meglio. Non sono inoltre riuscita a gestire al meglio gli accessi al database.

## Scopo

Il progetto è pensato per utenti che vogliono chattare fra di loro tramite canali.

# Analisi

## Analisi del dominio

L’applicazione permette a chiunque sia registrato nel sito di comunicare tramite canali. Il funzionamento è simile ad’ altri prodotti già esistenti come WhatsApp o Telegram ma senza la possibilità di comunicare direttamente fra due utenti. L’applicazione è stata utilizzata utilizzando principalmente tre linguaggi: HTML, JavaScript e CSS per la parte grafica ed’ è necessario acquisire le conoscenze per manipolare i dati nel database di firebase.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Installazione server Firebase |
| **Priorità** | alta |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Si necessita una maschera di login |
| **002** | Si dovranno poter immettere nuovi allievi |
| **003** | Dovrà essere possibile la ricerca di allievi |

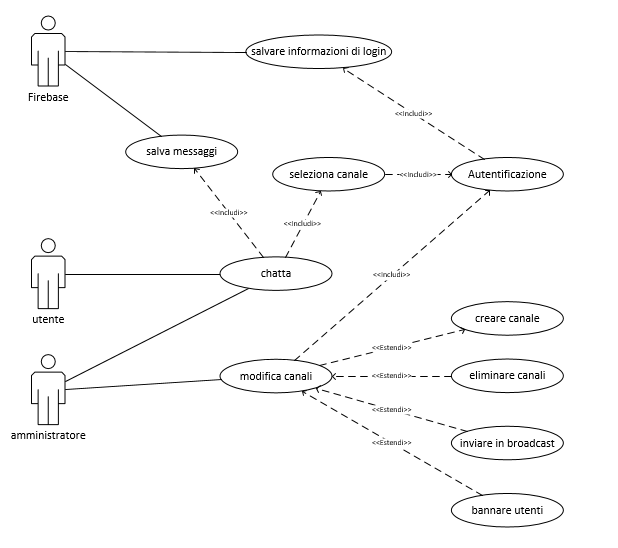
|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Autentificazione degli utenti |
| **Priorità** | alta |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | È necessario salvare nome e password |
| **002** | Le password devono essere salvate in modo sicuro sui database di Firebase |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Gli amministratori devono poter eseguire determinate azioni in più rispetto agli utenti normali |
| **Priorità** | 1 |
| **Versione** | media |
| **Sotto requisiti** | |
| **001** | Creare canali |
| **002** | Modificare canali |
| **003** | Eliminare canali |
| **004** | Inviare messaggi in broadcast |
| **005** | Possibilità di bannare utenti |

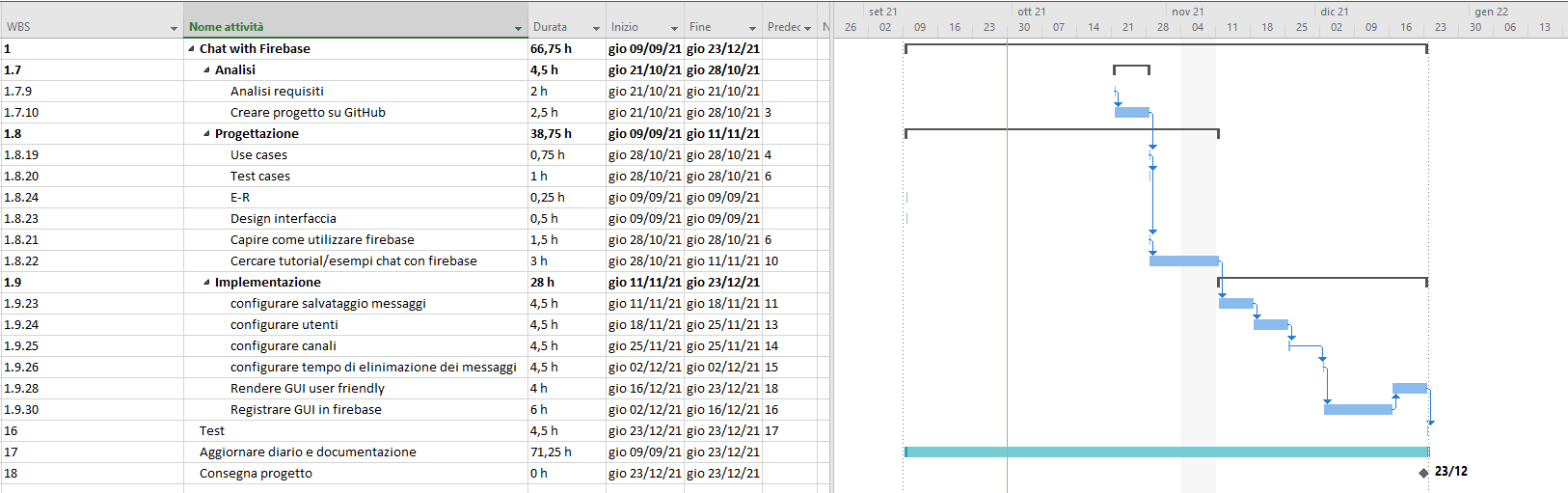
|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Configurazione dei messaggi |
| **Priorità** | bassa |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | L’applicazione deve gestire i messaggi auto purge |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Deve essere supportata l’intera mappa unicode |
| **Priorità** | bassa |
| **Versione** | 1.0 |

## Use case



## Pianificazione



## Analisi dei mezzi

Per realizzare questo progetto ho utilizzato VS Code come editor.

### Software

* Visual Studio Code (v 1.62)
* Firebase (v 8.2.1)
* Bootstrap (v 5)

### Hardware

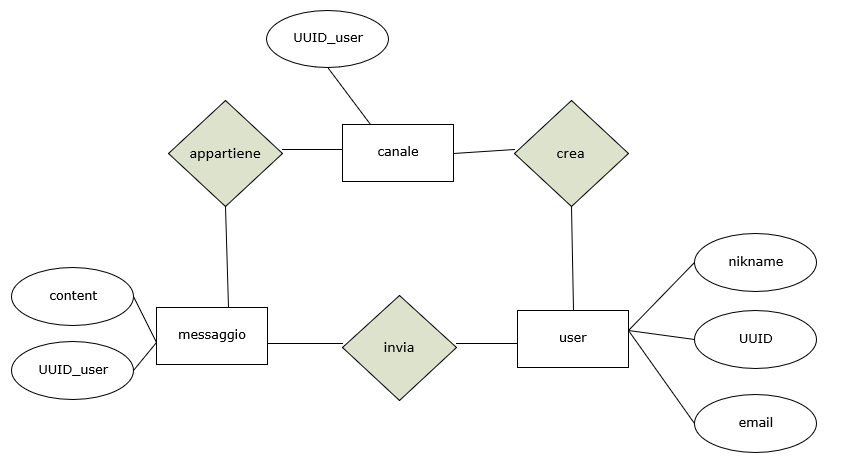
L’applicazione è responsive e quindi utilizzabile su qualsiasi dispositivo dotato di un browser e una connessione internet, non è quindi necessaria un hardware specifica.

# Progettazione

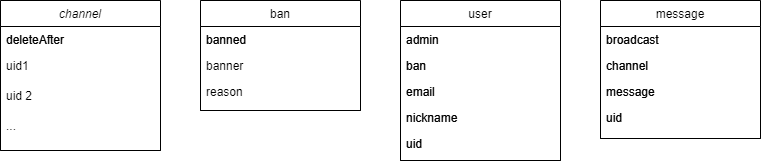
## Design dell’architettura del sistema

Data la necessità di connettersi al database di Firebase per lavorare sono dovuta uscire su internet tramite il proxy della scuola.

## Design dei dati e database

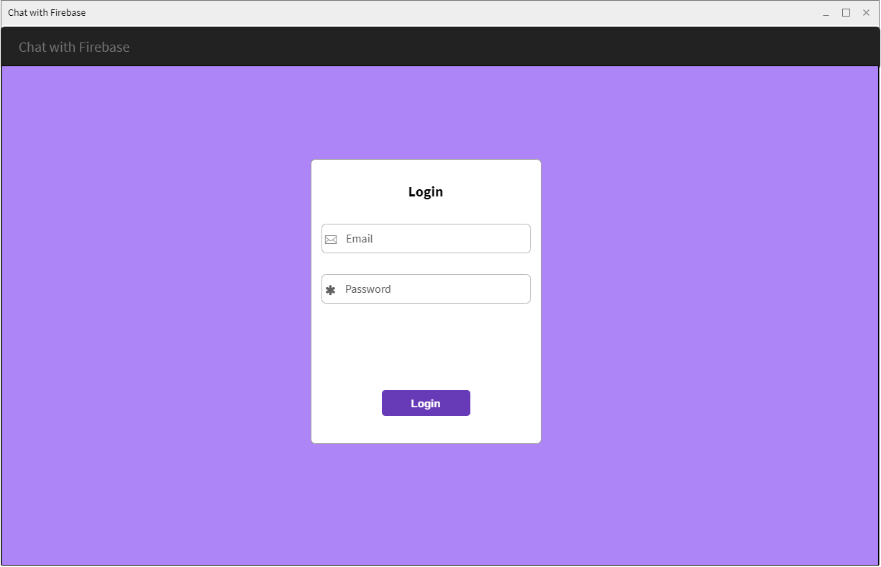


Questa era l’idea iniziale prima di essermi documentata scoprendo che firebase utilizza un database ad oggetti e non relazionale. Questa è quindi la versione definitiva.

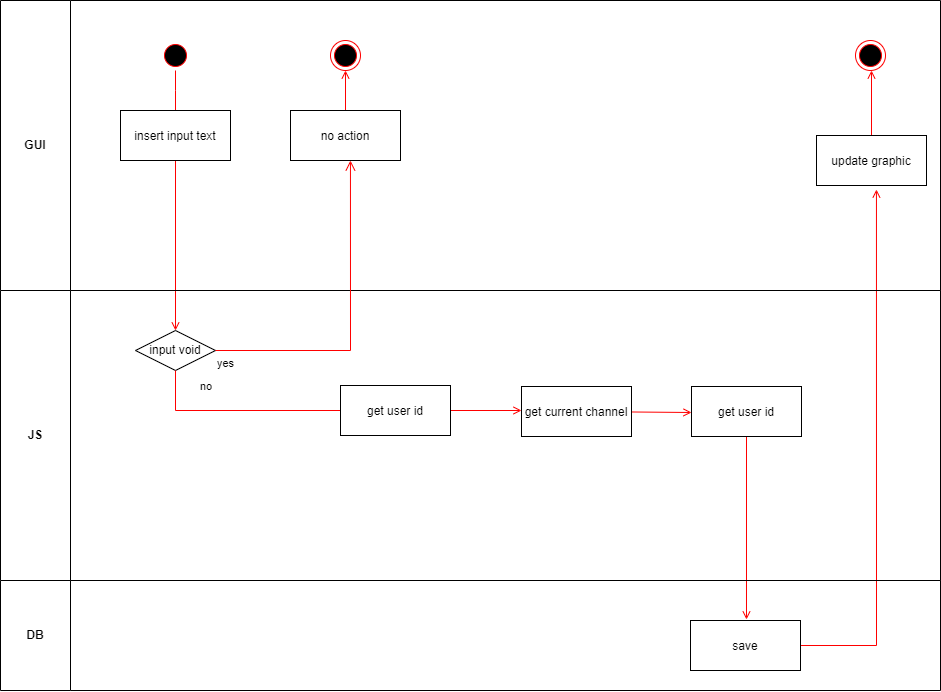


## Design delle interfacce

Ho realizzato la mia progettazione delle interfacce iniziale nel seguente modo.



## Design procedurale



# Implementazione

## Creazione di un progetto con Firebase

Per creare un progetto con Firebase aprire il seguente link: <https://console.firebase.google.com/> e con un account Google è possibile accedere all’interfaccia.

Per poter iniziare a programmare l’applicazione è necessario registrarla ricevendo un id e le chiavi di accesso da inserire nei file per connettersi al database.

Serve anche abilitare l’autenticazione tramite indirizzo e-mail e inizializzare il database Realtime.

## Creazione del progetto

## Index.html

## Chat.html

## Index.js

Queste sono le informazioni che permettono all’applicazione di comunicare con firebase. Queste istruzioni si trovano nelle impostazioni di ogni singola applicazione che comunica con firebase.

// Configurazioni di Firebase

var firebaseConfig = {

    apiKey: "AIzaSyAjJ0bNKGMPoIeU6jbq8SNlkAiIbEt7TL8",

    authDomain: "chat-with-firebase-1dc0f.firebaseapp.com",

    databaseURL: "https://chat-with-firebase-1dc0f-default-rtdb.europe-west1.firebasedatabase.app",

    projectId: "chat-with-firebase-1dc0f",

    storageBucket: "chat-with-firebase-1dc0f.appspot.com",

    messagingSenderId: "699801194182",

    appId: "1:699801194182:web:b5a7617363dd7cc08651c7",

    measurementId: "G-1337V0ESK6"

};

### Immagine che contiene testo Descrizione generata automaticamenteregisterNewUser

Il metodo registerNewUser legge il valore degli input del file index.html con nickname, e-mail e password del nuovo utente. Se la password rispetta le condizioni e non esiste un altro account con lo stesso indirizzo e-mail al database sotto gli utenti viene creato un nuovo oggetto che contiene le informazioni dell’utente. L’opzione admin è di default a false e deve essere impostata manualmente.

### loginUser

Il metodo loginUser legge il valore degli input del file index.html con email e password. In seguito controlla che l’account esista e che l’utente non sia in quel momento bannato e mostra un messaggio di errore. Se invece l’utente esiste e può accedere viene caricata la pagina chat.html.

### logoutUser

Tramite il metodo fornito da Firebase viene chiusa la sessione dell’utente corrente.

### sendMessage

Immagine che contiene testo

Descrizione generata automaticamenteIl metodo sendMessage salva nel database un nuovo messaggio che ha come valore il timestamp salvato nel formato UNIX time.

L’opzione broadcast specifica se il messaggio deve essere visibile su tutti i canali ed’è quindi stato inviato da un account con un ruolo admin. L’uid è un codice univoco per ogni account per poter risalire a chi ha inviato il messaggio.

### viewAllUsers

Il metodo viewAllUsers permette di visualizzare nel dropdown nel modal all’interno della pagina chat.html per poter aggiungere gli utenti quando si sta creando un nuovo canale. Per farlo vado a leggere i dati dal database su tutti gli utenti esistenti.

### addToChannel

Il metodo addToChannel viene scatenato ogni volta che si clicca su un utente nel dropdown per la creazione di un nuovo canale per poi mostrare sopra al dropdown la lista delle persone che sono state aggiunte.

### Immagine che contiene testo Descrizione generata automaticamentecreateChannel

Il metodo createChannel legge il valore degli input all’interno del modal per la creazione del canale. Il database salverà il nome del canale e l’uid di tutti gli utenti che ne fanno parte, facendo attenzione a comprendere anche l’utente che sta creando il canale. Viene anche salvato il tempo dopo il quale i vecchi messaggi devono essere cancellati (rappresentato in ore).

### channelExists

Il metodo channelExists controlla se nel database esiste già un altro canale con lo stesso nome, se è così mostra nel modal un messaggio che avvisa l’utente di modificare il nome.

### deleteOldMessages (da finire)

### loadChannels

Il metodo loadChannels viene richiamato al caricamento della pagina chat.html e mostra nella sidebar laterale tutti i canali di cui l’utente corrente fa parte.

### getChatBox

Il metodo getChatBox ha come parametri nickname, messaggio, floatRight e broadcast. Serve p L’er creare una nuova riga per i messaggi della chat. Il parametro floatRight viene impostata a true se il messaggio da mostrare è stato inviato dall’utente corrente per essere allineato a destra. Il parametro broadcast se impostato a true aggiunge dopo il nickname una stringa per avvisare che il messaggio è stato inviato in broadcast.

### changeChannel

Il metodo changeChannel modifica la variabile globale currentChannel e ricarica i messaggi per aggiornarli a quelli del canale corrente richiamando il metodo reloadMessages.

### showNickname

Il metodo showNickname mostra nella sidebar laterale il nickname dell’utente corrente. Per fare questo è necessario cercare nel database fra gli utenti registrati quale ha l’uid uguale a quello dell’utente della sessione corrente.

### reloadMessages

Il metodo reloadMessages ricarica i messaggi del canale corrente secondo la variabile globale currentChannel. Per farlo legge tutti i messaggi presenti nel database mostrando a schermo solo quelli che corrispondono al canale corrente e utilizza il metodo getChatBox.

### deleteChannel

Il metodo deleteChannel elimina il canale specificato dal parametro channel dal database di firebase e richiama il metodo loadChannels per rimuovere dalla lista il canale cancellato.

Vengono anche cancellati tutti i messaggi contenuti in quel canale.

### openModifyChannel

Il metodo openModifyChannel quando viene cliccato il tasto di modifica di un canale mostra il modal per la modifica del canale.

### modifyChannel (da finire)

### addToBan

Il metodo addToBan viene scatenato ogni volta che si clicca su un utente nel dropdown per il ban di un utente. È possibile bannare un solo utente alla volta. Viene poi mostrato a schermo l’utente selezionato

### Immagine che contiene testo Descrizione generata automaticamentebanUser

Il metodo banUser legge gli input del modal dalla pagina chat.html. L’utente bannato, l’utente che ha bannato, il motivo del ban e la durata vengono salvati nel database.

### loadAdminOption

Il metodo loadAdminOption viene richiamato al caricamento della pagina chat.html e va a leggere dal database se l’utente corrente è admin. Se lo è mostra nella sidebar la possibilità di inviare messaggi in broadcast.

### resizeMessagesDiv

Il metodo resizeMessagesDiv viene richiamato ogni volta che viene ridimensionata la finestra per poter adattare la lunghezza del div della chat rispetto alla dimensione della pagina chat.html.

### onEnterSendMessage

Il metodo onEnterSendMessage viene richiamato al caricamento della pagina chat.html e permette di inviare messaggi in chat premendo il tasto Enter

## Sidebars.js

## Bootstrap.bundle.min.js

## Bootstrap.css

## Sidebars.css

# 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-01  REQ-01 | **Nome:** | Installazione server Firebase |
| **Descrizione:** | Controllare che il server sia online e che gestisca correttamente le richieste al DB | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Aprire la console online di firebase 2. All’invio di un messaggio nell’applicazione verificare che nel database appaia il nuovo record | | |
| **Risultati attesi:** | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-02  REQ-02 | **Nome:** | Autentificazione degli utenti |
| **Descrizione:** | Controllare che gli utenti di possano autentificarsi nella piattaforma. | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Aprire la schermata di login 2. Inserire una mail e una password 3. Controllare che sulla console di firebase si veda il nuovo utente | | |
| **Risultati attesi:** | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-03  REQ-03 | **Nome:** | Gli amministratori devono avere la possibilità di modificare i canali. |
| **Descrizione:** | Gli amministratori devono poter:   1. Creare canali 2. Eliminare canali 3. Inviare messaggi in broadcast 4. Bannare utenti | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | - | | |
| **Risultati attesi:** | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-04  REQ-04 | **Nome:** | Configurazione messaggi |
| **Descrizione:** | L’utente deve poter decidere se i messaggi si devono auto cancellare alla lettura degli altri utenti | | |
| **Prerequisiti:** | Aver eseguito il login nell’applicazione | | |
| **Procedura:** | 1. Autentificarsi con due utenti diversi 2. Inviare un messaggio con l’opzione auto-purge attivata 3. Controllare con l’altro utente che una volta letto il messaggio si cancelli | | |
| **Risultati attesi:** | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-05  REQ-05 | **Nome:** | Deve essere supportata l’intera mappa unicode |
| **Descrizione:** | Deve essere supportata l’intera mappa unicode comprendendo anche alcune emoji che ne fanno parte. | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Accedere all’applicazione con un account 2. Provare ad inviare varie emoji | | |
| **Risultati attesi:** | - | | |

## 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

L’applicazione può essere ampliata aggiungendo per esempio la possibilità di inviare nei canali foto o documenti che possono essere salvati sullo storage online messo a disposizione da firebase.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

## Sitografia

* Wireframe pro gui design
* <https://wireframepro.mockflow.com/editor.jsp?editor=on&spaceid=MdlnWCz73h&bgcolor=white&perm=Create&pcompany=C624bea04c2134b5bb452efc4fff21f7a&ptitle=prova&category=bootstrap4&projectid=MUAx2CiyLmb&publicid=4c02aef144d74813bf24e895a7c3d70f#/page/D2fc7f7c9c5fbabeef99897fd0d66f4cc>

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

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
* …