



UNIVERSITA' DEGLI STUDI DI  
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base  
Corso di Laurea Magistrale in Ingegneria Informatica

Elaborato in Web and Real Time Communication System

## *Chat real time in react native*

Anno Accademico 2022/2023

Candidato

**Dario Riccardi**

**matr. M63000990**

# Contents

<b>1</b>	<b>Introduzione</b>	<b>1</b>
<b>2</b>	<b>Tecnologie</b>	<b>3</b>
2.1	React Native . . . . .	3
2.2	React . . . . .	4
2.3	Firebase . . . . .	7
2.4	Expo . . . . .	8
2.4.1	EAS . . . . .	10
<b>3</b>	<b>Implementazione</b>	<b>11</b>
3.1	Analisi dei casi d'uso . . . . .	11
<b>4</b>	<b>Conclusione</b>	<b>16</b>

# Introduzione

La comunicazione in tempo reale è diventata un elemento essenziale nella nostra società altamente connessa. Questo progetto rappresenta un esempio pratico dell'applicazione di tecnologie moderne per creare un'app di messaggistica in tempo reale utilizzando il framework React Native. Il mio obiettivo principale è stato sviluppare un'app di chat real-time completa e funzionale, in grado di consentire agli utenti di scambiare messaggi istantanei in un ambiente mobile. Attraverso questo progetto, ho voluto raggiungere i seguenti obiettivi:

- **Apprendimento di React Native:** Acquisire competenze pratiche nello sviluppo di applicazioni mobili utilizzando React Native, un framework ampiamente utilizzato per la creazione di app cross-platform.
- **Comunicazione in Tempo Reale:** Implementare un sistema di comunicazione in tempo reale che consenta agli utenti di in-

viare e ricevere messaggi istantanei in modo sincrono.

- **Gestione degli Utenti:** Creare un sistema di gestione degli utenti che consenta agli utenti di registrarsi, effettuare l'accesso in modo sicuro e gestire il proprio profilo.
- **Interfaccia Utente Intuitiva:** Progettare un'interfaccia utente intuitiva e accattivante per offrire un'esperienza utente gradevole e semplice da utilizzare.



Figure 1.1: Logo UninaChat

# Tecnologie

## 2.1 React Native

React Native è un framework open-source sviluppato da Meta che consente di creare applicazioni mobili native per piattaforme multiple utilizzando JavaScript e React. Questa tecnologia si basa sulla libreria React, ampiamente utilizzata per lo sviluppo di applicazioni web, adattata per la creazione di app mobili cross-platform.

Caratteristiche Principali di React Native:

- **Cross-Platform:** Una delle caratteristiche più evidenti di React Native è la sua capacità di scrivere codice una sola volta e utilizzarlo su diverse piattaforme, come iOS e Android. Ciò riduce notevolmente il tempo e gli sforzi necessari per sviluppare e mantenere app per piattaforme multiple.
- **Componenti Riutilizzabili:** React Native utilizza una strut-

tura basata su componenti, dove è possibile creare componenti riutilizzabili che semplificano lo sviluppo e migliorano la manutenibilità del codice.

- **Performance:** React Native utilizza il concetto di "ponte" (bridge) per comunicare tra il codice JavaScript e il codice nativo della piattaforma. Questo approccio consente prestazioni native e un'esperienza utente fluida.
- **Libreria di Componenti:** La community di React Native ha creato una vasta libreria di componenti pronti all'uso, consentendo agli sviluppatori di accedere a un'ampia gamma di funzionalità predefinite senza doverle sviluppare da zero.
- **Hot Reloading:** React Native supporta il "hot reloading", il che significa che è possibile vedere immediatamente gli effetti delle modifiche apportate al codice senza dover ricompilare l'intera app.

## 2.2 React

React è una libreria JavaScript open-source sviluppata da Meta che viene ampiamente utilizzata per la creazione di interfacce utente dinamiche e reattive per applicazioni web. Si basa su un paradigma di programmazione dichiarativo e componenti riutilizzabili, il che lo

rende un'ottima scelta per la costruzione di interfacce utente modulari e facili da mantenere. Caratteristiche Principali di React

- **Componenti:** React si basa su un sistema di componenti, che permette di suddividere l'interfaccia utente in pezzi riutilizzabili e gestibili separatamente.
- **Rendering Efficiente:** React utilizza un motore di rendering virtuale (Virtual DOM) per ottimizzare il processo di aggiornamento dell'interfaccia utente. Questo porta a un miglioramento delle prestazioni e a un'esperienza utente più reattiva.
- **Unidirectional Data Flow:** React promuove un flusso di dati unidirezionale, il che semplifica il tracciamento dei dati e lo stato dell'applicazione.

React ha avuto un ruolo fondamentale per lo sviluppo dell'app.

L'interfaccia utente è stata creata mediante i **componenti** di React personalizzati. Ad esempio sono stati creati dei componenti per gestire la lista dei messaggi, la barra di input dei messaggi e i dettagli del profilo dell'utente.

Grazie a React è stato possibile gestire in maniera agile lo stato dei componenti e delle variabili: viene utilizzato l'**hook useState** per la gestione dello stato delle variabili.

Ha permesso anche la gestione degli eventi generati dall'utente: è stato possibile attraverso l'uso dei **hook useEffect, useLayoutEffect** e

**useCallback.**

Il primo hook consente di eseguire effetti collaterali in componenti funzionali. Gli effetti collaterali sono azioni che possono verificarsi al di fuori del normale flusso di rendering di un componente, come l'accesso a API esterne, l'aggiornamento del DOM, la sottoscrizione a eventi, e così via. Questo hook ha una particolarità, esso accetta anche un secondo argomento che è un array di dipendenze. Queste dipendenze indicano a React quando l'effetto dovrebbe essere eseguito nuovamente. Se l'array delle dipendenze è vuoto, l'effetto viene eseguito solo una volta, subito dopo il primo rendering. Se il contenuto delle dipendenze cambia tra un rendering e l'altro, l'effetto viene eseguito nuovamente. Per quanto attiene il secondo hook si ha che esso è simile a `useEffect` ma viene sincronizzato in fase di aggiornamento prima che vengano applicate le modifiche al DOM o all'interfaccia utente. Questo può essere utile in situazioni in cui si desidera eseguire operazioni che richiedono il layout o la misurazione degli elementi prima che l'utente possa vederli. Infine il terzo hook, ovvero `useCallback`, è un hook di React viene utilizzato per memorizzare una funzione in modo che non venga creata nuovamente ogni volta che il componente viene ri-renderizzato. Questo è utile quando si desidera ottimizzare le prestazioni di un componente React che utilizza funzioni callback come proprietà, specialmente se quelle funzioni non cambiano nel tempo ma vengono passate come dipendenze ai componenti figli.



## 2.3 Firebase

Firebase è una piattaforma di sviluppo mobile e web basata su cloud che offre una vasta gamma di strumenti e servizi per la creazione di applicazioni di alta qualità. Sviluppata da Google, Firebase semplifica molte delle sfide comuni dello sviluppo, inclusa la gestione dell'infrastruttura server, l'autenticazione degli utenti e l'archiviazione dei dati.

Le feature di firebase utilizzate nel progetto sono:

- **Autenticazione:** Firebase offre servizi di autenticazione facili da utilizzare che consentono agli sviluppatori di gestire l'accesso degli utenti in modo sicuro e personalizzabile. Questo è fondamentale per proteggere l'accesso all'app di chat real-time.
- **Cloud Firestore:** Firebase offre anche Cloud Firestore, un database cloud NoSQL altamente scalabile e flessibile, che può essere utilizzato per archiviare dati strutturati in modo avanzato.

Per integrare firebase nel progetto la prima cosa da fare è crearsi un account sul sito di firebase. Dopodiché, dalla console di firebase, bisogna creare un app e abilitare le feature necessarie al progetto. A questo punto nelle impostazioni dell'app è possibile accedere alle variabili di configurazione di firebase.

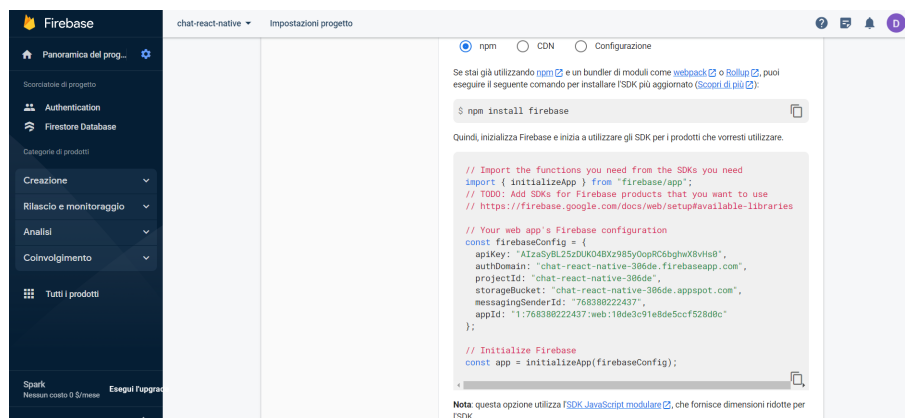


Figure 2.1: Configurazione di firebase da console

## 2.4 Expo

Expo è una piattaforma e un framework open-source per la creazione di applicazioni mobili React Native. Sviluppato per semplificare il processo di sviluppo, Expo offre un set di strumenti, servizi e librerie che consentono agli sviluppatori di concentrarsi sulla creazione di app mobili senza doversi preoccupare di dettagli complessi come la configurazione nativa del progetto.

Le caratteristiche Principali di Expo sono:

- **Ambiente di Sviluppo Unificato:** Expo fornisce un ambiente di sviluppo unificato che consente agli sviluppatori di scrivere codice JavaScript e React Native senza la necessità di configurare manualmente i progetti nativi per iOS e Android.
- **Strumenti di Build e Distribuzione:** Expo offre strumenti di build e distribuzione che semplificano la creazione di pacchetti di distribuzione per iOS e Android, nonché la distribuzione su

app store e app store alternativi.

- **API e Componenti Predefiniti:** Expo fornisce un vasto set di API e componenti predefiniti che coprono una varietà di funzionalità comuni, come la fotocamera, la geolocalizzazione, la notifica push e altro ancora.

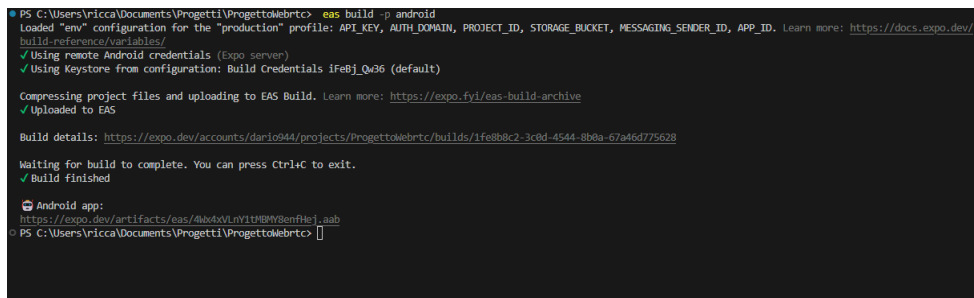
L'utilizzo di expo ha reso lo sviluppo e il testing del codice molto rapido, infatti, tramite l'utilizzo di questo framework è stato possibile evitare di usare simulatori di dispositivi android o ios. Questo ha reso la configurazione del progetto più snella e veloce. E' stato sufficiente avere a disposizione un dispositivo target (ios o android) con l'app di expo già installata sul dispositivo per simulare l'esecuzione dell'applicazione. Tramite la rete locale, l'ambiente di sviluppo carica sul dispositivo target l'applicazione da eseguire.



Figure 2.2: Avvio di della simulazione dell'applicazione attraverso expo

### 2.4.1 EAS

**EAS (Expo Application Services)** è una piattaforma che estende le capacità di Expo, rendendo più agevole il processo di sviluppo, test, build e distribuzione delle applicazioni React Native. L'uso di questa feature di expo rende la pubblicazione negli store abbastanza agile. Per quanto riguarda il mondo android, attraverso l'uso di questa feature è possibile creare l'apk del progetto così da poterlo installare su qualsiasi dispositivo.



```
PS C:\Users\ricca\Documents\Progetti\ProgettoWebRTC> eas build -p android
Loaded "env" configuration for the "production" profile: API_KEY, AUTH_DOMAIN, PROJECT_ID, STORAGE_BUCKET, MESSAGING_SENDER_ID, APP_ID. Learn more: https://docs.expo.dev/build-reference/variables/
✓ Using remote Android credentials (Expo server)
✓ Using Keystore from configuration: Build Credentials ifeBj_Qw36 (default)

Compressing project files and uploading to EAS Build. Learn more: https://expo.fyi/eas-build-archive
✓ Uploaded to EAS

Build details: https://expo.dev/accounts/dario944/projects/ProgettoWebRTC/builds/1fe8b8c2-3c8d-4544-8bda-67a46d775628

Waiting for build to complete. You can press Ctrl+C to exit.
✓ Build finished

📱 Android app:
https://expo.dev/artifacts/eas/4b64xVLnY1tH9M8enfikej.aab
PS C:\Users\ricca\Documents\Progetti\ProgettoWebRTC>
```

Figure 2.3: Creazione apk progetto

# Implementazione

## 3.1 Analisi dei casi d'uso

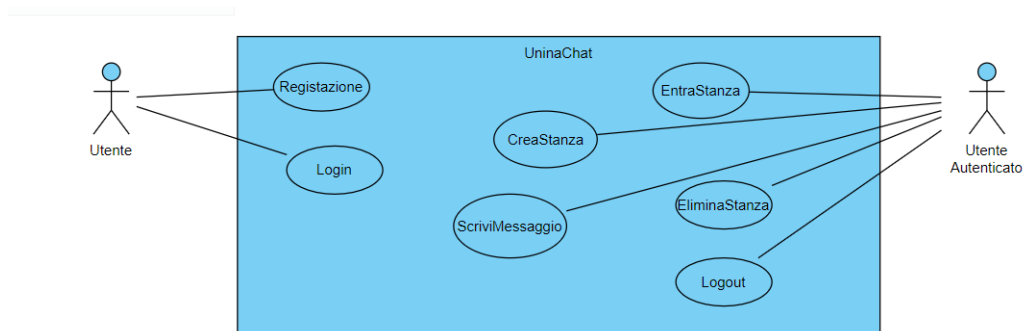


Figure 3.1: Diagramma dei casi d'uso

**UC1.** Registrazione

<b>Attori:</b> Utente
<b>Obiettivo:</b> Creare un nuovo account utente.
<b>Flusso Principale:</b> <ul style="list-style-type: none"> <li>• L'utente avvia l'app.</li> <li>• L'utente seleziona l'opzione di registrazione.</li> <li>• L'utente fornisce email e password.</li> <li>• L'app verifica e registra l'utente nel sistema.</li> <li>• L'utente viene reindirizzato alla schermata di accesso.</li> </ul>

**UC2.** Login

<b>Attori:</b> Utente
<b>Obiettivo:</b> Accedere all'applicazione per iniziare a chattare.
<b>Flusso Principale:</b> <ul style="list-style-type: none"> <li>• L'utente avvia l'app.</li> <li>• L'utente seleziona l'opzione di accesso.</li> <li>• L'utente inserisce le credenziali di accesso (email e password).</li> <li>• L'app verifica le credenziali e permette l'accesso all'utente.</li> <li>• L'utente viene reindirizzato alla schermata principale della chat.</li> </ul>

**UC3.** CreaStanza

<b>Attori:</b> Utente Autenticato
<b>Obiettivo:</b> Creare un nuova stanza.
<b>Flusso Principale:</b> <ul style="list-style-type: none"><li>• L'utente avvia l'app.</li><li>• L'utente seleziona l'opzione di creare una nuova stanza.</li><li>• L'utente fornisce il nome della stanza.</li><li>• L'app verifica la presenza della stanza.</li><li>• L'app registra la stanza nella lista delle stanze accessibili.</li></ul>
<b>Flusso Alternativo:</b> <p>4a. L'utente riceve una notifica di errore se la stanza è già presente.</p>

**UC4.** EntraStanza

<b>Attori:</b> Utente Autenticato
<b>Obiettivo:</b> Entrare in una stanza
<b>Flusso Principale:</b> <ul style="list-style-type: none"> <li>• L'utente avvia l'app.</li> <li>• L'utente seleziona la stanza.</li> <li>• L'utente viene reindirizzato alla schermata dei messaggi.</li> </ul>

#### UC5. EliminaStanza

<b>Attori:</b> Utente Autenticato
<b>Obiettivo:</b> Eliminare una stanza già presente.
<b>Flusso Principale:</b> <ul style="list-style-type: none"> <li>• L'utente avvia l'app.</li> <li>• L'utente seleziona l'opzione di eliminare la stanza.</li> <li>• L'utente fornisce il nome della stanza.</li> <li>• L'app verifica la presenza della stanza.</li> <li>• L'app elimina la stanza dalla lista delle stanze.</li> </ul>
<b>Flusso Alternativo:</b> <p>4a. L'utente riceve una notifica di errore se la stanza non è presente</p>



**UC6.** ScriviMessaggio

<b>Attori:</b> Utente Autenticato
<b>Obiettivo:</b> Scrivere un messaggio nella stanza.
<b>Flusso Principale:</b> <ul style="list-style-type: none"><li>• L'utente avvia l'app.</li><li>• L'utente scrive il messaggio.</li><li>• L'utente invia il messaggio.</li><li>• L'app notifica a schermo il messaggio.</li></ul>

**UC7.** Logout

<b>Attori:</b> Utente Autenticato
<b>Obiettivo:</b> Effettuare il Logout.
<b>Flusso Principale:</b> <ul style="list-style-type: none"><li>• L'utente avvia l'app.</li><li>• L'utente seleziona l'opzione di effettuare il logout.</li><li>• L'app reindirizza l'utente alla pagina di login.</li></ul>

# Conclusione

Con questo progetto ho compreso i principi fondamentali dello sviluppo di applicazioni mobili moderne e delle soluzioni di comunicazione in tempo reale. I punti chiave che sono stati affrontati:

- **Tecnologie Utilizzate:** sono state introdotte le tecnologie principali utilizzate nel progetto, tra cui React Native, Firebase, Expo ed EAS. Queste tecnologie hanno fornito gli strumenti necessari per creare un'app di chat avanzata e cross-platform.
- **Casi d'Uso:** Ho definito una serie di casi d'uso che descrivono in dettaglio come gli utenti interagiscono con l'app di chat real-time. Questi casi d'uso sono serviti da base per la progettazione e l'implementazione delle funzionalità dell'app.
- **Implementazione Tecnica:** Ho esaminato le fasi di implementazione tecnica, comprese la configurazione di Firebase, l'utilizzo

di Expo ed EAS, la gestione degli utenti e la comunicazione in tempo reale. Questi aspetti hanno formato il nucleo dell'applicazione.

- **Flusso di Lavoro Continuo:** L'adozione di un flusso di lavoro continuo con EAS ci ha permesso di automatizzare la compilazione, il testing e il rilascio dell'applicazione, migliorando l'efficienza dello sviluppo.

Le prospettive Future posso essere molteplici: esistono numerose opportunità per migliorare e ampliare l'applicazione. Si possono esplorare ulteriori funzionalità, come l'invio di file multimediali, migliorare la sicurezza e l'esperienza utente, nonché affinare le prestazioni dell'app.