

CoCMessenger



Gabriel Piercecchi
Tosca Pierro
Lorenzo Zulli



Università Politecnica delle Marche
Facoltà di Ingegneria Informatica e dell'Automazione
Anno Accademico 2022 - 2023

Indice

| | | |
|----------|------------------------------------|-----------|
| 1 | Introduzione | 2 |
| 2 | Versione Android | 2 |
| 2.1 | Requisiti funzionali | 2 |
| 2.2 | Requisiti non funzionali | 3 |
| 2.3 | Casi d'uso | 3 |
| 2.4 | Architettura | 4 |
| 2.5 | Database | 5 |
| 2.6 | Mockup | 6 |
| 2.6.1 | Login e Sign Up | 6 |
| 2.6.2 | Home | 7 |
| 2.6.3 | Menu | 7 |
| 2.6.4 | Chat | 9 |
| 2.6.5 | Statistiche giocatore | 10 |
| 2.6.6 | Top giocatori | 11 |
| 2.7 | Sviluppo | 12 |
| 2.8 | Testing | 12 |
| 3 | Versione Flutter | 13 |
| 3.1 | Requisiti funzionali | 13 |
| 3.2 | Requisiti non funzionali | 13 |
| 3.3 | Casi d'uso | 14 |
| 3.4 | Architettura | 14 |
| 3.5 | Database | 14 |
| 3.6 | Mockup | 15 |
| 3.6.1 | Login e Sign Up | 15 |
| 3.6.2 | Home | 16 |
| 3.6.3 | Menu | 16 |
| 3.6.4 | Statistiche giocatore | 18 |
| 3.6.5 | Chat | 18 |
| 3.6.6 | Top giocatori | 19 |
| 3.7 | Sviluppo | 20 |

1 Introduzione

Clash of Clans è un popolare gioco di strategia online che coinvolge milioni di giocatori in tutto il mondo. In questo gioco i giocatori possono creare e gestire il proprio villaggio, formare clan con altri giocatori, partecipare a guerre, sfide e molto altro. Tuttavia, il gioco non offre una funzione di chat integrata che permetta ai giocatori di comunicare tra loro in modo facile e veloce su scala mondiale. Inoltre, il gioco non permette di visualizzare in maniera semplice e veloce le informazioni dettagliate sulle statistiche dei giocatori e dei clan, come il livello, i trofei, le donazioni, le vittorie, le sconfitte, ecc.

Per colmare questa lacuna il nostro progetto propone la realizzazione di un'applicazione, collegata al gioco Clash of Clans, la quale offre ai vari utenti la possibilità di messaggiare tra loro, visualizzare le loro statistiche e, nel caso volessero, visualizzare le classifiche dei Top giocatori e clan in tutto il mondo (oltre a poter controllare i loro dati). L'applicazione sarà compatibile con i principali sistemi operativi mobili, come Android e iOS, e si baserà su una piattaforma cloud per gestire i dati e le comunicazioni. L'obiettivo del nostro progetto è di fornire ai giocatori di Clash of Clans uno strumento utile e divertente per interagire con la comunità del gioco e conoscere le proprie prestazioni.

2 Versione Android

2.1 Requisiti funzionali

L'applicazione che abbiamo proposto è una piattaforma di comunicazione e informazione per i giocatori di Clash of Clans. L'applicazione permette agli utenti di:

- **Registrarsi e accedere utilizzando l'email.**

Per registrarsi gli utenti devono inserire un nome utente, un indirizzo email valido, il loro tag preso direttamente dal proprio account di Clash of Clans e una password. Per accedere, invece, gli utenti devono inserire la propria email e password (salvate alla registrazione). In caso di smarrimento della password è possibile, tramite l'applicazione, inviare al proprio indirizzo email una mail che permette di cambiare la password.

- **Messaggiare con altri giocatori di Clash of Clans.**

Gli utenti possono inviare e ricevere messaggi di testo da altri giocatori se registrati su quest'app.

- **Visualizzare le statistiche del proprio villaggio, di altri giocatori e clan.**

Gli utenti dopo aver inserito il proprio tag giocatore e la propria chiave API possono vedere le statistiche dettagliate del proprio villaggio, tra cui il proprio nome, tag, clan di appartenenza, attuale league e altri elementi.

Gli utenti possono anche visualizzare le statistiche di tutti gli altri utenti registrati nell'app.

- **Visualizzazione classifica.**

In caso si volessero vedere statistiche di altri giocatori o clan è possibile farlo tramite una ricerca su scala mondiale. Gli utenti possono visualizzare di conseguenza le classifiche dei giocatori e clan di tutto il mondo.

2.2 Requisiti non funzionali

I requisiti non funzionali di un'app di messaggistica per il gioco Clash of Clans sono i seguenti:

- **Affidabilità:** L'app deve garantire la consegna dei messaggi in modo tempestivo e corretto, senza perdita o alterazione dei dati. L'app deve anche gestire i possibili errori o eccezioni che possono verificarsi durante la comunicazione, come la mancanza di connessione, la congestione della rete, la corruzione dei dati, etc.
- **Sicurezza:** L'app deve proteggere la privacy e la sicurezza degli utenti, evitando l'accesso non autorizzato ai dati e alle comunicazioni. L'app deve implementare meccanismi di autenticazione attraverso l'utilizzo di Firebase Authentication.
- **Usabilità:** L'app deve essere facile da usare e da capire per gli utenti, offrendo un'interfaccia grafica intuitiva e user-friendly. L'app deve essere compatibile con i principali sistemi operativi mobili, come Android e iOS.
- **Prestazioni:** L'app deve garantire una buona qualità del servizio, offrendo una comunicazione fluida e veloce tra gli utenti.
- **Manutenibilità:** L'app deve essere facile da modificare e aggiornare in caso di cambiamenti nei requisiti o nelle tecnologie.

2.3 Casi d'uso

In questo diagramma, si possono vedere:

- Gli attori sono due: Giocatore e Sistema. Il Giocatore è l'utente dell'app, che vuole comunicare con altri giocatori e visualizzare le statistiche del gioco. Il Sistema è l'app stessa, che offre le funzionalità al Giocatore.
- I casi d'uso sono sei: Registrarsi, Accedere, Messaggiare, Visualizzare statistiche del villaggio, Visualizzare classifiche dei giocatori e clan e infine Visualizzare le statistiche di un giocatore o clan delle classifiche. Questi sono i servizi che il Sistema fornisce al Giocatore per soddisfare le sue esigenze.
- Le relazioni sono diverse:

- Tra Giocatore e Registrarsi c'è una relazione di associazione la quale indica che il Giocatore può registrarsi al Sistema inserendo: nome, tag, email e password.
- Tra Giocatore e Accedere c'è una relazione di associazione la quale indica che il Giocatore può accedere al Sistema inserendo email e password salvate all'atto di Registrazione.
- Tra Giocatore e Messaggiare c'è una relazione di associazione la quale indica che il Giocatore può inviare e ricevere messaggi da altri giocatori tramite il Sistema.
- Tra Giocatore e Visualizzare statistiche del villaggio c'è una relazione di associazione la quale indica che il Giocatore può vedere le statistiche del suo villaggio di Clash of Clans tramite il Sistema (dopo aver inserito la una chiave Api valida).
- Tra Giocatore e Visualizzare classifiche dei giocatori c'è una relazione di associazione la quale indica che il Giocatore può vedere le classifiche dei giocatori e clan di Clash of Clans tramite il Sistema.
- Tra Giocatore e Visualizzare statistiche dei giocatori e clan delle classifiche c'è una relazione di associazione la quale indica che il Giocatore può vedere le statistiche dei giocatori e clan di Clash of Clans mostrati nelle classifiche tramite il Sistema.
- Tra Registrarsi e Accedere c'è una relazione di inclusione, che indica che il caso d'uso Registrarsi include il caso d'uso Accedere, ovvero dopo aver completato la registrazione, il Giocatore accede automaticamente al Sistema.

2.4 Architettura

Questa applicazione è stata sviluppata con Android Studio, un ambiente di sviluppo integrato (IDE) per la creazione di applicazioni Android. Per implementare le View è stato utilizzato il linguaggio di markup XML mentre per descrivere le funzioni delle View e altre funzionalità abbiamo utilizzato Kotlin: il linguaggio ufficiale di Android Studio.

L'architettura MVVM (Model-View-ViewModel), basata su componenti, è stata utilizzata per costruire l'applicazione:

- **Model:** All'interno del package Model sono state inserite tutte quelle classi che vanno a definire tutti i modelli necessari al Sistema.
- **View:** All'interno del package View sono state inserite tutte le classi che si occupano di visualizzare i dati nelle varie schermate dell'App.
- **ViewModel:** All'interno del package ViewModel sono state inserite tutte le classi che manipolano i dati ottenuti dalle chiamate Api o dal Realtime Database (database di Firebase), servendosi anche delle varie classi presenti nel package Model, e li passano alle view.

All'interno di questo package ne è presente anche un altro chiamato Util che contiene le classi adibite alla memorizzazione di costanti, come link, o per gestire eventi particolari, come la mancanza di connessione intern quando ci si trova a navigare nell'applicazione.

Poiché si tratta di una applicazione che è basata su database salvato su Cloud richiede una costante connessione ad internet per funzionare.

2.5 Database

Per fornire un Database alla nostra applicazione abbiamo scelto di utilizzare Firebase: una piattaforma serverless per lo sviluppo di applicazioni mobili e web.

Abbiamo utilizzato Firebase Authentication per la registrazione e autenticazione dell'utente.

Abbiamo fatto uso di Realtime Database, un database che permette di creare e gestire dinamicamente i suoi attributi, per salvare e sincronizzare i dati in tempo reale tra i vari dispositivi degli utenti (nel nostro caso per salvare i dati degli utenti e i messaggi scambiati tra loro).

2.6 Mockup

2.6.1 Login e Sign Up

La schermata di avvio dell'Applicazione è quella del Login. In questa pagina l'applicazione richiede all'utente di eseguire il login o, nel caso non si è ancora registrati, proseguire con la registrazione (Sign Up). Nel caso in cui l'utente si dimentichi la propria password, tramite il pulsante "Forgot Password" presente sulla schermata, è possibile cambiarla inserendo la propria e-mail e cliccando sulla mail temporanea inviata dal Sistema. Tutto ciò è salvato nel Real Time Data Base di Firebase.

Oltre a tutto questo è anche prese un pulsante che porta alla schermata di Tutorial che offre una semplice ma precisa spiegazione su come utilizzare le funzionalità più specifiche dell'App.

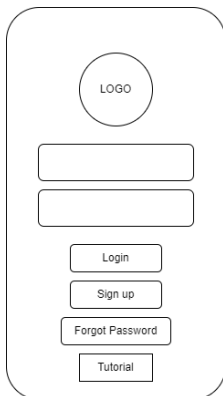


Figure 1: Login

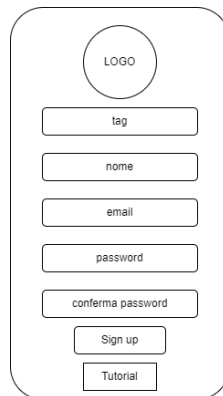


Figure 2: Sign up

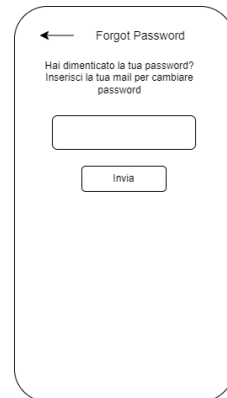


Figure 3: Forgot Password

2.6.2 Home

La pagina immediatamente successiva al Login, o alla registrazione, è la Home dove verranno mostrate le statistiche del proprio villaggio. In fondo allo schermo abbiamo una BottomNavigationView dove ci sono due icone, le quali permettono di ricercare i giocatori (icona di sinistra) e di visualizzare la Top giocatori e clan (icona di destra).

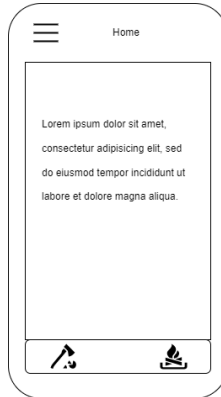


Figure 4: Home

2.6.3 Menu

Sempre dalla Home sarà possibile visualizzare ed eventualmente usare un Hamburger menu (tre linee in alto a sinistra) contenente dei link che porteranno alle pagine di:

- **Home:** Pulsante che riporta alla HomePage.
- **About:** Questa pagina mostrerà i componenti del gruppo che ha portato alla creazione dell'App e il pdf di questa tesina.
- **Tutorial:** Questa pagina offre una breve spiegazione sul corretto inserimento dei dati da parte dell'utente per una corretta registrazione e dove e come poter recuperare il proprio player Tag e l'Api Key che permettono l'ottenimento delle statistiche di tutti i giocatori (compreso l'utente appena loggato).
- **Api Key:** Questa pagina è fondamentale poiché permette l'inserimento della chiave necessaria al funzionamento delle Api. E' contenuta anche qui una breve guida su come ricavare la Key.

N.B.:La chiave dovrà essere inserita ogni qual volta dovesse cambiare l'indirizzo IP del dispositivo utilizzato per eseguire l'applicazione (nel caso non cambiasse non sarà necessario poiché viene memorizzata all'interno del Realtime Database).

- **Change Tag:** Questa pagina offre la possibilità all'utente di poter aggiornare il proprio TAG di Clash of Clans nel caso lo abbia inserito in maniera errata alla registrazione.
- **Logout:** Questo pulsante permette di uscire dall'applicazione e tornare alla pagina di login.

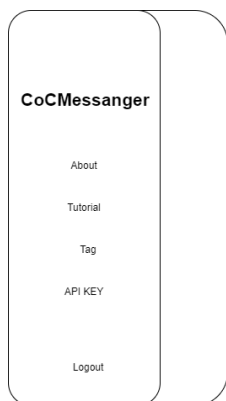


Figure 5: Menu

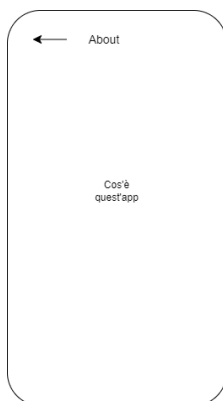


Figure 6: About

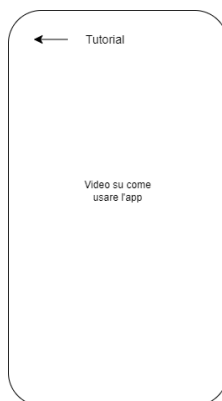


Figure 7: Tutorial

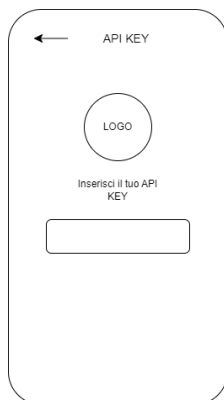


Figure 8: Api Key



Figure 9: Tag

2.6.4 Chat

A seguito del click sull'icona di sinistra presente sulla BottomNavigationView la prima pagina inerente alla messaggistica è la Chat List. Questa pagina si occupa di mostrare e ricercare tutti gli utenti registrati nell'App. Anche in questo caso è presente un menu (tre puntini in alto a destra) che offre la possibilità di tornare alla HomePage o effettuare il Logout.

Cliccando uno degli utenti si entra nell'effettiva pagina di messaggistica. Qui gli utenti possono scambiarsi privatamente dei messaggi.

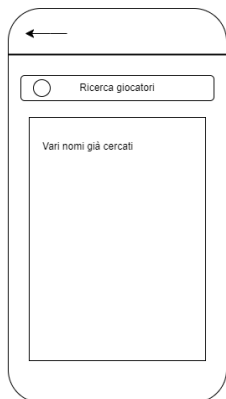


Figure 10: Ricerca

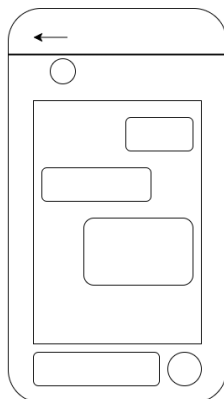


Figure 11: Chat

2.6.5 Statistiche giocatore

Cliccando sul menu (tre puntini in alto a destra situato nella pagina di messaggistica) è possibile aprire la pagina contenente le statistiche dell'utente con cui si sta scrivendo.

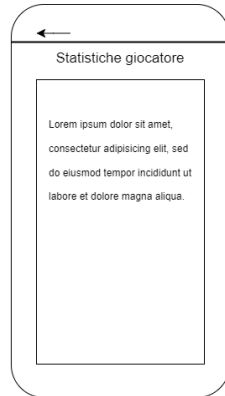


Figure 12: Statistiche

2.6.6 Top giocatori

Nel caso in cui si clicca l'icona di destra della BottomNavigationView, si apre la pagina contenente la Classifica dei Top giocatori, buildera (una sotto-categoria dei players) e i clan. In Kotlin è possibile ricercare le classifiche per Stato e, tramite una barra di ricerca, trovare il player o clan desiderato. In Flutter invece questa pagina mostra la Classifica dei Top Players in Italia. In entrambe le pagine delle due applicazione (Flutter e Kotlin) è sempre possibile visualizzare le statistiche di ogni elemento presente nelle classifiche.

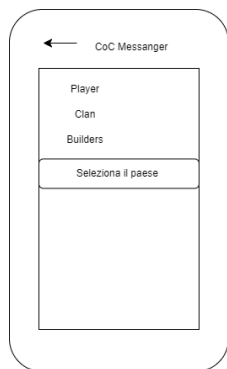


Figure 13: Top giocatori

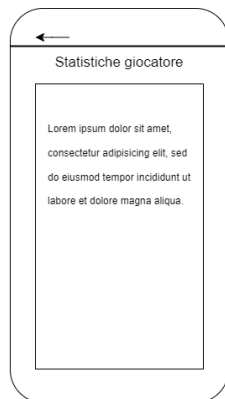


Figure 14: Statistiche

2.7 Sviluppo

Lo sviluppo è partito cominciando a creare le pagine di login e registrazione. Questo ha comportato la creazione dell'applicazione lato firebase in modo da poter interfacciare i metodi di autenticazione tra le pagine e Firebase Authentication.

Una volta eseguito con successo questo collegamento sono state create tutte le pagine relative alla messaggistica. Facendo ciò sono stati istanziati tutti i metodi necessari alla creazione, modifica e visualizzazione dinamica degli utenti e dei loro messaggi. Questo ha portato all'utilizzo del database dinamico Real-time Database.

La parte sviluppata alla fine è stata quella dedicata a visualizzare le statistiche dei giocatori e, quindi, all'utilizzo delle API di Clash of Clans poiché sussisteva, e sussiste tutt'ora, il problema legato alla chiave per usare le API. Questo problema consiste nel dover ricreare ad ogni cambiamento dell'indirizzo IP del dispositivo utilizzato per usare l'App la chiave da usare nelle chiamate URI (per ottenere i dati per le statistiche). Alla fine la soluzione è stata quella di mostrare ad ogni utente, nel Tutorial, come ricreare la chiave tramite delle credenziali preformite. Ovviamente il limite di questa soluzione si presenta nel caso in cui molti utenti dovessero utilizzare l'applicazione poichè vi è un limite sul numero di chiavi creabili.

2.8 Testing

Le classi e i metodi sottoposti ai test sono stati:

- **Login.**

Sono stati creati 2 metodi, uno con delle credenziali corrette, e l'altro con delle credenziali errate. Il primo metodo ha completato il test in maniera positiva riuscendo ad "accedere" all'App mentre il secondo ha fallito il test mostrando un errore nelle credenziali inserite (come era previsto che facesse).

- **Sign Up.**

E' stato creato un metodo che simula l'iscrizione di utente fornendo dei dati fittizi. Se i dati fittizi sono corretti l'iscrizione avviene in caso contrario il test, come previsto, fallisce.

3 Versione Flutter

3.1 Requisiti funzionali

L'applicazione in Flutter presenta un contenuto semplificato rispetto a quello presente nell'app Android. L'applicazione permette agli utenti di:

- **Registrarsi e accedere utilizzando l'email.**
Per registrarsi gli utenti devono inserire un nome utente, un indirizzo email valido, il loro tag preso direttamente dal proprio account di Clash of Clans e una password . Per accedere, invece, gli utenti devono inserire la propria email e password (salvate alla registrazione). In caso di smarrimento della password è possibile, tramite l'applicazione, inviare al proprio indirizzo email una mail che permette di cambiare la password.
- **Messaggiare con altri giocatori di Clash of Clans.**
Gli utenti possono inviare e ricevere messaggi di testo da altri giocatori se registrati su quest'app.
- **Visualizzare le statistiche del proprio villaggio, di altri giocatori e clan.**
Gli utenti dopo aver inserito il proprio tag e la propria chiave API possono vedere le statistiche dettagliate del proprio villaggio, tra cui il proprio nome, tag, clan di appartenenza, attuale league e altri elementi . Gli utenti possono anche visualizzare le statistiche di tutti gli altri utenti registrati nell'app.
- **Visualizzazione classifica.**
In caso si volessero vedere statistiche di altri giocatori, oltre a quelli iscritti all'app, è possibile farlo tramite una ricerca su scala nazionale. Gli utenti possono visualizzare di conseguenza la classifica dei giocatori italiani.

3.2 Requisiti non funzionali

Sono del tutto analoghi a quelli dell'app fatta in kotlin:

- **Affidabilità**
- **Sicurezza**
- **Usabilità**
- **Prestazioni**
- **Manutenibilità**

3.3 Casi d'uso

Sono del tutto analoghi a quelli dell'app fatta in kotlin

- Abbiamo l'attore Giocatore, che vuole comunicare con altri giocatori e visualizzare le statistiche del gioco, abbiamo poi l'attore Sistema che offre le diverse funzionalità al giocatore.
- Abbiamo sei casi d'uso: Registrarsi, Accedere, Messaggiare, Visualizzare statistiche del villaggio, Visualizzare classifica dei giocatori italiani e infine Visualizzare le statistiche di un giocatore. Questi sono i servizi che il Sistema fornisce al Giocatore per soddisfare le sue esigenze.
- Le relazioni sono sempre tra Giocatore e uno dei sei casi d'uso, in modo del tutto analogo all'app in kotlin.

3.4 Architettura

Analogamente all'App in kotlin, anche qui si segue l'architettura tramite pattern Model-View-ViewModel (MVVM) che separa la logica di funzionalità dalla presentazione dell'interfaccia utente.

Il Model rappresenta i dati e le regole di funzionalità dell'app, il ViewModel trasforma i dati dal modello e li rende disponibili alla View tramite flussi di dati, e la View rappresenta l'interfaccia utente dell'app senza contenere alcuna logica o stato. Per implementare il MVVM in Flutter, abbiamo separato le classi in 3 cartelle: View, del Model e del View-Model.

Poichè si tratta di una applicazione che basata su database salvato su Cloud richiede una costante connessione ad internet per funzionare.

3.5 Database

Per fornire un meccanismo di salvataggio dati e gestione delle autorizzazioni alla nostra applicazione abbiamo scelto di utilizzare sempre Firebase.

Abbiamo utilizzato Firebase Authentication per la registrazione e autenticazione dell'utente.

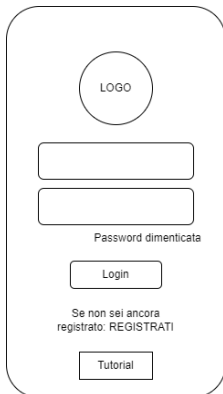
Abbiamo fatto uso di Realtime Database come database dinamico per salvare e sincronizzare i dati in tempo reale tra i vari dispositivi.

3.6 Mockup

3.6.1 Login e Sign Up

Appena l'app viene avviata richiede di fare il login, o nel caso in cui non si è registrati, proseguire con la registrazione. Nel caso ci si dimentichi la password la pagina di Login, tramite il pulsante "Forgot Passowrd" offre la possibilità di cambiarla tramite l'uso di una mail temporanea (dopo aver inserito l'e-mail a cui è collegata la password smarrita).

Tutto i dati vengono salvati nel Realtime DataBase di Firebase.



Logo

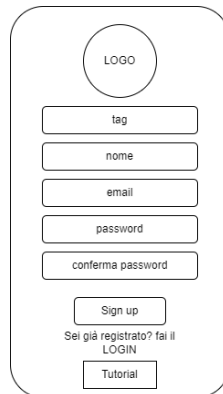
Password dimenticata

Login

Se non sei ancora registrato: REGISTRATI

Tutorial

Figure 15: Login



Logo

tag

nome

email

password

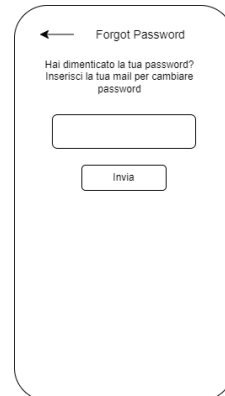
conferma password

Sign up

Sei già registrato? fai il LOGIN

Tutorial

Figure 16: Sign up



← Forgot Password

Hai dimenticato la tua password?
Inserisci la tua mail per cambiare password

Invia

Figure 17: Forgot Password

3.6.2 Home

Nella Home vengono mostrate le statistiche del proprio villaggio. In fondo alla pagina abbiamo una `CurvedBottomNavigationView` con all'interno tre icone ad indicare la Home, la ricerca dei giocatori e la visualizzazione della top giocatori Italiani di Clash of Clans.

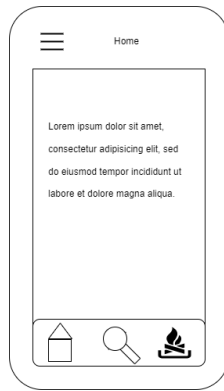


Figure 18: Home

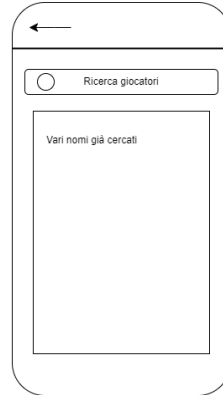


Figure 19: Ricerca

3.6.3 Menu

Sempre dalla Home sarà possibile visualizzare ed eventualmente usare un Hamburger menu (situato in alto a sinistra).

In questo menu ci saranno dei link che ti porteranno alle pagine di:

- **About:** Questa pagina mostrerà i componenti del gruppo che ha portato alla creazione dell'App.
- **Tutorial:** Questa pagina offre una breve spiegazione sul funzionamento dell'Applicazione.
- **Tag:** Questa pagina offre la possibilità all'utente di poter aggiornare il proprio TAG di Clash of Clans nel caso lo abbia inserito in maniera errata alla registrazione.
- **API KEY:** Pagina fondamentale poichè permette l'inserimento della chiave necessaria al funzionamento delle Api.
N.B.:La chiave dovrà essere inserita ogni qual volta dovesse cambiare l'indirizzo IP del dispositivo utilizzato per eseguire l'applicazione (nel caso non cambiasse non sarà necessario poichè viene memorizzata all'interno del Realtime Database).
- **Logout:** Questo pulsante permette di uscire dall'applicazione e tornare alla pagina di login.

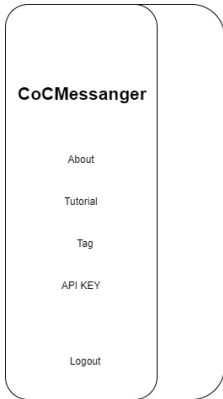


Figure 20: Menu

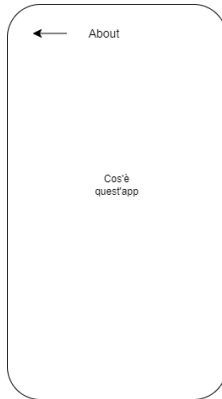


Figure 21: About

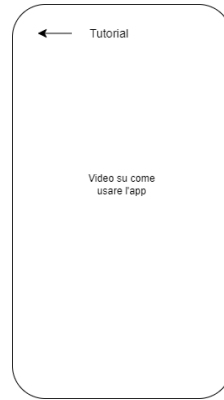


Figure 22: Tutorial

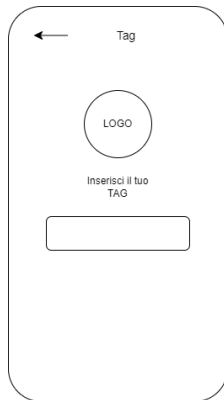


Figure 23: Tag

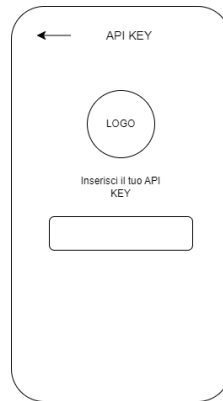


Figure 24: Api Key

3.6.4 Statistiche giocatore

Cliccato sull'icona della ricerca dei giocatori nel `CurvedBottomNavigationView` e, successivamente ricercato e/o cliccato un altro utente iscritto, è possibile visualizzare le sue statistiche.

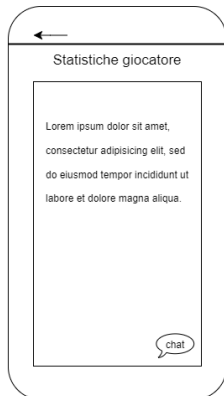


Figure 25: Statistiche

3.6.5 Chat

Una volta cliccato un utente nella pagina di ricerca e visualizzate le sue statistiche è possibile, tramite il pulsante situato nella zona in basso a destra della pagina, scriverci liberamente.

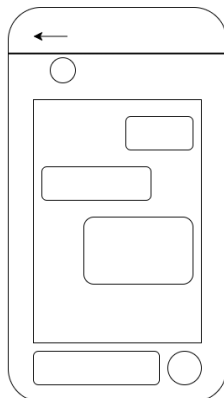


Figure 26: Chat

3.6.6 Top giocatori

In Flutter abbiamo la possibilità di visualizzare solo la top dei giocatori italiani su Clash of Clans (tramite l'apposito pulsante sulla `CurvedBottomNavigationView`). Anche in questo caso è possibile visualizzare le statistiche di ogni singolo membro della classifica.

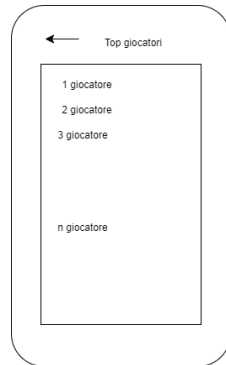


Figure 27: Top

3.7 Sviluppo

Anche in questo caso il primo passo è stato creare le pagine relative al Login e all'iscrizione (SignUp) e, quindi, collegare l'applicazione a Firebase (Firebase Authentication e Realtime database). Il metodo di collegamento, rispetto all'app in Kotlin, è stato differente poichè si è dovuto utilizzare quasi esclusivamente codice da riga di comando.

Successivamente sono state implementate le pagine di messaggistica ed infine le classi inerenti alla gestione delle chiamate alle API per la ricezione delle statistiche di tutti gli utenti.

N.B.: Le API di Clash of Clans per essere utilizzate necessitano una chiave creata utilizzando l'indirizzo IP del dispositivo sul quale si vogliono usare. Poichè l'indirizzo IP cambia dopo un determinato lasso di tempo è stato necessario creare una Pagina dove l'utente, una volta autenticato, può andare a inserire l'API KEY creata col suo indirizzo (come spiegato nella pagina Tutorial) sul sito Ufficiale delle API di Clash of Clans. Fatto ciò potrà avere accesso alle proprie statistiche e a quelle dei giocatori da lui cercati, nonchè visualizzare la top giocatori italiana.

```
//creare un nuovo utente
Future<UserCredential> signUpWithEmailAndPassword(String email, password, name, tag) async{
  try {

    UserCredential userCredential = await _firebaseAuth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );

    // Aggiungi i dati dell'utente nel database Realtime
    if (userCredential.user != null) {
      final userId = userCredential.user!.uid;
      var apiKey = "";
      _database.reference().child('user').child(userId).set({
        'uid': userCredential.user!.uid,
        'email': email,
        'name': name,
        'tag' : tag,
        'apiKey' : apiKey,
      });
    }
    return userCredential;
  } on FirebaseAuthException catch (e){
    throw Exception(e.code);
  }
}
```

Figure 28: Registrazione

Questo metodo ci permette di salvare nel nostro RealTime database di Firebase le credenziali dell'utente.

```
//accedere
Future<UserCredential> signInWithEmailAndPassword(String email, String password) async {
  try{
    //accede
    UserCredential userCredential = await _firebaseAuth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
    return userCredential;
  } on FirebaseException catch (e){
    throw Exception(e.code);
  }
}
```

Figure 29: Login

Questo codice rende possibile l'interfacciamento a Firebase Authentication per l'autenticazione e quindi fare in modo che l'utente possa accedere tramite l'utilizzo dell'e-mail e password usate all'iscrizione.

Un altro aspetto che può essere analizzato è la `CurvedBottomNavigation` per come si collega alle varie pagine: in base all'icona cliccata il Sistema reindirizza l'utente alla pagina desiderata.

```
//questo seleziona l'indirizzo per controllare la bottom nav bar
int _selectIndex = 0;

//Questo metodo aggiorna l'indirizzo selezionato
//quando l'utente clicca sulla bottom bar
void navigationBottomBar(int index){
    setState(() {
        _selectIndex = index;
    });
}

//le pagine da mostrare
final List<Widget> _pages = [
    //la pagina di home dove visualizzi le tue statistiche
    HomePage(playerTag: ' '),

    //pagina per ricercare gli altri utenti
    SearchPage(),

    //pagina per visualizzare la top 50 players
    TopPage(),
];
```

Figure 30: Navigation Bottom Bar

Può essere anche descritta la modalità in cui è stato possibile collegare il DB di firebase alla modifica/inserimento del tag e dell'api key: utilizzando l'UID (unique identifier) dell'Utente loggato è possibile eseguire le azioni precedentemente citate senza danneggiare altri Utenti.

Oltre a questo può essere osservata la chiamata alle API di Clash of Clans usando la chiave fornita dall'Utente.

```
DatabaseEvent tagEvent = await databaseReference
    .child('user')
    .child(uid)
    .child('tag')
    .once();

if (tagEvent.snapshot.value != null) {
    rawTag = tagEvent.snapshot.value.toString();
} else {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('E' andato storto qualcosa nel recupero del TAG'))
    );
}

// Allo stesso modo, ottieni il DatabaseEvent per apiKey
DatabaseEvent apiKeyEvent = await databaseReference
    .child('user')
    .child(uid)
    .child('apiKey')
    .once();

if (apiKeyEvent.snapshot.value != null) {
    String finalTag = rawTag.replaceAll("#", "%23");
    var apiKey = apiKeyEvent.snapshot.value.toString();

    final finalUri = firstUrl + finalTag;

    final response = await http.get(
        Uri.parse(finalUri),
        headers: {'authorization': 'Bearer $apiKey'},
```

Figure 31: Api e Tag