Versione 1.0

10/02/2020



Presentato da: Marco dal zovo - 931121

APPLICAZIONE ANDROID online chess

corso di programmazione– a.a. 2019/2020

introduzione

*Online Chess* è un’applicazione Android che permette di affrontare altri giocatori al gioco degli scacchi, tenendo traccia delle loro vittorie e sconfitte per classificarli. Lo scopo dell’applicazione è quello di rendere possibile giocare ad una partita a scacchi ovunque attraverso dei semplici passaggi preliminari.

Per il corretto funzionamento dell’applicazione è stata sviluppata – in sede separata – una parte server che permetta la sincronizzazione dei dati di gioco.

LE ACTIVitY

Di seguito una lista delle **activity** presenti all’interno del progetto Android ed un’analisi dettagliata:

* *LauncherActivity*
* *MainActivity*
* *GameActivity*

# Launcher activity

Viene visualizzata appena avviata l’applicazione mostrando un indicatore di progresso mentre vengono caricate le risorse necessarie al completo funzionamento dell’app. In seguito consente l’autenticazione dell’utente tramite una procedura con sistema OAuth2 di Google, detta **Google Sign-In Flow**.

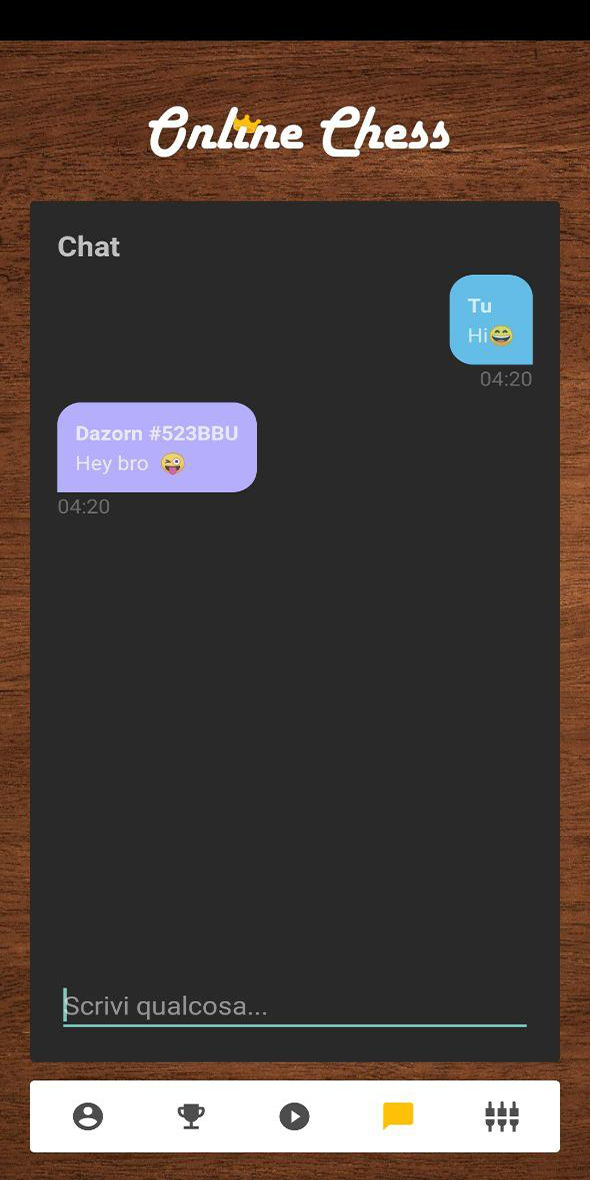
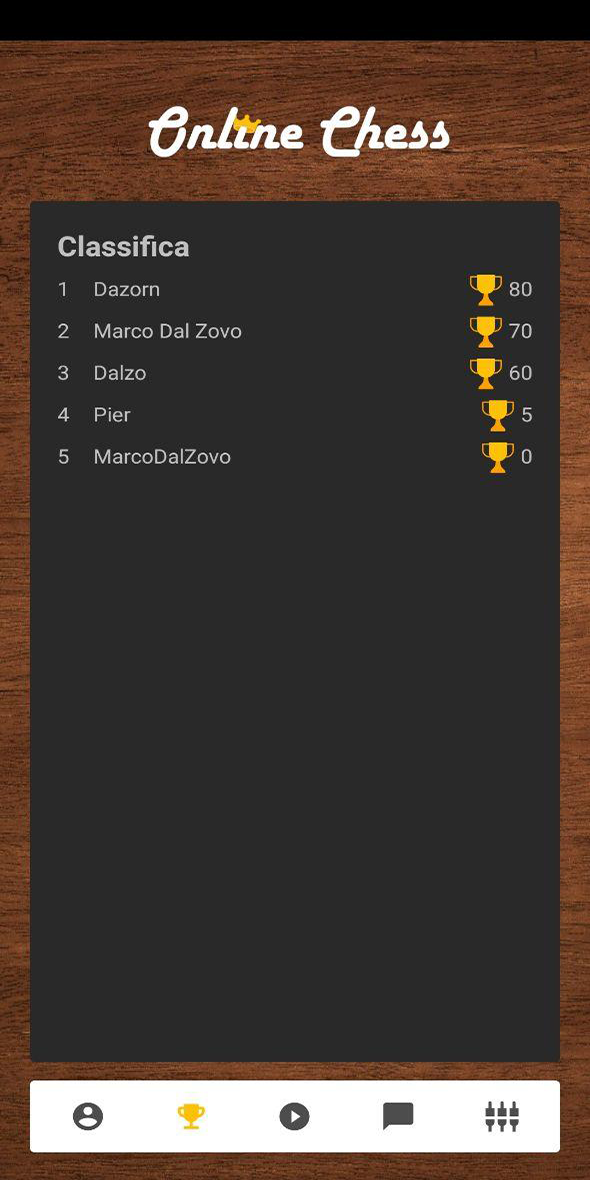
Infine, le credenziali ottenute dall’utente autenticato vengono inviate al server per confermarne l’identità e la corrispondenza univoca con una risorsa utente.

# main activity

È l’activity principale dell’applicazione, composta da un **ViewPager** (che consente lo scrolling in orizzontale dei contenuti) formato – a sua volta – da più **Fragment**:

* *MainFragment*
* *ChatFragment*
* *ProfileFragment*
* *RankingsFragment*
* *SettingsFragment*

Ogni fragment ha vita propria ed indipendente dagli altri, onde evitare un consumo di risorse eccessivo rispetto alle interazioni dell’utente. E’ possibile accedere ai fragment tramite **gesture** di swipe oppure toccando le relative icone presenti nel menù orizzontale sottostante.

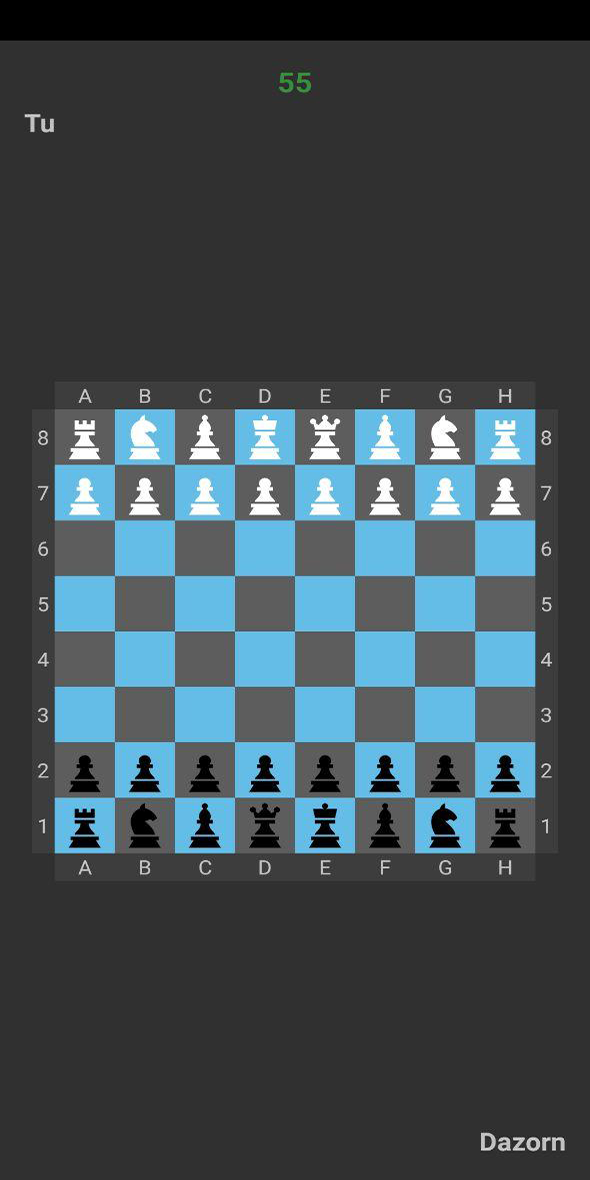


# game activity

Contiene l’ambiente di gioco ed è accessibile solamente in presenza di una partita valida da giocare. Onde evitare chiusure improvvise dell’applicazione, disabilita qualsiasi tentativo di modificare lo stack delle activity aperte.

È composta, nel dettaglio, da due *scacchiere* per garantire un corretto posizionamento delle unità di gioco, un *countdown* per mostrare il tempo rimanente alla fine del turno di gioco e due *liste orizzontali* per renderizzare le pedine catturate da ciascun giocatore.

La posizione delle pedine viene affidata totalmente al server remoto, infatti l’applicazione si limita a visualizzarle graficamente nella loro posizione assegnata, evitando eventuali *mosse scorrette* da parte del giocatore. Inoltre, ad ogni fine turno viene inviato al server – insieme ad altri parametri - l’array delle unità presenti a schermo, per controllarne la validità.



API LAYER

Per garantire una corretta autenticazione dell’utente durante l’avvio dell’applicazione (e per fornire dati in tempo reale, quando richiesti) è presente un livello di comunicazione tra l’applicazione ed il server remoto. Questo livello (presente nella classe pubblica **APIHelper**) espone dei metodi pubblici – e statici – richiamabili in qualsiasi parte logica dell’applicazione che forniscono degli **handler** implementabili per catturare eventuali eventi di risposta. In caso di errore viene segnalato all’utente un errore generico di comunicazione e fatta riavviare l’applicazione, onde evitare malfunzionamenti di quest’ultima.

Per effettuare le richieste tramite protocollo http viene utilizzata la libreria **http** fornita da *Apache*.

SOCKET LAYER

Il cuore dell’applicazione è all’interno della classe singleton **SocketHelper**. L’implementazione effettuata permette di garantire la presenza di un’**unica** istanza di quest’ultima per tutta la durata del ciclo di vita dell’applicazione. Infatti per garantire una comunicazione bilaterale e biunivoca tra server e client era strettamente necessaria una singola connessione per applicazione avviata, onde evitare eventuali perdite di messaggi tra le due parti.

A questo scopo, l’inizializzazione della comunicazione **socket** avviene all’interno del costruttore privato, che viene richiamato una sola volta in seguito all’autenticazione dell’utente. La classe espone dei metodi pubblici, principalmente di due tipi:

* Metodi che emettono dei messaggi verso il server
* Metodi che rimangono in ascolto per eventuali messaggi spediti dal server

Al fine di garantire la persistenza della connessione tra server e client – senza la quale l’applicazione potrebbe andare in contro a gravi problemi – la classe singleton necessita come parametro l’**activity** dalla quale viene recuperata l’istanza. Infatti, quest’ultima verrà memorizzata in una variabile privata ed utilizzata nell’eventualità di una possibile caduta della comunicazione bilaterale per riavviare l’applicazione stessa.

AUTO-RESTART

Come anticipato nel paragrafo precedente, l’applicazione ha un sistema di **auto-restart** quando viene a mancare una connessione ad Internet stabile e funzionante. Ovviamente nel caso in cui l’utente stia effettuando una partita proprio nel momento in cui la connessione cade, è prevista anche una procedura per riprendere la partita e riportare l’applicazione allo stato in cui era prima del riavvio forzato.

Questa procedura può essere effettuata in qualsiasi contesto dell’applicazione utilizzando un **Runnable**, ovvero un’azione specifica che viene registrata e messa in coda di esecuzione sul **thread principale** (ottenuto chiamando *getMainLooper()* rispettivamente al contesto dell’activity corrente) riservato al ciclo di vita dell’applicazione, tutto ciò facendo uso di un **handler**.

CONCLUSIONE

L’applicazione **Online Chess** sarà presto disponibile su **Play Store** della piattaforma Android (attualmente è in stato di *attesa di pubblicazione*) accessibile gratuitamente. Inoltre è disponibile l’intero codice sorgente dell’applicazione a questo indirizzo:

<https://github.com/Dazorn96/node-chess-android>