**Embeh(dded)Chat**

Prozap

Manfè Alessandro, Steffenel Filippo & Zamai Andrew

Completare un progetto richiede di:

1. effettuare una **ricerca bibliografica** sulla tecnologia,
2. scrivere un **report** contenente le informazioni raccolte con la ricerca,
3. progettare una o più semplici **app** che implementino i punti principali illustrati nel report.

Il report deve includere una **bibliografia** che elenchi le fonti della ricerca effettuata.  
Il report **non deve basarsi su un'unica fonte**.

Non è consentito effettuare operazioni di "taglia e incolla" dalle fonti: il report deve essere una **rielaborazione originale** delle informazioni trovate, **non un riassunto o una copia**.  
Poiché gli argomenti sono molto vasti, ciascun gruppo deve eseguire una **selezione del materiale da presentare**: in altre parole, deve scegliere che **taglio dare al repor**t, quali informazioni includere e quali escludere.

Nelle app è **vietato utilizzare senza rielaborazioni il codice sorgente di esempio** che si può trovare online.

Nelle app è **vietato utilizzare librerie che non fanno parte dei framework di Android** a meno che non si sia prima ottenuto il consenso scritto del docente. (FIREBASE?)

Per avere la percezione dell'impegno necessario a completare il progetto, si rifletta su quanto segue. "Programmazione di sistemi embedded" è un insegnamento da 9 CFU, quindi ciascuno studente che lo voglia superare deve investire, nominalmente, 225 ore di lavoro (25 ore per CFU). Di queste, 72 ore sono dedicate a seguire le lezioni frontali. Stimiamo che altre 80 ore (2 settimane di lavoro a tempo pieno, 8 ore al giorno dal lunedì al venerdì) siano necessarie per lo studio individuale del materiale presentato a lezione: rimangono **per il progetto 73 ore**, che è una quantità di lavoro sostanzialmente pari a quella richiesta per realizzare una tesina da 3 CFU. **Questo impegno è richiesto da ciascuno dei membri di un gruppo.**

**Notifications A11**

Le notifiche sono una delle funzionalità più riconoscibili di Android: una funzionalità copiata nella sua essenza da altre piattaforme, e aggiornata ogni anno introducendo continui raffinamenti.

**Indicazioni per la realizzazione del progetto**

Report: si fornisca una **panoramica sulle notifiche in Android**, sia dal punto di **vista estetico** (struttura di una notifica, funzionalità visibili all'utente) che da quello **tecnico** (funzionalità accessibili al progettista). Fatto questo, si descrivano in dettaglio le **novità introdotte con Android 11: nuove Conversation Notifications, Chat Bubbles, riposizionamento dei Media Controls**.  
App: si implementi **una app che esemplifichi i principali aspetti del sistema di notifiche in Android, nonché tutte le novità introdotte in Android 11.**

**Per iniziare la ricerca bibliografica**

https://developer.android.com/guide/topics/ui/notifiers/notifications  
https://material.io/design/platform-guidance/android-notifications.html  
https://developer.android.com/about/versions/11/features

**Minimum SDK version:** API level 30 (permetterebbe uso su 90 % dispositivi, consigliato dal libro, il povero Andrew avrebbe un dispositivo fisico per testare)

NOTA: usare una versione 21 può causare l’impossibilità di usare determinate funzioni introdotte in versioni successive: soluzioni--> aumento min SDK, uso if/else per verificare Android version (pag 140), USARE sempre Jetpack libraries quando possibile per migliore retroconpatibilità.

**Target SDK version:** API level 30 (Android 11) (dobbiamo implementare tutte le nuove Notification features)

**Compile SDK version:** API level 30

Nome Progetto? SendApp

Repository su Github? (pro: molto usato in ambito lavorativo)

Commenti codice italiano o inglese? Inglese

Usare KDoc (identico JavaDoc) per documentazione codice (SOLO per funzioni complicate)

CamelCase sintax

**Struttura applicazione:**

Master/Detail Flow: 2 activity (lista chat + dettaglio singola chat)

Le activity sono separate se su smartphone;

L’activity detail è contenuta come fragment affiancata alla list se su dispositivo tablet.

List Activity contiene una RecyclerView

Anche ogni activity detail conterrà un recyclerView per mostrare tutti i messaggi (simulare con immagine? icona messaggio)

Autentificazione iniziale: un’activity a sé stante dove permette di effettuare il login, se non ho credenziali mi rimanda ad un’altra activity per la registrazione

(Snellire tutto facendo direttamente 4 utenti: noi + prof), in caso aggiungere solo dopo queste due activity

**Implementazione del sistema di conversazione:**

* chat server
* database (LiveDataKTX e ReactiveStreamsKTX)
* Worker e PeriodicWorkRequest per verificare la presenza di nuovi messaggi e notificare l’utente (periodic ogni 15 minuti ☹)
* salvo ultimo messaggio visualizzato in SharedPreferences e verifico quali non ho visto dopo quel messaggio
* Firebase

**Notifying the User**

* pag. 563 BigNerdRanchGuide
* API level >= 26 richiedono creazione di (almeno) un Channel (NotificationCompat class per gestione semplice delle differenti versioni)
* Dove? All’interno della funzione doWork() di una sottoclasse di Worker, passata a sua volta ad un PeriodicWorkRequest
* PeriodicWorkRequest intervallo minimo ogni 15 minuti! (come risolvere?)
* Problema: le notifiche non devono comparire quando l’app è in uso (distrazione per utente + rischio di aprire altra activity) --> broadcast intents and receiver (capitolo 28)
* x

ID messaggio

from user

to user

data

tipo (testo, immagine, video)

byteA: dove salvo quello che devo inviare < 4GB

CONSEGNA STIMATA: 1 Giugno 2021

5 settimane

ANDREW: scheletro applicazione master/detail flow e recyclerView

FILIPPO: ricercare documenti notifiche

ALESSANDRO: database