Sviluppare per Android: alcuni concetti

Programmazione Mobile

A.A. 2021/22

M.O. Spata

Foreground Activity

- Si tratta di quelle applicazioni che sono utilizzabili solo quando sono in primo piano, mentre sono effettivamente sospese quando non visibili (Es. Giochi).
- Quando si creano questo tipo di applicazioni si deve stare attenti allo stato della stessa durante tutto il suo ciclo di vita; potrebbe switchare in maniera assolutamente trasparente tra foreground e background.
- Le applicazioni non hanno controllo sul proprio ciclo di vita ed applicazioni background che non offrono servizi, sono candidate ideali alla cancellazione dal gestore di risorse di Android. Questo implica che è necessario salvare lo stato dell'applicazione quando questa diventa invisibile in modo da presentare lo steso stato non appena la stessa viene rivisualizzata

Background Activity

- Si tratta di applicazioni che richiedono minima interazione con l'utente (a parte la fase iniziale di configurazione) e vengono eseguiti in modo nascosto (es. autorisponditore di sms, gestore delle chiamate).
- Sebbene sia possibile creare applicazioni completamente nascoste è opportuno fornire un minimo di interazione come ad esempio la notifica della effettiva esecuzione del servizio, la possibilità di riconfigurare l'applicazione e un meccanismo per sospendere e/o terminare l'esecuzione.

Intermittent Activity

- Le applicazioni di questo tipo richiedono qualche interazione con l'utente ma svolgono la maggior parte dei propri compiti in background. Spesso per queste applicazioni viene impostata la configurazione iniziale, vengono eseguiti in background e mandano notifiche all'utente durante l'esecuzione (es. media player).
- Spesso sarà necessario creare applicazioni che rispondano a input degli utenti ma che sono ancora utili e svolgono dei servizi quando sono in background. Per questo tipo di applicazioni si deve porre particolare attenzione quando vi è interazione con l'utente ovvero si deve aggiornare l'interfaccia utente quando questa è visibili e mandare notifiche all'utente quando l'applicazione è in background.

Fattori da considerare in fase di progettazione di un app

- Ci sono diversi fattori da tenere in considerazione quando si sviluppa per dispositivi mobili o embedded ed in particolare quando si sviluppa per Android.
  - I dispositivi host infatti se paragonati a notebook o pc desktop hanno:
    - Un processore di limitate capacità
    - RAM limitata
    - Limitata capacità di memorizzazione
    - Schermi piccoli a bassa risoluzione
    - Alti costi associati al trasferimento dati
    - Connessioni dati poco affidabili
    - Durata della batteria limitata.

Vincoli

- Restrizioni e caratteristiche da tenere in mente quando si crea una nuova applicazione:
  - Efficienza: I costruttori di dispositivi embedded ed in particolare di dispositivi mobili, considerano la dimensione del dispositivo ed il tempo di vita della batteria miglioramenti da privilegiare rispetto alla capacità di calcolo del processore.
  - Reattività: Questo deve indurre a progettare e scrivere codice ottimizzato che che consenta all'applicazione di essere veloce, reattiva assumendo che l'upgrade hardware non necessariamente giocherà a favore delle performance del processore.

Risorse limitate

- Capacità Limitata: Anche se l'evoluzione delle tecnologie delle memorie (flash o dischi a stato solido) hanno incrementato la capacità di memorizzazione dei dispositivi mobili, in pratica, la maggior parte dei dispositivi offrono ancora oggi spazio di memorizzazione per le applicazioni; pertanto è necessario che l'applicazioni utilizzi efficientemente le risorse.
- Inoltre è necessario considerare attentamente come memorizzare i dati dell'applicazione. Per rendersi la vita più semplice si può utilizzare Il database di Android ed i content Providers per memorizzare, utilizzare e condividere grandi quantità di dati.
- Per quantità limitata di dati è possibile utilizzare un tool chiamato Preferences (meglio esaminato in seguito) o la classe Java.IO.
  Ovviamente nessuno vieta di utilizzare il filesystem per scrivere e/o leggere file.

Progettare per schermi di piccola dimensione

- In genere la piccola dimensione dello schermo va in controtendenza rispetto alla richiesta da parte dell'utente di interfaccie grafiche se più ricche di risorse: una sfida per lo sviluppatore.
- L'utilizzo di controlli (views) custom e di risorse grafiche in genere devono essere utilizzati in sostituzione di schermi pieni di controlli per garantire un massimo di usabilità.
- Inoltre lo schermo è ormai il primo strumento di input (touch-screen) pertanto nella fase di progettazione si deve considerare come un "touch" deve o può modificare l'interfaccia. Progettare la stessa in maniera tale che sia grande abbastanza per consentire il "touch".

Velocità limitate

- In fase di progettazione, specie di applicazioni che utilizzano risorse di rete, è bene assumere che la connessione sia lenta, intermittente e costosa (design for the worst case).
- Costi: L'accesso alle funzionalità più potenti ed interessanti sono a pagamento.
  - Ogni costo associato a funzionalità della applicazione costruita, ovviamente, va minimizzato e l'utente deve essere avvertito quando un azione potrebbe essere causa di traffico e conseguente costo aggiuntivo.
  - E' buona norma assumere che ogni interazione con l'ambiente esterno possa generare costi aggiuntivi. Minimizzare pertanto le interazioni e i costi seguendo queste direttive:
    - Trasferire meno dati possibili.
    - Fare Caching di dati
    - Fermare ogni possibile trasferimento di dati quando la corrispondente attività non è visibile.
    - Effettuare il refresh dei dati imponendo un rate più basso possibile

Ambiente dell’utente

- L'applicazione "terze-parti" per quanto interessante possa essere sicuramente si posizionerà nel rating dell'utente sempre dopo applicazioni quali: telefono, sms, email, fotocamera, MP3 player.
- E' importante considerare come e quando gli utenti potranno usare l'applicazione. Chiaramente non si può imporre come e quando fare utilizzare l'applicazione, ma si deve fare in modo che l'applicazione non distragga l'utente più del necessario.

Conseguentemente....

- Sarebbe opportuno che un app deve essere sospesa quando non è in foreground.
- Android permette la gestione di eventi quando le attività sono sospese o riattivate, in maniera tale da poter sospendere gli aggiornamenti delle Interfacce grafiche e le ricerche di rete quando l'applicazione non è visibile.
- Se è necessario che l'applicazione svolga delle operazioni in background, Android mette a disposizione le classi Services che consentono l'esecuzione di operazioni in background appunto senza l'overhead della interfaccia Utente

Passaggio dell’app da background a foreground

- In un ambiente multitasking capita spesso che un'applicazione passi regolarmente da foreground a background e viceversa.
- Quando questo accade è importante che l'applicativo ritorni in vita in maniera veloce e trasparente per l'utente (seamlessly).
- A tal proposito si deve anche ricordare che il gestore dei processi di Android potrebbe in maniera assolutamente non deterministica e trasparente per l'utente decidere di uccidere un processo in background per liberare risorse.
- Pertanto è necessario che lo stato dell'applicazione venga conservato e ripristinato nel momento in cui l'applicazione diventa di nuovo visibile. L'utente deve rivedere la stessa interfaccia che aveva lasciato in precedenza.

Altri vincoli di una buona app...

- E’ rispettosa della privacy utente: l'applicazione non dovrebbe mai sottrarre il focus o interrompere in qualche modo l'attività corrente dell'utente. Esistono appositi strumenti per richiamare l'attenzione dell'utente anche quando un applicazione non è in foreground (notifications e toasts).
- Presenta una interfaccia utente consistente: è bene prevedere una interfaccia easy to use.
- E' reattiva: evitare le frustrazioni dell'utente dovute a rallentamenti a seguito alla mancata risposta a seguito di un input o richiesta dell'utente. E' importante che l'applicazioni usi threads e servizi in background per mantenere l'applicazione reattiva e ancora più importante fermi attività che non rispendano in tempo ragionevole.

App veloci ed efficienti

- In particolare quando si sviluppa per Android sarebbe bene che una applicazione sia :
  - Veloce ed Efficiente: in un ambiente in cui le risorse sono limitate, essere veloce significa essere efficiente.
  - Molte delle norme circa la scrittura di codice efficiente applicata in altri contesti di programmazione valgono anche per Android.
  - Non tutto però deve essere dato per scontato. Per esempio in fase di progettazione è bene studiare ed ottimizzare l'uso dello stack e dello heap limitando la creazione di oggetti ed essere consapevoli di come ogni oggetto utilizza la memoria.

Reattività di una app

- Android considera la reattività una priorità. Attraverso l'Activity Manager ed il Window Manager, monitorizza le applicazioni e se i due strumenti segnalano la mancata risposta, mostra un messaggio di Application Unresponsive (AUR
  - La condizione di mancanza di reattività viene controllata attraverso le due condizioni seguenti:
    - Una applicazione deve rispondere ad ogni azione dell'utente in al più 5 secondi.
    - Un BroadCast Receiver deve ritornare del gestore dell'evento onReceive entro 10 secondi.

Sicurezza di un app

- Una applicazione Android ha pieno ed accesso diretto all'hardware, può essere distribuita liberamente ed è costruita su una piattaforma open source, pertanto non è difficile immaginare quanto importante sia la sicurezza.
- In generale l'utente si assume le responsabilità di quali applicazioni installa e di quali permessi concede alle stesse.
- In pratica Android per accedere a determinate funzionalità obbliga l'applicazione a richiedere esplicitamente i permessi per l'accesso alle stesse.
- Durante la fase di installazione gli utenti decidono se concedere o meno i permessi all'applicazione. Ciò non è sufficiente; è necessario anche controllare che l'applicazione non possa essere utilizzata (hijacked) per danneggiare il dispositivo.