Lezione 8 31/10/2024

The Cloud and Distributed Computing

Cosa significa il cloud? Cosa significa se dico che le mie foto sono nel cloud?

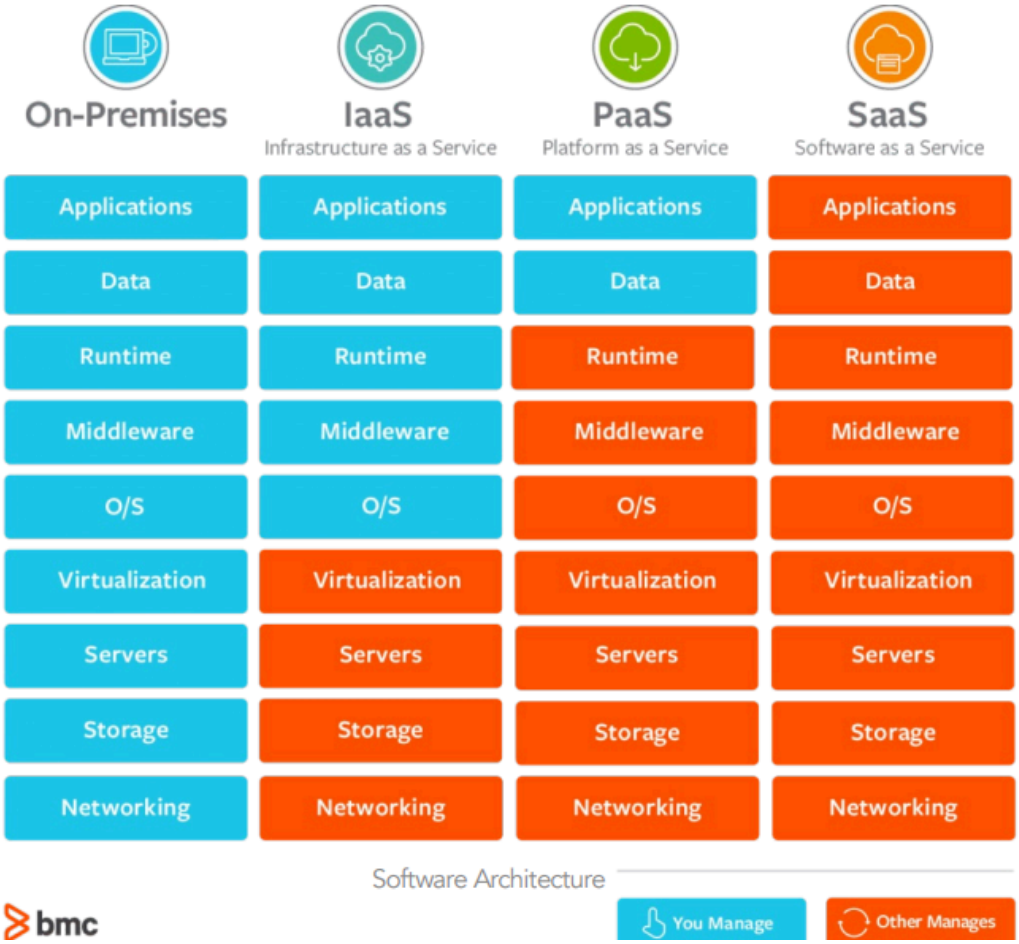
- Le mie foto sono memorizzate sui computer di qualcun altro
- Loro si occupano dell'investimento in capitale, della manutenzione e dei backup
- Le mie foto sono accessibili da me tramite Internet
- Pago solo per lo spazio che utilizzo o che mi serve
- Il servizio di archiviazione è elastico, il che significa che può espandersi o ridursi a seconda delle mie necessità
- Il mio utilizzo del cloud è self-service: creo un account e posso iniziare immediatamente a utilizzarlo per memorizzare i miei materiali

Esempi di capacità di elaborazione fornite dal cloud:

- Applicazioni
  - ad es., archiviazione di foto (o altri tipi di artefatti digitali)
- Servizi granulari esposti tramite API
  - ad es., traduzione di testi o conversione di valuta
- **Servizi di infrastruttura a basso livello**
  - ad es., virtualizzazione di processori, rete e archiviazione

Ci concentreremo su come un architetto software può utilizzare i **servizi di infrastruttura** del cloud per fornire i servizi che sta progettando e sviluppando.

I cloud possono essere pubblici (amazon, google), privati (tipo quello nell'ateneo), o ibridi.



Quando accedi a un cloud tramite un fornitore di cloud pubblico, stai effettivamente accedendo a data center sparsi in tutto il mondo

Il fornitore di cloud organizza i suoi data center in **regioni**. Una regione è una posizione fisica nel mondo dove i data center sono raggruppati. I data center all'interno di una regione sono suddivisi in **zone di disponibilità**, in modo tale che la probabilità che tutti i data center di due zone di disponibilità diverse falliscano contemporaneamente sia bassa

Scegliere la regione cloud in cui il tuo servizio verrà eseguito è una decisione progettuale importante.

Posso anche avere anche una zona di backup, in caso ci siano problemi in un'intera zona.

Supponi di voler avere una VM allocata per te nel cloud. Invii una richiesta a un gateway di gestione chiedendo una nuova istanza di VM. Questa richiesta ha molti

parametri, ma 3 sono essenziali:

- la regione cloud in cui verrà eseguita la nuova istanza
- il tipo di istanza (ad es., CPU e dimensione della memoria)
- l'ID di un'immagine VM

Il gateway di gestione è responsabile di decine di migliaia di computer fisici. Ogni computer fisico ha un hypervisor che gestisce le sue VM.

Il gateway di gestione identificherà un hypervisor che può gestire un'istanza VM aggiuntiva. Il gateway di gestione svolge altre funzioni oltre all'allocazione delle VM

- ad es., raccoglie informazioni per la fatturazione e fornisce la possibilità di monitorare e distruggere la VM

Failure in the cloud

Se la disponibilità è importante per il tuo servizio, devi riflettere attentamente sul livello di disponibilità che desideri raggiungere e come raggiungerlo.

Due concetti sono particolarmente rilevanti riguardo ai guasti nel cloud: timeout e latenza di coda lunga.

Timeouts

Il timeout è una tattica per garantire la disponibilità In un sistema distribuito, i timeout vengono utilizzati per rilevare i guasti.

Svantaggi dei timeout:

- I timeout non riescono a distinguere tra un **computer guasto** o una **connessione di rete interrotta** e una **risposta lenta** a un messaggio che supera il periodo di timeout
- Un timeout non indica dove (e quindi chi causa) si verifica il guasto o il rallentamento
- Le richieste di servizio possono innescare altre richieste
- se ogni risposta nella catena è lenta, la latenza complessiva potrebbe (falsamente) suggerire un guasto

Di solito, esiste un costo, come una penalità di latenza, per un'azione di ripristino. Ad esempio, l'avvio di una nuova VM potrebbe richiedere alcuni minuti prima che sia pronta per accettare nuove richieste

E se invece non fosse stato un guasto?

- I tempi di risposta nei sistemi cloud possono mostrare notevoli variazioni
- Saltare alla conclusione che si tratti di un guasto, quando si tratta solo di un ritardo temporaneo, può aggiungere un costo di ripristino non necessario

Come posso gestirlo?

I progettisti di sistemi distribuiti generalmente parametrizzano il meccanismo di rilevamento del timeout in modo che possa essere adattato a un sistema o a un'infrastruttura specifica

Parametri:

- intervallo di timeout
- numero di risposte mancate in un intervallo di tempo

Ad esempio, un timeout può essere impostato su 200 millisecondi e il recupero dal guasto viene attivato dopo 3 messaggi mancati in un intervallo di 1 secondo

Gli architetti dovrebbero impiegare le seguenti strategie:

Per i sistemi che operano in un singolo data center:

- i timeout e le soglie possono essere impostati in modo aggressivo, poiché i ritardi di rete sono minimi e le risposte mancate sono probabilmente dovute a crash software o guasti hardware

Per i sistemi che operano su reti più lente:

- è necessaria una maggiore attenzione nella definizione dei parametri, per evitare di attivare azioni di ripristino non necessarie

---

Long tail latency

Cause delle latenze di coda lunga: **congestione** o **guasto** da qualche parte nel percorso della richiesta di servizio

La **causa** della congestione è al di fuori del tuo controllo come sviluppatore di servizi

- Le tue tecniche di monitoraggio e le tue strategie per raggiungere le prestazioni e la disponibilità richieste devono riflettere la realtà di una distribuzione a coda lunga.

Due tecniche per gestire questi problemi sono:

- **Richieste hedge**: Effettua più richieste di quelle necessarie e poi annulla le richieste (o ignora le risposte) dopo aver ricevuto un numero sufficiente di risposte
- **Richieste alternative**: Effettua il numero corretto di richieste (ad es., 10). Quando un certo numero di esse è stato completato (ad es., 8), effettua nuovamente richieste pari a quelle mancanti (ad es., 2). Quando tutte le risposte sono state ricevute (ad es., 10), annulla le richieste ancora in corso (ad es., 2)

Vertical scaling

Se un **servizio** ospitato nel cloud riceve **più richieste di quante possa elaborare** entro la latenza richiesta, il servizio diventa **sovraccarico**.

In alcuni casi, questo è dovuto a risorse insufficienti, e puoi semplicemente eseguire il servizio in un'istanza diversa che fornisce più di quella risorsa.

Questo si chiama **scalabilità verticale** o **scalare verso l'alto**,

- corrispondente alla tattica di aumento delle prestazioni delle risorse.

Ci sono limiti a ciò che può essere ottenuto con la scalabilità verticale: ad es., potrebbe non esserci un'istanza VM abbastanza grande

In questo caso, si utilizza la **scalabilità orizzontale** o **scalare verso l'esterno**

La scalabilità orizzontale implica avere più copie dello stesso servizio e utilizzare un **bilanciatore di carico** per distribuire le richieste tra di esse: questo corrisponde alla tattica di mantenere più copie dei calcoli per le prestazioni e al modello del bilanciatore di carico.

Un bilanciatore di carico deve essere **molto efficiente** perché si trova nel percorso di ogni messaggio da un cliente a un servizio.

Load balancer per la performance

Un bilanciatore di carico risolve il seguente problema:

- Esiste un'**unica istanza di un servizio** in esecuzione su una VM o in un container, e troppe richieste stanno arrivando a questa istanza perché possa fornire una latenza accettabile

Con un bilanciatore di carico:

- più istanze del servizio
- le richieste vengono distribuite tra i servizi dal bilanciatore di carico

Un bilanciatore di carico distribuisce le richieste tra due istanze di VM (servizio)

Supponiamo che il bilanciatore di carico invii la prima richiesta all'istanza 1, la seconda all'istanza 2, la terza all'istanza 1, e così via.

- Algoritmo di round robin

Questo invia metà delle richieste a ciascuna istanza, bilanciando il carico tra di esse.

Dalla prospettiva di un client, l'indirizzo IP del servizio è l'indirizzo del bilanciatore di carico

- Il client non ha bisogno di sapere quante istanze del servizio esistono o l'indirizzo IP di nessuna di quelle istanze
- Questo rende il client **resistente** a cambiamenti di queste informazioni

Possono coesistere più client:

- Il bilanciatore di carico distribuisce i messaggi man mano che arrivano

I bilanciatori di carico possono diventare sovraccarichi

- La soluzione consiste nel bilanciare il carico tra più bilanciatori di carico, a volte chiamata **bilanciamento del carico globale**.

Load balancer per la disponibilità

I bilanciatori di carico potrebbero non avere informazioni su se un messaggio è stato elaborato o su quanto tempo ha impiegato l'elaborazione. Senza

meccanismi aggiuntivi, il bilanciatore di carico non saprebbe se un'istanza o tutte le istanze fossero fallite

Gli **health check** consentono al bilanciatore di carico di determinare se un'istanza sta funzionando correttamente, utilizzando tattiche di disponibilità di "rilevamento dei guasti" (ad es., ping, apertura di una connessione TCP, invio di un messaggio per l'elaborazione)

- Il bilanciatore di carico controllerà periodicamente la salute delle istanze a lui assegnate
- Le istanze guaste vengono contrassegnate come non sane, e non vengono inviati ulteriori messaggi a esse.

Un bilanciatore di carico con controlli di salute migliora la disponibilità **nascondendo il guasto** di un'istanza ai client. Il pool di istanze di servizio può essere dimensionato per accogliere alcuni guasti, mantenendo comunque la latenza desiderata.

Tuttavia, un'istanza di servizio potrebbe iniziare a elaborare una richiesta del client ma non restituire mai una risposta.

- I client devono essere progettati per **ri-inviare le richieste non riuscite**.

I servizi devono essere progettati in modo tale che **richieste identiche multiple** possano essere gestite correttamente.

State management

Lo **stato** si riferisce alle informazioni interne a un servizio che influenzano il calcolo di una risposta a una richiesta del client.

La gestione dello stato è importante quando un servizio può elaborare più richieste di client contemporaneamente.

Dove dovrebbe essere memorizzato lo stato?

- La cronologia viene mantenuta in ciascuna istanza; in tal caso, i servizi sono **stateful** (con stato).
- La cronologia viene mantenuta in ciascun client; in tal caso, i servizi sono **stateless** (senza stato).

- La cronologia persiste al di fuori dei servizi e dei client, in un database; in tal caso, i servizi sono **stateless**.

Time coordination

Determinare esattamente che ora sia può essere una sfida

- Avere due o più dispositivi che concordano sull'ora può essere ancora più difficile!

Dovresti assumere che **esista un certo margine di errore** tra le letture dell'orologio su due dispositivi diversi

La maggior parte dei sistemi distribuiti è progettata in modo tale che la sincronizzazione dell'ora non sia necessaria per funzionare correttamente

- È più importante conoscere l'ordine degli eventi

Per un architetto, una coordinazione temporale di successo comporta sapere se è necessario fare affidamento sugli orari effettivi, o se sia sufficiente garantire il corretto ordine sequenziale

I fornitori di servizi cloud forniscono riferimenti temporali molto precisi per i loro server orari

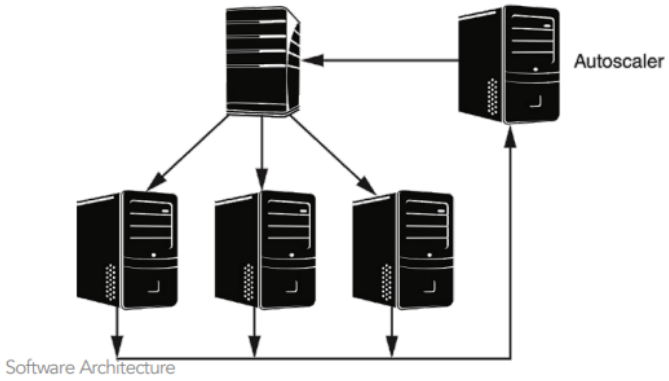
Ad esempio, Amazon e Google utilizzano orologi atomici, che hanno una deriva praticamente non misurabile

Autoscaling

I servizi di autoscaling creano automaticamente nuove istanze quando necessario e le rilasciano quando non sono più necessarie

Le attività di autoscaling sono invisibili ai client: non sanno quante istanze esistono né quale istanza sta servendo le loro richieste

L'autoscaling funziona in modo diverso per le macchine virtuali (VM) e i container



Come architetto di un servizio basato su cloud, puoi configurare una serie di regole per l'autoscaler che ne governano il comportamento, inclusi:

- L'immagine della VM da avviare quando viene creata una nuova istanza e i parametri di configurazione dell'istanza
- La soglia di utilizzo della CPU oltre la quale viene avviata una nuova istanza
- La soglia di utilizzo della CPU al di sotto della quale un'istanza esistente viene arrestata
- Le soglie di larghezza di banda di I/O di rete per la creazione e l'eliminazione delle istanze
- Il numero minimo e massimo di istanze che desideri in questo gruppo

Poiché i container vengono eseguiti su motori di runtime ospitati su VM, il ridimensionamento dei container implica due tipi di decisioni:

1. decidere che è necessario un container aggiuntivo per il carico di lavoro corrente
2. decidere se il nuovo container può essere allocato su un'istanza di motore di runtime esistente o se deve essere allocata una nuova istanza

Il software che controlla il ridimensionamento dei container è indipendente dal software che controlla il ridimensionamento delle VM

Sommario

- Il cloud è composto da data center distribuiti, gestiti tramite un gateway di gestione. È responsabile dell'allocazione, deallocazione e monitoraggio delle VM, oltre a misurare l'uso delle risorse e calcolare la fatturazione

- Dovresti assumere che, a un certo punto, le VM su cui viene eseguito il tuo servizio si guasteranno
- Dovresti anche assumere che le tue richieste per altri servizi presenteranno una distribuzione a coda lunga
- Pertanto, devi preoccuparti della disponibilità del tuo servizio
- Poiché singole istanze del tuo servizio potrebbero non essere in grado di soddisfare tutte le richieste, potresti decidere di eseguire più VM o container
- Queste istanze sono posizionate dietro un bilanciatore di carico
- La presenza di più istanze del tuo servizio e di più client ha un impatto significativo su come gestisci lo stato
- L'infrastruttura cloud può scalare automaticamente il tuo servizio creando nuove istanze quando la domanda aumenta e rimuovendo istanze quando la domanda diminuisce

Mobile Systems

Un sistema mobile può essere in movimento mentre continua a fornire parte o tutte le sue funzionalità

Questo rende la gestione di alcune sue **caratteristiche** diversa rispetto alla gestione dei sistemi fissi

Le più importanti sono: **energia, connettività di rete, sensori e attuatori, risorse e ciclo di vita**.

I sistemi mobili hanno fonti di energia limitate → devono quindi preoccuparsi di utilizzare l'energia in modo efficiente

Per molti dispositivi mobili, la fonte di energia è una **batteria** con una capacità molto limitata di fornire quell'energia

Altri dispositivi mobili, come auto e aerei, funzionano grazie all'energia prodotta da generatori, che a loro volta possono essere alimentati da motori che funzionano a combustibile, anch'esso una risorsa limitata

Architect concerns

Monitorare la fonte di energia:

- La maggior parte dei laptop o degli smartphone utilizza una batteria intelligente, ovvero un pacco batteria ricaricabile con un sistema di gestione della batteria (BMS) integrato
- Il BMS può essere interrogato per ottenere lo stato attuale della batteria

Limitare l'uso dell'energia

- L'uso dell'energia può essere ridotto terminando o degradando alcune parti del sistema (tattica di limitazione dell'uso)
- Questo può essere realizzato tramite:
  - riduzione della luminosità o della frequenza di aggiornamento del display su uno smartphone
  - riduzione del numero di core attivi del processore
  - riduzione della frequenza di clock dei core
  - riduzione della frequenza delle letture dei sensori

Tollerare una perdita di energia

- I sistemi mobili dovrebbero tollerare guasti e riavvii dell'alimentazione
- Requisiti hardware di esempio:
  - Il computer del sistema non subisce danni permanenti se l'alimentazione viene interrotta in qualsiasi momento
  - Il computer del sistema (ri)avvia il sistema operativo ogni volta che è fornita energia sufficiente
  - Il sistema operativo del sistema ha il software programmato per avviarsi non appena è pronto
- Requisiti software di esempio:
  - L'ambiente di runtime può essere terminato in qualsiasi momento senza compromettere l'integrità dei file binari, delle configurazioni e dei dati operativi, mantenendo allo stesso tempo lo stato coerente dopo un riavvio

    - Le applicazioni necessitano di una strategia per gestire i dati che arrivano mentre l'applicazione è inoperativa

Context

I sistemi mobili tendono a fornire gran parte delle loro funzionalità scambiando informazioni con altri dispositivi mentre sono in **movimento**. Pertanto, devono connettersi a questi dispositivi, ma la loro mobilità **rende queste connessioni complicate**.

L'architetto dovrebbe bilanciare molteplici preoccupazioni, tra cui:

- Numero di interfacce di comunicazione da supportare.
    - Solo le interfacce strettamente necessarie dovrebbero essere incluse per ottimizzare il consumo energetico, la generazione di calore e l'allocazione dello spazio.
- Transizione da un protocollo a un altro.
    - L'architetto deve tenere conto della possibilità che il sistema mobile possa passare da un ambiente che supporta un protocollo a un ambiente che supporta un altro protocollo.
    - Tali transizioni dovrebbero essere invisibili per l'utente.
- Scelta dinamica del protocollo appropriato.
    - Se più protocolli sono disponibili contemporaneamente, il sistema dovrebbe scegliere un protocollo in base a fattori come costo, larghezza di banda e consumo energetico.
- Modificabilità.
    - Data l'elevata quantità di protocolli e la loro rapida evoluzione, il sistema dovrebbe essere progettato per supportare cambiamenti o sostituzioni negli elementi del sistema coinvolti nella comunicazione.
- Connettività intermittente/limitata/assenza di connettività.
    - La comunicazione può essere persa mentre il dispositivo è in movimento. Il sistema dovrebbe essere progettato in modo tale da mantenere l'integrità dei dati e consentire la ripresa dei calcoli quando la connettività ritorna. Il

sistema dovrebbe essere progettato per gestire in modo efficace la connettività limitata o addirittura l'assenza di connettività.

- Sicurezza.
  - I dispositivi mobili sono particolarmente vulnerabili agli attacchi, quindi la risposta a tali attacchi dovrebbe essere parte delle preoccupazioni dell'architetto.

I sistemi mobili tendono a raccogliere più informazioni dai sensori rispetto ai sistemi fissi e spesso utilizzano attuatori per interagire con il loro ambiente.

- **Sensore**
  - Un dispositivo che rileva le caratteristiche fisiche del suo ambiente e traduce tali caratteristiche in una rappresentazione elettronica.
- **Hub dei sensori**
  - Un coprocessore che aiuta a integrare i dati provenienti da diversi sensori e a elaborarli.
- **Attuatore**
  - L'opposto di un sensore: riceve una rappresentazione digitale come input e provoca un'azione nell'ambiente.

Un architetto ha diverse preoccupazioni riguardo ai sensori:

- Come creare una rappresentazione accurata dell'ambiente basata sugli input dei sensori.
- Come il sistema dovrebbe rispondere a quella rappresentazione dell'ambiente.
- Sicurezza e privacy dei dati dei sensori e dei comandi degli attuatori.
- Funzionamento degradato. Se i sensori si guastano o diventano illeggibili, il sistema dovrebbe entrare in una modalità degradante.

Risorse

I dispositivi mobili che devono essere piccoli e leggeri hanno limiti sulle risorse che possono fornire.

- Il compromesso nella scelta delle risorse è tra il contributo della risorsa e il suo volume, peso e costo.
- I costi includono sia i costi di produzione che quelli di ingegneria.
- Le limitazioni di volume, peso e costo possono essere imposte dal dipartimento marketing e da considerazioni fisiche sull'uso del dispositivo.
- Le considerazioni fisiche per l'uso del dispositivo dipendono sia da fattori umani che da fattori di utilizzo.

Altri vincoli sulle risorse mobili includono:

- **Considerazioni di sicurezza.**
  - Le risorse fisiche che hanno conseguenze sulla sicurezza non devono guastarsi o devono avere sistemi di backup.
  - I processori, le reti o i sensori di backup aumentano il costo e il peso, oltre a consumare spazio.
- **Limiti termici.**
  - Il calore può essere generato dal sistema stesso, il che può avere un effetto dannoso sulle prestazioni del sistema, fino al punto di indurre guasti.
- **Altre preoccupazioni ambientali.**
  - Altre preoccupazioni includono l'esposizione a condizioni avverse come umidità o polvere, o la possibilità di essere lasciati cadere.

Un architetto deve prendere molte decisioni riguardo alle risorse e al loro utilizzo:

- **Assegnazione dei compiti alle unità di controllo elettronico (ECU).**
  - Questa decisione può basarsi su diversi fattori, tra cui: criticità, idoneità dell'ECU alla funzione, località della comunicazione e connettività.
- **Scarico delle funzionalità nel cloud.**
  - C'è sufficiente energia per funzioni specifiche o connettività adeguata per scaricare funzioni nel cloud?
- **Arresto delle funzioni in base alla modalità operativa.**

    - I sottosistemi che non sono in uso possono ridurre il loro footprint, consentendo ad altri sottosistemi di accedere a più risorse.
- **Strategia per la visualizzazione delle informazioni.**
    - Diverse opzioni di visualizzazione sono possibili, basate sulla dimensione e sulla risoluzione dello schermo.

Life cycle

Il testing dei sistemi mobili differisce dal testing di altri sistemi. Il rilascio di nuove versioni introduce anche alcune problematiche speciali.

- Il ciclo di vita dei sistemi mobili tende a presentare alcune idiosincrasie che un architetto deve considerare, e queste differiscono dalle scelte fatte per i sistemi tradizionali (non mobili).
- L'architetto deve occuparsi del testing, del rilascio degli aggiornamenti e della registrazione.

I dispositivi mobili presentano alcune considerazioni uniche per il testing:

- **Testare i layout dei display.**
    - Gli smartphone e i tablet sono disponibili in una vasta gamma di forme, dimensioni e rapporti di aspetto. Verificare la correttezza del layout su tutti questi dispositivi è complicato.
- **Testare i casi limite operativi.**
    - Un'applicazione dovrebbe resistere all'esaurimento della batteria e allo spegnimento del sistema, preservando lo stato.
- **L'interfaccia utente opera tipicamente in modo asincrono.** Quando l'interfaccia utente non reagisce correttamente, ricreare la sequenza di eventi che ha causato il problema è difficile.
- **Testare l'uso delle risorse.**
    - Alcuni fornitori rendono disponibili simulatori dei loro dispositivi agli architetti software. Tuttavia, testare il consumo della batteria con un simulatore è problematico.
- **Testare le transizioni di rete.**

    - Assicurarsi che il sistema faccia la scelta migliore quando sono disponibili più reti di comunicazione è anch'esso difficile.

Gli aggiornamenti al sistema possono riguardare il software, i dati o (meno frequentemente) l'hardware. Le seguenti problematiche riguardano il rilascio degli aggiornamenti:

- **Mantenere la coerenza dei dati.**
    - Per i dispositivi di consumo, gli aggiornamenti tendono ad essere automatici e unidirezionali (non c'è modo di tornare a una versione precedente). Questo suggerisce che mantenere i dati nel cloud sia una buona idea.
- **Sicurezza.**
    - L'architetto deve determinare quali stati del sistema possono supportare un aggiornamento in modo sicuro. Questo implica che il sistema deve essere consapevole degli stati rilevanti per la sicurezza in relazione agli aggiornamenti.
- **Distribuzione parziale del sistema.**
    - Rideployare un'applicazione o un sottosistema importante consuma sia banda che tempo. L'applicazione dovrebbe essere architettata in modo tale che le parti che cambiano frequentemente possano essere aggiornate facilmente.
- **Estensibilità.**
    - I sistemi veicolari mobili tendono ad avere lunghe aspettative di vita. L'adeguamento di automobili, treni, aerei, satelliti e simili diventerà probabilmente necessario a un certo punto.
    - L'adeguamento significa aggiungere nuove tecnologie a sistemi obsoleti, sia attraverso la sostituzione che l'aggiunta.

Loggin

I log sono fondamentali quando si indaga e si risolvono incidenti che si sono verificati o che potrebbero verificarsi.

Nei sistemi mobili, i log dovrebbero essere trasferiti in una posizione accessibile.

Questo è utile non solo per la gestione degli incidenti, ma anche per eseguire analisi sull'uso del sistema.

Molte applicazioni fanno qualcosa di simile quando incontrano un problema e chiedono il permesso di inviare i dettagli al fornitore.

Per i sistemi mobili, questa capacità di registrazione è particolarmente importante.

Summary

- I sistemi mobili coprono una vasta gamma di forme e applicazioni, dagli smartphone e tablet a veicoli come automobili e aerei.
- Abbiamo categorizzato le differenze tra i sistemi mobili e i sistemi fissi in base a cinque caratteristiche chiave: energia, connettività, sensori, risorse e ciclo di vita.