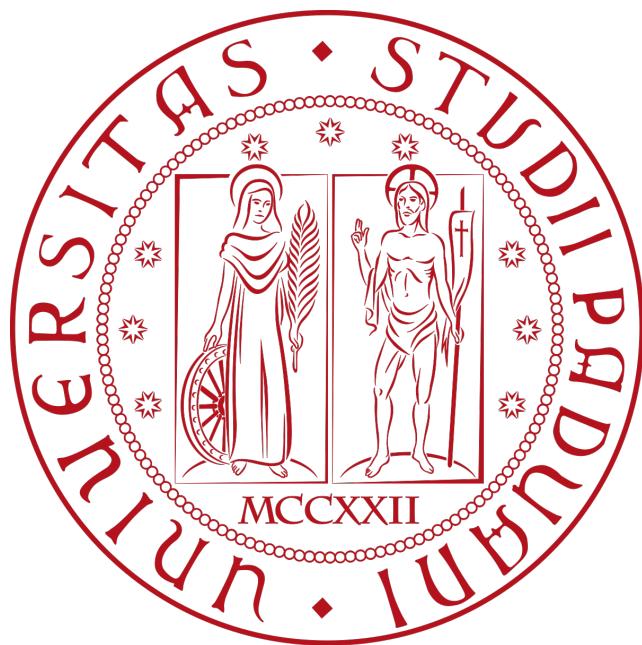


# Telemetria

Descrizione, scopi benevoli e malevoli, strumenti e illustrazione di un'applicazione che ne fa uso in Android

## Autori:

Leonardo Arduino Piccoli  
Federico Meneghetti  
Francesco Visonà



Dipartimento di Ingegneria dell'Informazione  
Università degli Studi di Padova

Programmazione di sistemi embedded

# Contents

<b>I</b>	<b>Introduzione</b>	<b>3</b>
<b>1</b>	<b>Panoramica</b>	<b>4</b>
1.1	Definizione di telemetria . . . . .	4
1.2	Mezzi e metodi di trasmissione . . . . .	5
1.3	Telemetria, monitoraggio ed osservabilità . . . . .	5
<b>2</b>	<b>Principali applicazioni</b>	<b>6</b>
2.1	Industria . . . . .	6
2.2	Difesa e aeronautica . . . . .	6
2.3	Medicina . . . . .	7
2.4	Energia . . . . .	7
2.5	Ambiente . . . . .	7
2.6	Economia . . . . .	8
<b>II</b>	<b>Telemetria in informatica</b>	<b>9</b>
<b>3</b>	<b>Sviluppo software</b>	<b>10</b>
3.1	Panoramica . . . . .	10
3.2	GDPR e consenso . . . . .	11
3.3	Sfide . . . . .	11
<b>4</b>	<b>Utilizzo</b>	<b>12</b>
4.1	Best practice . . . . .	12
4.1.1	Specifiche delle metriche . . . . .	12
4.1.2	Acquisizione dati . . . . .	12
4.1.3	Trasmissione dei dati . . . . .	12
4.1.4	Elaborazione delle informazioni . . . . .	13
4.1.5	Condivisione ed analisi dei risultati . . . . .	13
4.2	Campi di utilizzo . . . . .	13
<b>5</b>	<b>Telemetria malevola</b>	<b>15</b>
5.1	Quando si presenta . . . . .	15
5.2	Casi di cronaca . . . . .	15
5.3	Come evitare questi avvenimenti . . . . .	17
<b>III</b>	<b>Strumenti</b>	<b>19</b>
<b>6</b>	<b>Introduzione agli strumenti</b>	<b>20</b>
6.1	Glossario . . . . .	20
6.1.1	Dati raccolti . . . . .	20
6.1.2	Dati aggiuntivi . . . . .	21
6.1.3	Strumenti delle piattaforme . . . . .	21

6.1.4	Strumenti ausiliari . . . . .	22
6.2	Android Vitals . . . . .	22
<b>7</b>	<b>Strumenti omessi per brevità</b>	<b>23</b>
<b>IV</b>	<b>Strumenti a pagamento</b>	<b>24</b>
<b>8</b>	<b>DATADOG</b>	<b>25</b>
8.1	Panoramica . . . . .	25
8.2	Android . . . . .	26
<b>9</b>	<b>DYNATRACE</b>	<b>30</b>
9.1	Panoramica . . . . .	30
9.2	Android . . . . .	31
<b>10</b>	<b>APPDYNAMICS</b>	<b>33</b>
10.1	Panoramica . . . . .	33
10.2	Android . . . . .	33
<b>11</b>	<b>RAYGUN</b>	<b>36</b>
11.1	Panoramica . . . . .	36
11.2	Android . . . . .	36
<b>12</b>	<b>INSTANA</b>	<b>38</b>
12.1	Panoramica . . . . .	38
12.2	Android . . . . .	38
<b>V</b>	<b>Strumenti gratuiti</b>	<b>40</b>
<b>13</b>	<b>SENTRY</b>	<b>41</b>
13.1	Panoramica . . . . .	41
13.2	Android . . . . .	41
<b>14</b>	<b>LOGROCKET</b>	<b>44</b>
14.1	Panoramica . . . . .	44
14.2	Android . . . . .	44
<b>15</b>	<b>FIREBASE</b>	<b>47</b>
15.1	Panoramica . . . . .	47
15.2	Android . . . . .	47
<b>VI</b>	<b>Sviluppo dell'applicazione StepByStep</b>	<b>51</b>
<b>16</b>	<b>StepByStep</b>	<b>52</b>
16.1	Panoramica . . . . .	52
<b>17</b>	<b>Telemetria in StepByStep</b>	<b>58</b>
17.1	Performance Monitoring . . . . .	58
17.2	Crashlytics . . . . .	60
17.3	Google Analytics . . . . .	62
<b>VII</b>	<b>Sitografia</b>	<b>67</b>

# **Part I**

# **Introduzione**

# Chapter 1

## Panoramica

### 1.1 Definizione di telemetria

La definizione di telemetria è:

"Insieme di metodi di osservazione aventi lo scopo di fornire la misura della distanza di un oggetto dall'osservatore o anche di misurare un fenomeno che avviene a distanza dal luogo di osservazione." Enciclopedia Treccani

La parola, di origine greca (tele = lontano, metron = misura), sta ad indicare l'insieme delle tecnologie che permettono la misurazione e la trasmissione di dati sul funzionamento di un sistema ad un operatore o un progettista. Lo scopo principale è quindi quello di permettere la rilevazione di grandezze, senza essere in prossimità del sistema osservato e senza vincoli temporali.

Queste informazioni vengono raccolte con diversi scopi, ma solitamente sono aggregate ed analizzate in appositi centri di monitoraggio, consentendo la costruzione di modelli che verranno utilizzati per interventi di manutenzione o modifica del sistema stesso.



Figure 1.1: Il sistema GPS, utilizzabile come strumento per raccogliere dati di telemetria

## 1.2 Mezzi e metodi di trasmissione

Il funzionamento della telemetria, in generale, si basa su una rete di sensori posizionati in più locazioni remote che misurano dati fisici da inviare ad un centro operativo. L'informazione, nella maggior parte dei casi, viene trasmessa tramite onde elettromagnetiche (radio o ottiche) e, in misura minore, su cavi elettrici.

Esistono diversi modi con cui identificare il tipo di connessione telemetrica:

- classificazione di flusso
  - monodirezionale: il flusso di dati viaggia in un'unica direzione (usata nel rilevamento di informazioni);
  - bidirezionale: flusso di dati in entrambe le direzioni (utilizzata sia per misurare che per modificare i parametri del sistema);
- classificazione temporale
  - in tempo reale: i dati vengono inviati istantaneamente;
  - al passaggio: in un dato istante vengono inviati i dati prodotti nell'intervallo di tempo precedente;
- classificazione spaziale
  - linea commutata: collegamento temporaneo tra due terminali all'interno di una rete;
  - linea dedicata: collegamento permanente e non condiviso.

## 1.3 Telemetria, monitoraggio ed osservabilità

Mentre, come già detto, la telemetria è l'insieme di mezzi con cui si eseguono misure di un dato sistema, si può dire che l'osservabilità sia la proprietà dello stesso di essere misurato e il monitoraggio l'azione propria della misura.

Nella pratica, per monitoraggio si intende la raccolta dei dati con lo scopo di controllare se un sistema funziona come previsto: risponde dunque alla domanda "Cosa avviene all'interno del sistema?". Questa tecnica archivia poi tali dati in modo che possano venire analizzati. L'osservabilità, invece, comporta possibilità più ampie, consentendo di indagare più a fondo le cause di un eventuale problema. L'idea è quella di dedurre ciò che avviene all'interno di un sistema in base agli input e agli output osservati in determinati contesti: consente dunque di rispondere alla domanda "Perchè avviene un determinato evento?".

Si può dire dunque che il monitoraggio sia una componente essenziale dell'osservabilità, e che entrambi sfruttino gli strumenti telemetrici per raggiungere il loro scopo.

# Chapter 2

## Principali applicazioni

Gli usi pratici della telemetria sono innumerevoli e spaziano in molti campi con lo scopo di ottimizzare e salvaguardare gli oggetti sotto misura. Quelli che seguiranno sono solo alcuni esempi dei principali utilizzi, a dimostrare la grande varietà delle possibili applicazioni.

### 2.1 Industria

#### Impianti

Viene utilizzata in tutte le fasi di produzione industriale per monitorare aspetti legati alla sicurezza. Un esempio è costituito dal controllo accurato delle temperature in altoforni o nei processi petrolchimici.

#### Automobilismo

Nella Formula 1 vengono raccolti i dati delle auto in corsa per migliorarne le prestazioni: la trasmissione avviene via cavo dai box o tramite radiofrequenze quando le auto stanno percorrendo il circuito. Inoltre, i tecnici, grazie alla bidirezionalità, possono intervenire direttamente per impostare al meglio le vetture.



Figure 2.1: Box Ferrari

### 2.2 Difesa e aeronautica

#### Veicoli spaziali

I satelliti vengono utilizzati per raccogliere una grandissima mole di dati riguardanti la superficie terrestre. Questi dati sono poi impiegati per differenti di scopi, tra cui le previsioni atmosferiche, i sistemi GPS e persino per attività di spionaggio.

Le agenzie spaziali utilizzano anche sonde comandate a distanza per l'esplorazione del cosmo.

## **Sistemi missilistici**

Questa tecnologia è presente nei sistemi di difesa missilistica di vari Stati e ha subito un grande sviluppo soprattutto durante la Guerra Fredda.

## **2.3 Medicina**

### **Monitoraggio della glicemia**

I pazienti affetti da diabete vengono forniti di dispositivi CGM (Continuos Glucose Monitoring) che è in grado di rilevare i livelli di glucosio nel sangue e notificare il portatore sul proprio cellulare.

### **Servizi sanitari intelligenti**

Questi strumenti di misurazione raccolgono i dati sulla salute dei pazienti per poi inviarli al medico che potrà intervenire fornendo le cure migliori in base ai risultati ricevuti.

## **2.4 Energia**

### **Produzione**

La telemetria svolge un ruolo fondamentale per la raccolta dei dati di produzione energetiche, sia per la grandezza degli impianti (campi di pannelli fotovoltaici), sia per la loro pericolosità (impianti nucleari).

### **Consumo**

Occorre tenere traccia dei consumi energetici qualora quest'ultimi possano essere diminuiti, si pensi per esempio a tutti i dispositivi mobili dotati di batteria o ai consumi di un'abitazione (domotica).

## **2.5 Ambiente**

### **Qualità di aria e acqua**

Le quantità di polveri sottili e sostanze chimiche all'interno dell'atmosfera e in fonti idriche, vengono rilevate ed inviate da reti di sensori a degli specifici centri di controllo preposti a monitorare i livelli di inquinamento. Allo stesso modo, sensori meteorologici di questo tipo contribuiscono a rilevare dati come pressione atmosferica o velocità del vento per produrre previsioni atmosferiche più accurate.

### **Monitoraggio della fauna**

La telemetria viene usata tramite dispositivi GPS o specifiche targhette che vengono indossate da alcuni animali per tracciarne le rotte migratorie, per studiarne le popolazioni in casi di pericolo d'estinzione o per controllarne gli spostamenti vicino ai centri abitati, come gli orsi in Trentino Alto Adige.



Figure 2.2: L'orso M49 (Trentino)

## 2.6 Economia

### Logistica

Si monitorano tutte le attività connesse alle catene del freddo, dalla produzione alla vendita, attraverso sensori che inviano informazioni in tempo reale sulla temperatura o le memorizzano per poi trasmetterle nuovamente al ripristino della connettività.

### Commercio al dettaglio

Vengono utilizzati sistemi che rilevano automaticamente le quantità di prodotti disponibili in inventario tramite targhette RFID, permettendo di risparmiare viaggi inutili agli eventuali fornitori; questo stesso sistema può essere usato anche per prevenire il taccheggio inviando periodicamente dati ad una stazione di base.

## **Part II**

# **Telemetria in informatica**

# Chapter 3

## Sviluppo software

### 3.1 Panoramica

#### Utilità

Sviluppatori e amministratori IT utilizzano la telemetria per osservare da remoto le applicazioni software, i loro componenti e le strutture informatiche, con diversi scopi.

In particolare aiuta a:

- tenere sotto controllo lo stato e la salute di un programma,
- migliorare l'esperienza utente (ad esempio analizzando le analitiche di siti e social network),
- tracciare le prestazioni (controllare i tempi di avvio e i crash),
- monitorare la sicurezza del sistema,
- gestire da remoto aggiornamenti e configurazioni.

#### Stato e salute del programma

Tramite la telemetria gli amministratori riescono a raccogliere informazioni in tempo reale da qualsiasi dispositivo in locazioni remote (anche inaccessibili), andando ad agire in maniera immediata qualora ci fossero problemi.

#### Prestazioni e user experience

Il controllo dell'utilizzo, dei tempi e dei crash delle applicazioni aiuta gli sviluppatori a migliorarle, comprendendo quali funzionalità approfondire e quali ristrutturare.

#### Sicurezza

La telemetria può essere anche usata nell'analisi dei sistemi e delle reti per fornire importanti informazioni su eventuali problemi di sicurezza, evitando dunque falle nei sistemi, o andando ad arginarle quanto prima.

#### Aggiornamenti e configurazioni

Tramite la telemetria è molto più semplice distribuire una nuova versione di un programma, permettendo quindi un'integrazione continua di nuove funzionalità che soddisfano le richieste degli utilizzatori, o configurare ad hoc lo stesso per rispondere ad eventuali esigenze che cambiano nel tempo.

## 3.2 GDPR e consenso

Il GDPR è la legge dell'Unione Europea che tutela i dati dei cittadini europei. Si applica se:

- l'azienda che gestisce i dati risiede in Unione Europea;
- l'azienda processa i dati di persone presenti in Unione Europea.

Essa regola l'utilizzo e il processamento di dati sensibili, come nome, indirizzo IP e dati sanitari.

Il principale effetto di questa legge sulle applicazioni è l'obbligo di richiedere all'utente il consenso per il trattamento dei dati.

Gli utilizzatori finali, se esistono, devono quindi sempre essere informati da parte degli sviluppatori che i loro dati verranno utilizzati per monitorare l'utilizzo e le prestazioni delle applicazioni da remoto, e devono quindi concederne i permessi. Questa è in generale buona norma etica, anche dove la legge non dice nulla al riguardo, o è lacunosa.

## 3.3 Sfide

### Accesso ai dati

La telemetria è efficace solo se i dati arrivano effettivamente a destinazione per essere analizzati. Se gli utenti, informati dell'utilizzo di tali strumenti, dovessero impedirla, limiterebbero la quantità di informazioni disponibili e dunque la possibilità di accorgersi di problemi o di migliorare la loro esperienza.

E' necessario quindi presentare all'utente tale funzionalità non come un modo per sottrarre dati, ma come uno strumento di supporto per migliorare l'esperienza complessiva nell'uso dell'applicazione.

### Eccessive informazioni

Se da un lato la scarsità di informazioni è un problema, lo è anche la loro sovrabbondanza. L'elaborazione di una quantità troppo grande di dati, generati da una sempre più grande diffusione dei dispositivi informatici, potrebbe comportare un aumento della complessità computazionale degli algoritmi eseguiti dai sistemi di analisi centrali, senza contare la crescita delle basi di dati necessarie a memorizzarli.

Oltretutto, laddove ci sia un'infrastruttura in grado di supportare grandi moli di dati, è certamente una sfida capire come analizzarli, e come sfruttarli per farli diventare informazioni significative per il proprio obiettivo.

### Sistemi legacy

Dato che i programmi e i sistemi più vecchi non sempre supportano i protocolli più recenti, spesso per questioni di retrocompatibilità e per gestire insiemi eterogenei di dispositivi si fa affidamento a protocolli abbastanza datati, e che quindi potrebbero non permettere tutto ciò che è possibile con i protocolli più recenti.

Per monitorare le reti ad esempio si utilizza il protocollo SNMP (Simple Network Management Protocol) del livello OSI 7 che consente la configurazione, gestione e supervisione di apparati collegati.

# Chapter 4

## Utilizzo

### 4.1 Best practice

Esiste un metodo nell'implementazione di un sistema di telemetria, costituito da vari passi.

#### 4.1.1 Specifica delle metriche

Il primo passo fondamentale per creare un sistema di telemetria efficace è decidere le metriche che si vogliono raccogliere.

Scegliendo i dati più significativi per il proprio obiettivo, si potrà poi facilmente pianificare i cambiamenti e prioritarizzare il lavoro. Per ogni metrica è inoltre necessario impostare il ciclo di vita, ovvero ogni quanto tempo essa debba essere aggiornata.

#### 4.1.2 Acquisizione dati

Una volta definite le metriche, il passaggio alla strumentazione, ovvero all'installazione di metodi con cui raccogliere tali metriche, segue direttamente.

In base alle metriche scelte gli strumenti per l'acquisizione possono variare ma tra i più comuni si trovano:

- strumenti di logging: creano dei file di log facilmente leggibili, utili per comprendere lo stato del sistema;
- strumenti di tracciamento delle prestazioni: raccolgono informazioni sulle tendenze delle metriche come le variazioni stagionali;
- strumenti di issues detection: monitorano eventuali problemi e crash
- strumenti di security analytics: controllano gli accessi e l'utilizzo del sistema.

Una volta terminato questo passaggio, i dati della telemetria potranno cominciare ad essere acquisiti.

#### 4.1.3 Trasmissione dei dati

Una volta acquisiti i dati, questi dovranno essere inviati ad una infrastruttura centrale che li memorizzi.

Per la trasmissione dei dati, come già osservato, è possibile utilizzare diversi mezzi, ma si possono scegliere anche vari protocolli in base al tipo di dato, sempre garantendo la sicurezza delle connessioni e il rispetto delle regolamentazioni sulla privacy per impedire che i dati trasmessi cadano nelle mani di chi non ne ha accesso (malintenzionati e non). Il sistema di raccolta potrebbe anche cambiare le impostazioni di trasmissione dinamicamente in base al flusso.

Per effettuare questo passaggio è necessario che l'utente dia il proprio consenso alla trasmissione dei dati: in caso contrario, i dati acquisiti possono essere scartati.

#### 4.1.4 Elaborazione delle informazioni

Dopo aver trasportato e memorizzato i dati, occorre che essi siano processati da vari algoritmi per estrarne delle informazioni: è solo attraverso l'analisi, il raggruppamento e la correlazione di tali dati che si può comprendere ciò che avviene nel sistema.

In questo passaggio diventa fondamentale aver specificato bene le metriche.

#### 4.1.5 Condivisione ed analisi dei risultati

Infine, le informazioni elaborate vanno condivise, sempre assicurandone la protezione, a tutti quelli che ne sono interessati, che siano colleghi di lavoro che contribuiscono allo sviluppo del programma o anche un pubblico più generale, come nel caso di indagini statistiche.

Molto spesso, gli strumenti di telemetria possiedono delle dashboard che permettono di visualizzare i dati anche in tempo reale, e permettono di creare dei report in maniera molto veloce.

## 4.2 Campi di utilizzo

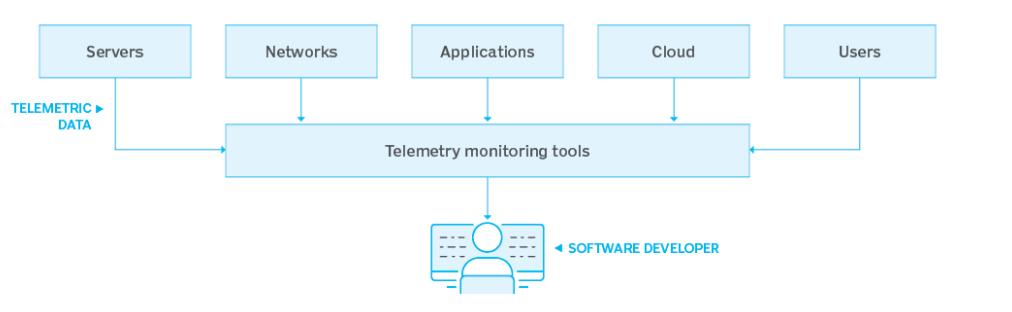


Figure 4.1: Tipi di sistemi che possono essere analizzati con la telemetria

### Server

Vengono utilizzati per fornire servizi ai client. Per questo motivo si monitorano:

- richieste degli utenti: permettono ad esempio di analizzare le feature più richieste, in modo da migliorare le prestazioni per tali richieste;
- prestazioni: con questo si intende ad esempio il carico di I/O e l'uso della memoria, che possono mostrare i primi segnali di un malfunzionamento.
- utilizzo dei processori: se non lavorano a pieno regime significa che non tutte le funzionalità sono utilizzate, che molte richieste non arrivano a destinazione o anche che le CPU dei client si stanno sforzando più del dovuto;

### Reti

Costituiscono la struttura portante dei sistemi informatici, le cui metriche principali sono:

- capacità (banda): definisce quanti dispositivi può supportare e misura il suo carico di lavoro;
- ritardi: devono essere più piccoli possibile, possono indicare problemi di traffico intenso o applicazioni troppo lente;
- porte: vengono tracciate per scongiurare problemi di violazione della sicurezza e ottimizzare il routing;
- storage: viene monitorato per vederne la capacità residua o controllare la velocità di restituzione dei dati. Potrebbero essere necessarie anche azioni di backup.

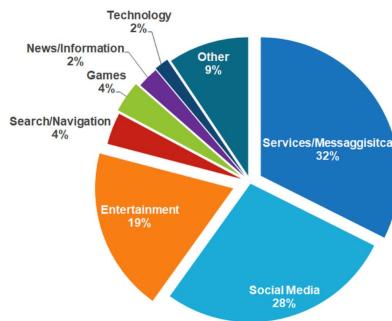


Figure 4.2: Distribuzione percentuale dell'utilizzo della rete internet in Italia

## Programmi

Sono la parte più importante dei sistemi perché si interfacciano con l'utente, si misurano quindi:

- database processing: si monitorano il numero di interrogazioni, il tempo di risposta e la quantità di dati passati. Il loro aumento potrebbe rallentare le prestazioni;
- errori: si tengono d'occhio attività inusuali o errori nei dati memorizzati che indicano breccie nella sicurezza o bug nei sistemi;
- indicatori chiave: dipendono dal dominio specifico dell'app e servono a valutare l'esperienza dei clienti;
- attività di sviluppo: si tiene traccia di sviluppo, aggiornamenti e rilasci dell'applicazione.

## Cloud

Il cloud si è sviluppato molto negli ultimi anni perché permette di delegare i compiti di gestione dei sistemi ad aziende IT specializzate. Le metriche raccolte sono simili agli altri sistemi:

- disponibilità: tempo medio di funzionamento prima che il server smetta di funzionare;
- routing: viene ottimizzato come nelle reti;
- errori: riguardano i dati ritornati dalle interrogazioni;
- latenza delle richieste: il tempo che passa dalla ricezione di una richiesta all'arrivo della risposta al destinatario;
- consumo di energia: è fondamentale che sia minimizzato soprattutto nei momenti nei quali il traffico è molto basso.

# Chapter 5

## Telemetria malevola

### 5.1 Quando si presenta

Con la recente crescita esponenziale della diffusione di dispositivi digitali sono aumentati anche i dati che circolano tra di essi: per non permettere l'accesso alle suddette informazioni da parte di individui di terze parti sono stati sviluppati vari protocolli di sicurezza. Se la telemetria di certo aiuta ad individuare tentativi di violazione da parte di hacker o possibili falle nei sistemi di sorveglianza, dall'altra parte potrebbe favorire l'invio di dati sensibili a persone con cattive intenzioni tramite delle applicazioni falsificate o malevoli. Alcuni esempi sono riportati in seguito.

### 5.2 Casi di cronaca

#### ToTok (TheNewYorkTimes 2019)

Negli Emirati Arabi Uniti, una nazione dove altri servizi come Whatsapp e Skype sono limitati, ToTok è considerata una soluzione sicura per la messaggistica. Nella realtà, invece, si tratta di uno strumento di spionaggio della popolazione che traccia conversazioni, movimenti e relazioni di chi la installa.

Sebbene sia stata scaricata milioni di volte anche in Europa, Medio Oriente, Nord America e Asia, la maggioranza degli utenti risiede comunque negli EAU. L'applicazione è stata sviluppata per conto del governo che è riuscito così ad eliminare ogni intermediario per monitorare la popolazione. Il sistema è attivo a spiare avversari stranieri, criminali o terroristi, facendo sì che siano gli utenti stessi a permettere l'accesso remoto ai loro dispositivi mobili.

Un'indagine approfondita ha scoperto che Breej Holding, la compagnia che ha sviluppato l'applicazione, è in realtà una società di facciata affiliata alla DarkMatter, un'azienda di cybersecurity e hacking con sede ad Abu Dhabi.

Sia Google che Apple, dopo ulteriori investigazioni indipendenti in merito, hanno deciso di rimuovere l'app dai loro store ufficiali.

## Weather Channel (TheNewYorkTimes 2019)

Uno dei servizi di previsioni meteorologiche più popolari degli Stati Uniti, con 45 milioni di utenti attivi, è stato condannato dal tribunale della città di Los Angeles per aver collezionato, condiviso e ricavato profitto dalle informazioni sulle locazioni dei consumatori.

L'applicazione ingannava gli utenti facendo credere che la posizione sarebbe stata utilizzata solo per poter dare informazioni più pertinenti riguardanti il meteo della zona, mentre l'azienda, facente parte di IBM, usava i dati per scopi commerciali come azioni di marketing o analisi. Altre 75 compagnie beneficiavano della raccolta dei dati nonostante non si facesse nessun riferimento rispetto alla condivisione delle informazioni con altre entità.

Le regolamentazioni riguardanti la locazione geografica precisa sono molto stringenti perché conoscendo la posizione degli utenti è possibile rilevare dettagli privati sulla loro vita, per cui l'opinione pubblica è molto sensibile a questo tema.

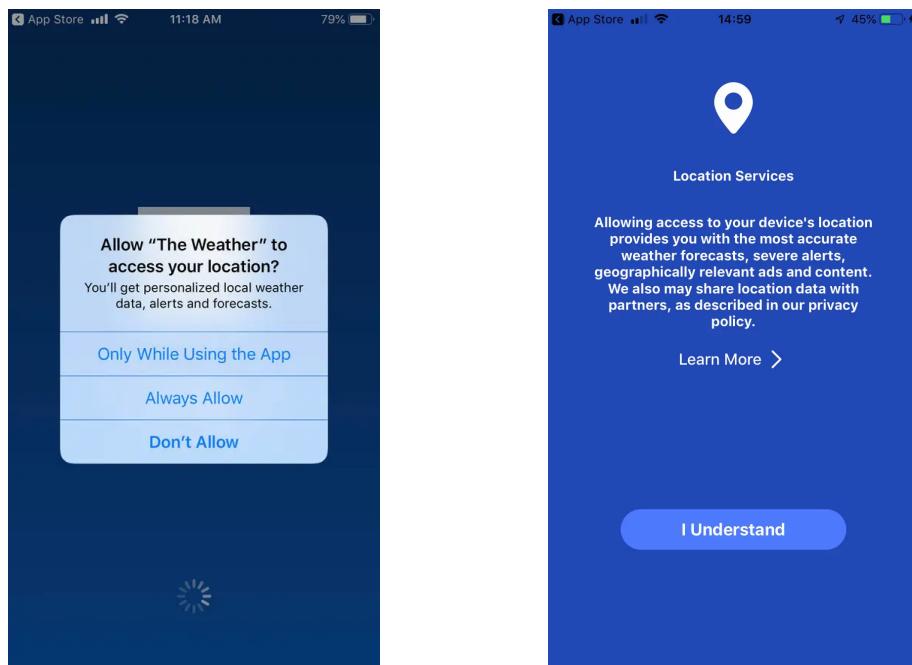


Figure 5.1: Differenze nella schermata di dialogo per la richiesta dei permessi di Weather Channel tra USA (sinistra) e EU (destra)

## Copia di Whatsapp (Kaspersky 2023)

I ricercatori di Kaspersky hanno recentemente scoperto una nuova versione dannosa di WhatsApp, che si sta diffondendo all'interno di Telegram, che ha colpito 340.000 unità in un solo mese a livello mondiale, soprattutto nei Paesi arabi come Yemen, Egitto e Turchia ma anche Stati Uniti, Germania, Russia e Regno Unito.

Come per altre versioni modificate dei servizi di messaggistica, questa aveva lo scopo di migliorarne le funzionalità, aggiungendo messaggi programmabili ed opzioni personalizzate, ma conteneva anche un modulo spyware dannoso. Il file Manifest del client, infatti, includeva componenti sospetti (un service ed un broadcast receiver) non presenti nella versione originale.

Quando il ricevitore avvia la funzionalità, il modulo spia si attiva con il cellulare acceso o in carica inviando al server informazioni sul dispositivo (IMEI, Paese, rete, ...) e sull'utente (numero di telefono, account, contatti, ...) ogni cinque minuti.



Figure 5.2: Canale Telegram che distribuisce mod di WHatsapp

### 5.3 Come evitare questi avvenimenti

Seguendo alcune importanti regole è possibile limitare la proliferazione di queste app.

#### Misure introdotte dai fornitori

Le ultime versioni di Android si stanno muovendo verso maggiori restrizioni dei permessi e un'accentuazione della privacy.

Con l'aggiornamento di PlayIntegrity API da Android 15, con il quale gli sviluppatori possono rendere più sicure le proprie app in relazione ai malware, si ha la possibilità di vedere se altri programmi stanno catturando lo schermo, creando sovrapposizioni o controllando il dispositivo durante l'esecuzione della propria applicazione, con lo scopo di proteggere le informazioni sensibili. Si ha anche la possibilità di controllare se Google Play Protect sia attivo e se il dispositivo sia libero da malware.

In Android 13 è stata introdotta una funzionalità chiamata restricted settings che limita le app non certificate dall'accedere alle notifiche e ai permessi per i service. Negli ultimi rilasci queste feature sono state espansse ulteriormente, richiedendo l'approvazione dell'utente durante l'installazione per l'abilitazione di alcune autorizzazioni; i programmatore inoltre possono ricevere report sulle recenti attività dei dispositivi, prevenendo o combattendo eventuali attacchi.



Figure 5.3: Distintivo fornito alle case di sviluppo certificate da Google

## Misure per gli utilizzatori

Nonostante i continui controlli, anche negli store ufficiali si possono trovare app mascherate con nomi, immagini e descrizioni simili alla loro controparte certificata, o addirittura con recensioni che ne prodigano l'originalità. Alcune di queste possono anche restare dormienti per molto tempo, finché non si verificano certe condizioni.

Quali sono alcuni segnali che possono mettere in guardia gli utenti sulla possibile malevolenza di un'applicazione? Ne citiamo alcuni:

- la continua comparsa di pop-up pubblicitari,
- il consumo più veloce della batteria,
- il rallentamento del dispositivo nel processare una richiesta nell'app,
- l'installazion in automatico, senza consenso dell'utente, di alcuni componenti o applicazioni ulteriori.

Alcuni metodi per limitarne la diffusione sono:

- usare i marketplace ufficiali, più controllati, ed evitare quelli di terze parti;
- controllare sempre la casa sviluppatrice e le sue recensioni;
- non concedere permessi che non siano necessari all'utilizzo primario dell'applicazione;
- disinstallare le app inutilizzate;
- mantenere un software antivirus ed eseguire scansioni periodiche.

## Android e iOS

Nonostante Google e Apple, i gestori delle due principali piattaforme di dispositivi mobili, siano in prima linea per la difesa degli utenti da fenomeni di telemetria malevola, anche loro sfruttano la propria posizione di oligopolio per inviare dati sull'utente al di fuori dalla propria volontà.

Diversi studi hanno mostrato che anche i dispositivi con una minima configurazione inviano continuamente alle due case produttrici dati sulla SIM, sul dispositivo, sui cookies, sugli indirizzi MAC vicini e sulla localizzazione, nonostante l'utente si rifiuti esplicitamente di inviarli.

# **Part III**

# **Strumenti**

# Chapter 6

## Introduzione agli strumenti

### 6.1 Glossario

Per comprendere al meglio la descrizione degli strumenti che seguirà, diamo ora la definizione di alcuni dei termini utilizzati nel seguito, suddividendoli anche per contesto.

#### 6.1.1 Dati raccolti

##### Metriche

Le metriche sono delle misure quantificabili dello stato di un sistema, come la memoria RAM utilizzata, il numero di cache hits, il tempo di esecuzione di una certa richiesta, l'utilizzo percentuale della CPU, l'utilizzo della batteria.

Il loro principale scopo è analizzare se le performance dell'applicazione intaccano in modo significativo lo stato del sistema.

##### Tracce e span

Uno span è un'unità di lavoro interna ad una o più tracce, e può contenere logs o eventi. Una traccia è un flusso completo (dovuto a una richiesta dell'utente o a una transazione) attraverso un sistema: possiede un identificatore che viene passato da una parte all'altra del sistema ed è costituita da span.

Tramite le tracce, è possibile analizzare l'insieme degli eventi, logs, e tempo di esecuzione di determinati servizi.

Molto utili sono le tracce distribuite: in questo modo è possibile monitorare un'intero sistema, comprendendo anche il back-end oltre all'applicazione, scambiandosi delle informazioni aggiuntive all'interno delle richieste HTTP per correlare due o più transazioni.

##### Logs

I logs sono dei semplici messaggi di testo che vengono inviati in un certo punto del codice. Questi servono principalmente per debug, per capire se è stato eseguito o meno una determinata parte del codice.

Se non sono aggregati, non è semplice andare a capire lo stato dell'applicazione tramite essi.

##### Eventi

Un evento è qualsiasi cosa avvenga nel sistema che valga la pena essere tracciata: sono dei logs strutturati, di solito in formato JSON. Questo significa che è più semplice analizzare e visualizzare i loro attributi.

## Breadcrumbs

Le breadcrumbs sono l'insieme dei logs che avvengono prima di un determinato errore o evento. Questi vengono spesso registrati per avere un'idea di ciò che è avvenuto prima degli errori.

## Frozen frames o ANR (Application Not Responding)

Sono i frames che richiedono più di un determinato tempo per essere renderizzati. Questo significa che l'utente non può interagire con l'applicazione.

### 6.1.2 Dati aggiuntivi

Esistono vari meccanismi molto utili per aggregare e correlare i dati raccolti.

#### Attributi

Ad ogni tipo di dato è possibile aggiungere vari attributi, che possono essere il tipo di dispositivo, l'internet service provider o la posizione.

#### Sessioni utente

Una sessione utente è definita come l'insieme di tutto ciò che accade da quando l'utente apre l'applicazione a quando la chiude. Normalmente, se l'applicazione viene messa in stop (ovvero non può più ricevere l'input dell'utente) per un determinato periodo di tempo, alla sua riapertura viene avviata una nuova sessione.

#### Cold, warm, hot start

E' riferito all'avvio dell'applicazione: cold è quando l'applicazione viene avviata dopo molto tempo o dopo aver riavviato il dispositivo, warm è quando viene avviata dopo essere stata chiusa, hot è quando non è stata chiusa (o almeno non dall'utente).

Per ogni tipo di avvio c'è un tempo massimo di inizializzazione dell'applicazione consigliato da Google.

### 6.1.3 Strumenti delle piattaforme

Sono due gli strumenti principali offerti dalle piattaforme:

#### Application Performance Monitoring

E' lo strumento di monitoraggio che traccia le metriche relative allo stato di un sistema, comprendendo errori, tempi di risposta ed altro.

E' utilizzato per capire se ci sono problemi di performance nell'applicazione.

#### Real User Monitoring (RUM)

E' un processo di monitoraggio che colleziona dati sulle sessioni di utenti reali, per capire con quali strumenti essi vadano ad interagire maggiormente (è quindi contrapposto al Synthetic Monitoring, in cui ad interagire con l'applicazione sono software di testing).

E' usato per comprendere l'utilizzo che fanno gli utenti dell'applicazione.

#### **6.1.4 Strumenti ausiliari**

##### **Dashboard**

La dashboard è lo strumento grafico con cui tutte le piattaforme permettono di gestire, aggregare ed analizzare i dati raccolti.

##### **Agent**

Un agent è un componente software da installare su un dispositivo affinchè esso possa essere monitorato. Non tutte le piattaforme ne richiedono l'installazione.

##### **Strumentazione e iniettamento di codice**

Con il termine strumentazione si intende l'inserimento di codice all'interno dell'applicazione per ottenere il monitoraggio della stessa. L'iniettamento di codice è un sinonimo di strumentazione, anche se esso è spesso usato per riferirsi a un processo di strumentazione automatico.

## **6.2 Android Vitals**

E' un'iniziativa di Google che permette di tracciare la qualità di un'applicazione andando a misurare diverse metriche, soprattutto due:

- crash rate percepito dall'utente
- ANR (Application Not Responding) rate percepito dall'utente

I valori di queste due metriche sono molto importanti anche per la visibilità dell'applicazione sul Play Store: per questo molti strumenti danno la possibilità di tenere sotto controllo queste metriche.

### **Android Performance Tuner**

E' uno strumento fornito da Android specificatamente per la misura e l'ottimizzazione di giochi per dispositivi mobili.

Esso funziona con Android Vitals: permette di aggregare metriche come il tempo di caricamento per poi andare ad inviarle a Google una volta che una versione dell'applicazione viene caricata sul Play Store.

Funziona con tutti i livelli di API superiori al 16.

## Chapter 7

# Strumenti omessi per brevità

Nominiamo di seguito alcuni strumenti per il monitoraggio del software, ma che non supportano la piattaforma Android:

- DotCom Monitor (a pagamento)
- Scout APM (con piano gratuito)
- Stackify Retrace (a pagamento)
- Application Insights (a pagamento)
- LogicMonitor (a pagamento)
- ManageEngine Applications Manager (a pagamento)
- Microsoft SCOM (a pagamento)
- Foglight (a pagamento)
- JenniferSoft (prezzi non chiari)
- CA Application Performance Management
- AppOptics APM (a pagamento)
- Sumo Logic (con piano gratuito)

Altre applicazioni che supportano Android, ma non sono state incluse nel report per brevità:

- NewRelic APM (con piano gratuito)
- Riverbed Aternity (a pagamento)
- Instabug (a pagamento)

## **Part IV**

# **Strumenti a pagamento**

# Chapter 8

# DATADOG



## 8.1 Panoramica

### Servizi offerti

Datadog è una piattaforma open source il cui scopo principale è il monitoraggio di vari tipi di software (incluse applicazioni in cloud e database) al fine di scoprire trends, errori comuni, malfunzionamenti, fallo nella sicurezza e molto altro ancora.

Inoltre garantisce una vasta gamma di strumenti, anche supportati da un motore di intelligenza artificiale, per:

- sviluppare codice: alcuni esempi sono strumenti per il pair programming e per l'analisi di codice;
- testare applicazioni, API o l'intero workflow di continuous integration senza scrivere codice, semplicemente registrando le azioni da compiere;
- rilevare attacchi e minacce per la sicurezza in base al tipo di richieste.

Nel seguito ci concentreremo sugli strumenti offerti all'utente per il monitoraggio delle applicazioni, prima in generale e poi nello specifico per la piattaforma Android.

### Monitoraggio

**Metriche raccolte** Datadog permette di registrare logs, metriche e tracce. Inoltre, è possibile visualizzare le azioni compiute dall'utente all'interno dell'applicazione ed avere un replay della sessione tramite il Real User Monitoring (RUM).

All'interno del sito è presente una dashboard che permette di visualizzare tali dati, per poi aggugarli e filtrarli in funzione di un'area geografica, tipo di device o altri parametri decisi dall'utente.

Dalla dashboard è possibile visualizzare anche i monitor impostati dall'utente basati sul superamento di una certa soglia per le metriche, sulla disponibilità del servizio e molto altro.

**Integrazione con altri servizi** E' possibile integrare le metriche ottenute tramite la piattaforma Datadog con centinaia di altri servizi di monitoraggio, autenticazione e visualizzazione tramite le Integrations.

In particolar modo, Datadog supporta OpenTelemetry, un framework open source di monitoraggio, in modo da poter esportare le metriche ottenute da tale sistema nella piattaforma, per poterle associare agli altri dati ottenuti.

**Installazione di un Agent** Per monitorare un dispositivo, è generalmente richiesta l'installazione di un agent su di esso, che raccolga già di default eventi e metriche da inviare alla piattaforma, come l'utilizzo della CPU, di sistemi di I/O, della memoria, della rete. Questo non è strettamente necessario, in quanto si può utilizzare anche la Datadog API per inviare logs, tracce, eventi e metriche alla piattaforma, ma è il metodo consigliato poichè non richiede alcuna fase di setup per iniziare a raccogliere una serie di metriche di base.

**Automatizzazione** Come ricordato in precedenza, è possibile impostare dei monitors per essere avvisati di quando certe metriche superano le soglie impostate, o in generale quando accadono determinati eventi.

Oltre a questo, Datadog permette di definire delle azioni da eseguire manualmente o in maniera automatica in seguito all'accadimento di determinati eventi: questo riduce il tempo di risposta del sistema in seguito a vari tipi di problemi, garantendo una migliore esperienza per l'utente.

## Prezzi

Datadog offre i suoi servizi unicamente a pagamento (il prezzo preciso è determinato dal tipo di servizio scelto e dalla quantità di dati raccolta), ma include la possibilità di effettuare una prova gratuita di 14 giorni.

## 8.2 Android

Per il monitoraggio sulla piattaforma Android, Datadog permette di collezionare:

- logs
- tracce
- metriche ed eventi tramite Real User Monitoring

### Logs

Tramite questa funzionalità, è possibile mandare logs di varia natura a Datadog, aggiungendo un contesto ed eventuali altri attributi custom ad ognuno.

Inoltre, è possibile inviare anche i messaggi di log presenti nel codice al di sopra di una certa priorità.

Nei logs si possono inserire:

- il tipo di connessione (3G, 4G, WiFi, ...) e il fornitore del servizio di rete;
- la traccia in cui avviene il messaggio di log, in modo da poter collegare i due tipi di dati;
- eventuali tags specificati in modo programmatico (utili per aggregare e filtrare i dati inviati);
- informazioni su: dispositivo, sistema operativo, http user agent, indirizzo IP e geolocalizzazione (mandati di default);
- eventuali altri attributi come la versione del codice e il nome.

Essi sono visibili nel Log Explorer, in cui si possono visualizzare, filtrare ed aggregare in base agli attributi presenti, oltre che esportarli in un'altra piattaforma.

A partire da questi log è poi possibile generare metriche riassuntive, processarli, aggiungere attributi o inviarli ad altri server.

Tramite il collegamento con le tracce, è possibile trovare per gli errori frequenti la probabile causa e quanti e quali utenti sono stati affetti da tale problema.

## Tracce

Questa sezione degli strumenti di Datadog permette di analizzare il tempo speso dall'applicazione nell'offrire un certo servizio: le tracce, che rappresentano l'intero path di esecuzione della richiesta, sono suddivise in span, ovvero le singole unità di lavoro come le queries di un database o la chiamata di una API, che possono quindi essere presenti anche in più tracce, in modo da identificare in maniera più agevole il probabile collo di bottiglia dell'applicazione. Una volta che un certo numero, definito dall'utente, di questi span è stato immagazzinato, questi vengono poi mandati alla piattaforma.

Le tracce permettono anche di tracciare le transazioni del database SQLite e le richieste Http.

Per gestire in maniera agevole i dati raccolti, è possibile andare a definire diversi tipi di tags agli span inviati e arricchirli con le informazioni riguardanti la RUM View corrente.

Di default, queste tracce monitorano:

- il numero di richieste
- il numero di errori
- la latenza, ovvero il tempo di servizio.

Si può configurare il volume di tracce catturate (che sono disponibili per soli 15 minuti, per non sovraccaricare il sistema) e di quelle indicizzate dai filtri (che possono essere mantenute per 15 o 30 giorni).

Le tracce possono essere collegate sia con i logs, come visto nella sezione precedente, sia con le RUM View.

## Real User Monitoring

Tramite il Real User Monitoring è possibile visualizzare ed analizzare dati relativi alle performance del dispositivo e agli input dell'utente su tutte le View dell'applicazione.

Il RUM genera eventi con metriche ed attributi associati, quali:

- il tipo di dispositivo
- informazioni sull'utente, come il nome, la nazione

Questo strumento va a registrare le varie sessioni dell'utente (ovvero ciò che va da quando l'utente lancia l'applicazione a quando la chiude, esse vengono resettate dopo 15 minuti di inattività), e all'interno di esse, per ogni View utilizzata, registra:

- le risorse, come le richieste Http o API (customizzabili)
- gli errori generati (customizzabili)
- le azioni dell'utente sulla View (customizzabili)
- le long tasks generate dalla View (ovvero quelle tasks che superano il tempo limite)

Alcuni degli attributi collezionati in maniera automatica (è sempre possibile aggiungerne di custom) sono:

- tipo, marca, modello e nome del dispositivo
- interfacce di rete, stato della connettività, fornitore del servizio di rete
- informazioni sul sistema operativo
- geolocalizzazione
- informazioni per il tracciamento dell'utente (ID, nome, email)
- tempo speso in una sessione, quali e quante View, errori, risorse, azioni e long tasks sono avvenute, indirizzo IP, Http user agent

E' possibile tracciare eventi anche quando l'applicazione è in background, e utilizzare tutti questi dati per stabilire delle metriche su cui eventualmente andare a impostare dei monitors.

In questo modo è possibile tracciare le performance dell'applicazione, gli errori (anche in base alla versione), quali utenti interagiscono maggiormente con il prodotto e su quali strumenti si concentrano, e molto altro.

Datadog permette inoltre di visualizzare i replay delle esperienze degli utenti quando interagiscono con l'applicazione, in maniera da identificare eventuali problemi ed errori.

In maniera analoga ad Android Vitals, esiste un tool chiamato Mobile Vitals che garantisce una comprensione generale dello stato di salute dell'applicazione (tempo di utilizzo della CPU, utilizzo della memoria, numero di crash, refresh rate, caricamenti lenti, blocchi della UI, numero di Application Not Responding) e delle sue performances.

## Compatibilità

Il SDK di Android è compatibile con il livello 21 di API Android (Android 5). Alcune funzionalità, come la possibilità di includere dati sulla connettività nei logs, sono limitate al livello 28 delle API Android (Android 9).

## Dati raccolti

**Sicurezza dei dati** Per ragioni di sicurezza, quando si monitora un'applicazione mobile è necessario utilizzare un client token, in modo che le API keys non siano presenti nell'APK dell'applicazione.

I dati sono prima salvati in locale sul dispositivo, per poi essere spediti quando la rete è disponibile e il livello di batteria è abbastanza alto. Questo significa che i dati raccolti in assenza di rete non vengono persi, ma viene garantita la cancellazione di tali dati quando diventano troppo vecchi.

Tali dati sono salvati in cleartext nella cache dell'applicazione, quindi in caso sia stato effettuato il root sul dispositivo potrebbero essere visibili.

**GDPR e consenso** Il SDK di Datadog permette di avere tre livelli di consenso per l'utente:

- PENDING: colleziona i dati, ma non li invia. A seconda di come viene aggiornato successivamente il permesso può inviare anche tutti i dati raccolti durante questa fase o eliminarli completamente
- GRANTED: colleziona i dati e li invia
- NOT GRANTED: non colleziona dati e non permette di mandare logs, traces o eventi RUM

Questo può anche essere modificato durante l'esecuzione del programma.

## Test

Il sistema è stato provato su un'applicazione di base, in modo da poter analizzare quali e quanti dati venissero raccolti con un setup minimo: vengono raccolti dati sull'utente, sul dispositivo, sulla posizione (riguardo a quest'ultima, solitamente la posizione non è precisa, ma indica solamente una città adiacente) e in generale su tutte le attività. L'integrazione fra RUM, tracce e logs è veramente facile, il setup iniziale è condiviso.

Grazie alla documentazione molto ben fornita, a vari esempi di applicazioni contenenti i tools di monitoraggio di Datadog e all'intuitività di gestione della dashboard, l'applicazione è fruibile in maniera molto veloce, e permette di filtrare in modo efficace i dati raccolti, le prestazioni della CPU e della memoria.

Una nota a sfavore è nell'inizializzazione: per poter fruire del servizio su Android, che si ottiene solo installando un Integration (installazione immediata in realtà), è prima necessario installare un agente su un dispositivo Desktop o un container.

# Chapter 9

## DYNATRACE



### 9.1 Panoramica

#### Servizi offerti

Dynatrace è una piattaforma che concentra i propri servizi sul monitoraggio: con essa è possibile monitorare infrastrutture e applicazioni, automatizzando determinate tasks in funzione dei dati ottenuti, ma aiuta anche nel risolvere eventuali falle di sicurezza, e il tutto è supportato da una AI, detta Davis, che consente di gestire e analizzare al meglio i dati ricevuti.

Presenta inoltre numerose estensioni per poter integrare i software più utilizzati all'interno del sistema.

#### Monitoraggio

**Dati raccolti** Dynatrace è in grado di rilevare principalmente due tipi di metriche:

- tracce: è in grado di rilevare in automatico i servizi del server, e permette di seguire le tracce attraverso tutte le tecnologie dell'infrastruttura, in maniera distribuita, in modo da trovare facilmente colli di bottiglia o errori;
- Real User Monitoring: tramite questa tecnologia, è possibile ottenere informazioni sulle sessioni degli utenti, sugli utenti stessi e sulle performance dell'applicazione.

Entrambe le metriche sono visualizzabili all'interno di una dashboard, che permette di filtrarle e raggrupparle: molto utile è il fatto che Dynatrace sia in grado di tracciare l'attività di un utente anche se questi utilizza strumenti diversi per accedere alla stessa applicazione (ad esempio vi accede tramite browser web).

**Installazione di OneAgent** OneAgent è un agente (software) che può essere installato su vari dispositivi desktop, piattaforme cloud, container e server. Tramite questo strumento è possibile rilevare in automatico tracce, metriche del RUM, logs e le performance del sistema operativo e della macchina su cui è eseguito.

L'utilizzo di questo agent è inoltre molto utile per il RUM: se il sistema su cui è eseguita l'applicazione è un dispositivo su cui è installato OneAgent (come può essere un server), Dynatrace inietta in maniera automatica del codice Javascript per il RUM nella pagina HTML fornita all'utente.

## Prezzi

Tutti i servizi di Dynatrace sono a pagamento, nonostante sia presente una prova gratuita di 14 giorni: il prezzo viene calcolato in base al servizio utilizzato e si basa sull'utilizzo orario o sulle metriche raccolte.

## 9.2 Android

Dynatrace permette in due modi il monitoraggio delle applicazioni Android: in maniera automatica tramite un plugin Gradle o mediante l'installazione e la configurazione manuale di OneAgent SDK.

**OneAgent SDK** Questo strumento dovrebbe essere utilizzato solamente se non è possibile utilizzare il Gradle plugin, o se si vogliono configurare le opzioni con cui i dati vengono raccolti (disabilitando quindi completamente l'strumentazione automatica). Con esso, è possibile istanziare manualmente OneAgent all'interno dell'applicazione, andandolo a specificare all'interno dell'Activity che si vuole monitorare.

Con questa modalità si può inoltre:

- mettere un tag agli utenti, in modo da poter ricercare le sessioni in base a questo tag
  - terminare manualmente una sessione per poi ricominciare subito dopo quella successiva
- E' anche possibile creare diverse varianti dell'applicazione, da tracciare in maniera diversa.

**Android Gradle plugin** Per utilizzare questo strumento sono necessari:

- una versione di Gradle superiore a 6.1.1
- Java 11, o superiori
- il plugin Android per Gradle, di versione superiore alla 4.0

La strumentazione, ovvero il processo di iniettamento di codice all'interno dell'applicazione per eseguirne poi il monitoraggio, avviene in automatico prima che il Java bytecode (.class) sia convertito in Dalvik bytecode (.dex), e di default vengono strumentate tutte le classi Java e Kotlin e il file AndroidManifest.xml. Non vengono quindi analizzati i file in codice nativo, né altri file .xml o .html presenti tra le risorse.

Questo strumento può essere utilizzato solo con intere applicazioni Android, e non strumenta librerie stand-alone.

## Modalità di monitoraggio

Dynatrace permette di analizzare principalmente le azioni dell'utente, le richieste web, il lifecycle delle Activity, i crash, la localizzazione dell'utente ed è in grado di rilevare quando l'utente fa una "rage tap".

Altre metriche raccolte di default sono:

- numero totale di sessioni, e quelle di nuovi utenti;
- informazioni sul dispositivo utilizzato: sistema operativo e modello;
- versioni dell'applicazione utilizzate;
- indirizzo IP ;
- internet service provider.

Per tutti i dati raccolti, si possono modificare le modalità di raccolta, i sensori utilizzati e anche disabilitare la funzionalità completamente.

E' inoltre possibile abilitare la modalità opt-in: con essa, è l'utente a scegliere specificatamente se vuole che i suoi dati siano raccolti o meno.

**Azioni dell'utente** In automatico vengono strumentati tutti i listener per monitorare le azioni dell'utente sulla UI (definita anche con Jetpack Compose, mediante data binding o tramite file XML). Il nome di queste azioni (ad esempio selezioni, cambiamenti di pagina, refresh..), dato in automatico in base al contesto e al testo dell'elemento utilizzato, può essere impostato per nascondere le informazioni sensibili dell'utente.

E' anche possibile definire manualmente delle azioni custom in cui aggiungere informazioni addizionali (si può fare il report di valori specifici, errori, eventi).

**Richieste web** La maggior parte delle richieste web sono tracciate dalla piattaforma in maniera automatica, aggiungendo un header alle richieste HTTP. Queste possono poi essere correlate alle azioni dell'utente, se avvengono contestualmente ad esse.

Sono supportati i framework:

- HttpURLConnection
- OkHttp
- Apache Http Client

Devono essere tracciate in maniera manuale le richieste non HTTP e le richieste di tutti i frameworks non supportati. Anche queste richieste possono essere associate o meno ad un'azione dell'utente.

**Lifecycle delle Activity** Di default vengono riportati i momenti in cui avvengono le chiamate dei metodi del lifecycle delle Activity: viene misurato quando viene inizializzata, quanto ci mette ad essere visualizzata e quanto ci mette ad essere visualizzata dopo che è stata già creata (ritornando dallo stato stopped o paused).

**Crash** Il sistema è in grado di tracciare tutte le eccezioni non catturate, con il momento in cui sono avvenute e gli utenti coinvolti. Questi errori dovrebbero essere inviati subito, ma talvolta è necessario riaprire l'applicazione affinchè questo avvenga.

**Rage tap** Questo evento è ciò che accade quando un utente clicca più volte lo schermo con "frustrazione": Dynatrace è in grado di rilevarlo, in modo da sapere se, quando e perchè avvengono.

**Localizzazione** La localizzazione è presa con minor precisione di quella che sarebbe possibile, per proteggere la privacy dell'utente. Inoltre, essa è rilevata solo se l'applicazione processa già di suo tale informazione.

## Compatibilità

Dynatrace è compatibile con le versioni delle API di Android superiori alla 21.

## Visualizzazione

Le dashboard con cui si possono visualizzare i dati (sotto forma di diversi tipi di grafico) permettono anche di filtrarli e cercare qualsiasi cosa tramite il Dynatrace Query Language, anche se questo non è necessario per metriche e logs.

E' inoltre possibile aggiungere codice Javascript per ricevere dati esterni tramite API e correlarli ai dati già presenti e arricchire le dashboard con elementi grafici usando un linguaggio di markdown.

# Chapter 10

# APPDYNAMICS



## 10.1 Panoramica

### Servizi offerti

AppDynamics è una piattaforma che fornisce strumenti per il monitoraggio di applicazioni e infrastrutture, per la sicurezza e la risoluzione di problemi di performance e di rete, oltre che permettere di ottenere informazioni su come gli utenti interagiscono con il proprio prodotto. Il sistema è anche integrabile con molti servizi di terze parti.

### Monitoraggio

Per iniziare il monitoraggio, è necessario seguire i seguenti passi:

- installare il Controller: tutte le applicazioni monitorate invieranno dati al Controller, sulla cui interfaccia è possibile vedere e analizzare i dati. Il Controller può essere installato su una propria macchina (on-premises) o fatto installare su una macchina di AppDynamics (SaaS);
- installare un agent o strumentare manualmente l'applicazione da monitorare;
- (opzionale) abilitare un server SMTP per ricevere notifiche su ciò che accade.

### Prezzi

AppDynamics è a pagamento: si può avere un periodo di prova gratuito di 15 giorni, dopo i quali si dovrà pagare il prezzo in base al servizio scelto, e all'utilizzo.

## 10.2 Android

Sulla piattaforma Android, AppDynamics permette di rilevare informazioni (anche in tempo reale) su:

- utenti: tipo di dispositivo, sistema operativo, versione dell'applicazione, tipo di connessione e internet service provider
- errori e crash

- performance dell'applicazione: utilizzo della CPU, GPU, memoria, rete, batteria
- correlazione client-server: tracciare le performance tra applicazione e server

Per utilizzare tale servizio, è necessario possedere una licenza RUM mobile, da acquistare sul sito.

Anche per questa piattaforma, la strumentazione dell'applicazione può avvenire in modo automatico o manualmente.

Si può disabilitare la strumentazione automatica di determinate classi o permettere la strumentazione solo per determinati tipi di build del codice.

## Modalità di monitoraggio

Di default, il tool RUM monitora tre elementi dell'applicazione: sessioni utente, richieste HTTP e crash.

Si possono aggiungere altre metriche da misurare:

- info points: quante volte un metodo viene invocato, quanto ci mette ad essere eseguito ed eventualmente quali eccezioni vengono lanciate;
- custom timers: tempi di esecuzione di qualsiasi parte del codice, anche di diversi metodi;
- custom metrics: valori interi o coppie chiave-valore riguardanti l'utente;
- breadcrumbs: logs per sapere se una certa parte del codice viene eseguita, per poi sapere il motivo per cui avviene un futuro crash.

La configurazione dell'agent può essere customizzata per andare ad utilizzare un proprio server per raccogliere i dati o abilitare il logging. Si può anche memorizzare l'indirizzo IP del mittente, anche se di default tale servizio è disabilitato.

**Sessioni utente** Si può controllare programmaticamente quando una sessione finisce o inizia, ma anche suddividerla in Session Frames, in modo da controllare più granularmente ciò che accade all'interno dell'applicazione.

Le interazioni dell'utente di default non sono rilevate, ma è possibile abilitare il monitoraggio di tale metrica.

Sono anche abilitati di default gli screenshot dell'applicazione, che possono anche essere eseguiti in qualsiasi istante: questa funzionalità si può disabilitare del tutto o bloccare solo per certi periodi di tempo (essenzialmente per privacy).

**Richieste HTTP** Per questa metrica è necessario che l'applicazione utilizzi specifiche librerie, o si può programmaticamente andare a definire quali richieste HTTP rilevare. Inoltre, in automatico vengono correlate le richieste con il processing da parte del server (se viene utilizzata una libreria supportata).

**Crash** Quando viene fatto il report delle eccezioni è possibile specificare un livello di severità dell'errore avvenuto. Inoltre, è possibile riportare crash di codice nativo.

## Compatibilità

L'agente è compatibile con versioni di Android dalla 2.3.3 in su.

## Visualizzazione

La visualizzazione avviene in una dashboard che permette di vedere:

- una overview di metriche e statistiche importanti
- una lista di sessioni utente, con le relative azioni se rilevate
- una lista delle richieste HTTP
- una lista degli errori
- i dati custom che sono stati raccolti e inviati al server

E' anche possibile definire degli alert in modo da essere avvisati nel caso in cui accadono determinati eventi, come il superamento di valori di soglia di una metrica rilevata.

# Chapter 11

## RAYGUN



### 11.1 Panoramica

#### Servizi offerti

I servizi offerti dalla piattaforma Raygun sono essenzialmente tre:

- crash reporting: cattura di errori e bug con aiuto nella diagnostica e risoluzione dei problemi
- RUM: monitoraggio delle sessioni utente per comprendere come migliorare la user experience
- APM: raccoglimento di dati sulle performance dell'applicazione per identificare possibili migliorie (solo dalla parte del server).

#### Prezzi

Nonostante la presenza di un periodo di prova di 14 giorni, Raygun è a pagamento a seconda del servizio utilizzato. Il sistema è completamente open source.

### 11.2 Android

Per la piattaforma Android Raygun offre servizi di crash reporting e RUM, ma non servizi di APM.

#### Informazioni comuni

Con una installazione molto veloce il sistema è in grado di rilevare tutte le eccezioni non catturate e gli eventi di UI per inviarli al server, andando a tracciare lo specifico utente che ha incontrato quell'errore (con un ID, ma dando la possibilità di aggiungere nome e mail). Per rispetto della privacy è anche possibile rendere l'utente anonimo, disabilitare l'invio di informazioni sensibili e preprocessare i dati inviati.

Se il dispositivo non è in grado di collegarsi, Raygun salverà i dati nel dispositivo per poi inviarli appena la rete risulta di nuovo disponibile.

E' possibile anche tracciare diverse versioni dell'applicazione, in modo da vedere quanti utenti usano ogni specifica versione o in quale versione avviene un certo errore.

## **Real User Monitoring**

E' possibile riportare manualmente eventi relativi alla UI, oltre che ignorare specifiche View. In automatico, il sistema è in grado di rilevare e inviare gli eventi di connessione di rete, oltre che dare la possibilità di rilevarli manualmente.

Tramite questo strumento è molto semplice misurare i trends di ogni Activity, misurare le performance di visualizzazione di ogni View e richiesta di rete e capire come l'utente interagisce con l'app.

## **Crash reporting**

Per quanto riguarda le eccezioni, queste possono essere arricchite aggiungendo breadcrumbs, dati e tags custom per poter poi filtrare meglio gli errori. Inoltre, è possibile mandare eccezioni manualmente, oltre che cambiare le eccezioni prima che esse vengano inviate tramite delle funzioni di callback.

Tutti gli errori hanno un gruppo, che indica il "tipo" di errore di cui si tratta: di default viene dato in base al tipo di errore e al metodo, o in base al messaggio di errore o definito dall'utente.

## **Visualizzazione**

Ogni errore ha uno stato, che può andare da risolto a ignorato, ed è possibile filtrare tali eccezioni in base ad ogni parametro presente, oltre che creare dei reports molto facilmente. Per il RUM è possibile visualizzare i trends in base all'area geografica, al dispositivo, alla versione dell'applicazione, al tipo di utente (nuovo utente, se ha mai incontrato crash e molto altro), e vedere quante e quali sessioni sono attive in ogni momento.

# Chapter 12

## INSTANA



### 12.1 Panoramica

#### Servizi offerti

Instana è una piattaforma di IBM che permette di monitorare applicazioni, infrastrutture, browser e qualsiasi tipo di servizio tramite l'installazione di un agent, aiutando a trovare errori e problemi di performance.

E' disponibile la versione di monitoraggio che invia i dati a un server IBM o la versione on-premises, in cui il server è scelto da chi acquista il servizio.

Oltre a dare la possibilità di specificare degli alerts, questo sistema fornisce gli strumenti per scrivere degli scripts da avviare in automatico all'accadimento di determinati eventi, in modo da risolvere i problemi più velocemente.

#### Prezzi

Il sistema offre servizi solo a pagamento, anche se è possibile sperimentare una prova gratuita.

### 12.2 Android

In Android il sistema è ancora in via di sviluppo, quindi per ora la piattaforma è in grado di raccogliere metriche come:

- richieste e risposte HTTP (in automatico vengono rilevate le richieste effettuate con specifiche librerie, e optionalmente anche tutti gli header utilizzati). E' anche supportato il monitoraggio delle distributed traces;
- dati sui crash (beta);
- sessioni utente con nomi di Activity e Fragment (con path locale e tempo di esecuzione dei vari metodi di callback) (beta).

Tutti gli eventi rilevati possono essere raccolti manualmente (in questo caso, il monitoraggio delle View non è in beta), ed è possibile escludere delle specifiche richieste di rete o riportare eventi custom.

Fornisce inoltre la possibilità di stabilire degli alerts per vari avvenimenti come: codici di stato delle risposte HTTP, superamento di determinate soglie (ad esempio limiti di tempo o utenti affetti da un certo evento), avvenimento di eventi custom o di crash.

## **Identificazione**

Ogni utente di base viene identificato mediante un ID fornito dalla piattaforma, ma è possibile trasmettere altri dati identificativi, anche per permettere un miglior filtraggio.

Per rispetto della privacy, è possibile anonimizzare del tutto gli utenti e identificare solo le sessioni, oppure andare a nascondere informazioni sensibili come le password.

## **Dati aggiuntivi**

Ad ogni dato raccolto è possibile associare metadati di qualsiasi genere: il sistema raccoglie in automatico l'identificativo dell'applicazione, la versione, la lingua, la versione di Android, informazioni sul dispositivo e chi l'ha prodotto, se un dispositivo è rooted, informazioni sull'ISP e l'indirizzo IP per ottenere informazioni sull'area geografica.

## **Modalità di raccolta**

Tutti i dati forniti vengono inviati mediante ad un server di IBM da un thread che funziona in background (in modo da non sovraccaricare il lavoro richiesto per la UI), ed è disponibile una coda in locale per salvare i dati se il dispositivo è offline.

## **Visualizzazione**

La dashboard di Instana è altamente personalizzabile, permettendo il filtraggio e la visualizzazione di ogni tipo di dato in maniera custom.

## **Compatibilità**

Il minimo livello di API Android è 16.

## **Part V**

# **Strumenti gratuiti**

# Chapter 13

## SENTRY



### 13.1 Panoramica

#### Servizi offerti

Sentry è una piattaforma che offre servizi di performance monitoring e tracciamento degli errori: non si concentra dunque sul monitoraggio attivo della user experience (non possiede una sezione specificatamente per il RUM, sebbene ne implementi alcune funzionalità), ma sui problemi dell'applicazione.

Per questi problemi, permette anche di fare dei replay degli errori ottenuti, seguire i problemi di performance tramite il distributed tracing, in modo da risolvere in fretta e più facilmente i bug del sistema.

#### Prezzi

Sentry presenta un piano gratuito per singoli sviluppatori che offre i servizi di monitoraggio di performance e sicurezza, con anche la possibilità di ricevere alerts via mail.

In alternativa, sono presenti vari piani a pagamento con la possibilità di integrare servizi di terze parti.

### 13.2 Android

La piattaforma permette anche in questo caso di svolgere due principali funzioni: il tracciamento di errori e il monitoring delle performance (anche tramite distributed tracing).

#### Installazione

L'installazione è molto semplice: tramite un comando da riga di comando il sistema installa automaticamente il SDK, configurando il file Gradle, il Manifest e installando il token di autenticazione. In alternativa è possibile procedere con l'installazione manuale.

#### Configurazione

Il sistema è in grado di catturare in automatico le eccezioni non catturate e monitorare la latenza e il throughput dell'applicazione.

Oltre a ciò, si possono inviare logs con valori e dati di contesto sull'utente in maniera manuale.

Le opzioni di configurazione, che possono anche essere cambiate a runtime, sono varie, e permettono di:

- modificare il luogo dove vengono inviati i dati raccolti (DSN)
- aggiungere ulteriori informazioni agli eventi per debug, oltre che specificare diverse versioni dell'applicazione con opzioni diverse
- impostare il quantitativo di dati generati e inviati
- aggiungere le stack traces alle eccezioni spedite
- inviare informazioni personali come nome, indirizzo IP, email..
- includere all'interno dei dati inviati anche dati sulle sessioni, sull'utente, tags customizzabili..
- raccogliere i body delle richieste http
- inviare dei file di attachment, come screenshot o la gerarchia delle View
- filtrare quali e quanti dati inviare tramite delle funzioni di callback
- includere ed escludere moduli
- raccogliere dati su ciò che avviene nel codice nativo (scritto tramite NDK)
- riportare gli ANR (Application Not Responding), anche usando gli stessi strumenti di Google Play

E' anche possibile spedire messaggi o breadcrumbs (ovvero dei logs strutturati che vengono inviati con il successivo evento, in modo da capire cosa è successo prima di tale evento).

## Visualizzazione e filtraggio

La piattaforma permette di filtrare i dati tramite le funzioni di callback dell'applicazione oppure nella dashboard in base ai tags o ai dati aggiuntivi inviati.

## Performance Monitoring

Sentry è in grado di misurare alcune metriche come la latenza e il throughput, tramite il monitoraggio delle tracce (anche distribuite), che riguardano anche i file di input/output, i database, le richieste HTTP e le interazioni dell'utente (come scroll, swipes..).

Oltre a ANR, slow e frozen frames, vengono monitorate:

- Activity: viene catturata una transazione per ogni Activity utilizzata, aggiungendo uno span per ogni Fragment, monitorando anche il tempo per il display di tale activity, e per il passaggio da una all'altra;
- App Start: viene misurato il tempo necessario ad iniziare l'applicazione, andando a distinguere i cold start e i warm start.

## Metriche

Questo strumento è ancora in beta: è possibile misurare delle metriche custom per ottenere dati come la media, il massimo o il minimo, o misurare tempi di esecuzione di un certo codice.

## Feedback dell'utente

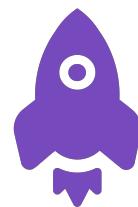
E' disponibile una API per ottenere un feedback dall'utente quando dovessero verificarsi errori di vario tipo.

## **Integrazioni con altri servizi**

Il sistema è integrabile con varie librerie di terze parti, come OkHttp o Apollo GraphQL.

# Chapter 14

## LOGROCKET



### 14.1 Panoramica

#### Servizi offerti

LogRocket è una piattaforma che offre strumenti per capire come gli utenti interagiscono con la propria applicazione (RUM), tracciare gli errori e i bugs, oltre che raccogliere diverse metriche sul funzionamento delle applicazioni.

E' anche possibile integrare il sistema con software di terze parti.

#### Prezzi

LogRocket ha un piano gratuito, oltre che diversi piani a pagamento con una prova gratuita di 14 giorni.

### 14.2 Android

LogRocket fornisce vari servizi per il monitoraggio delle sessioni utente (anche se la UI è definita con Jetpack Compose) e degli errori, oltre che il monitoraggio delle performance.

#### Real User Monitoring

Tutte le azioni dell'utente sono monitorate e vengono inviate con vari Selectors, ovvero l'elemento con cui si interagisce e il tag di tale oggetto, che permettono di filtrarli all'interno della dashboard.

Il sistema è ottimizzato per impedire di avere un overhead sui tempi dell'applicazione. E' anche possibile ricevere degli alerts in vari modi, tra cui tramite mail.

#### Inizializzazione

Durante l'inizializzazione e la strumentazione del SDK è possibile:

- andare ad identificare ogni utente mediante una stringa univoca, come un nome o una mail
- definire se l'upload dei dati può avvenire solo tramite WiFi o anche con connettività mobile

- disabilitare la raccolta degli indirizzi IP
- configurare un proxy
- includere le web views nella raccolta dei dati (questo è possibile solo inizializzando anche il Web SDK)
- terminare ed iniziare sessioni manualmente

## Dati raccolti

E' possibile monitorare eventi custom di qualsiasi tipo tramite una strumentazione manuale: è poi possibile filtrare le sessioni in base agli eventi in esse contenuti, o alle informazioni aggiuntive aggiunte a tali eventi.

Le Activity sono tracciate in automatico (sono riferite con il termine pagine), ed è possibile aggiungere delle sottopagine per ottenere un filtraggio migliore.

I logs Android non sono catturati in automatico, ma è possibile aggiungere dei logs monitorati all'interno del codice.

Di default non vengono catturate le richieste di rete, ma queste possono essere impostate manualmente (catturando solo i parametri voluti, sia per la richiesta che per la risposta).

## Errori

Si possono catturare gli errori, aggiungendo vari tag per identificarli, anche se le eccezioni non catturate dal codice vengono monitorate in maniera automatica. E' possibile fare il replay degli errori e delle sessioni.

## Privacy

Per questione di privacy è possibile saltare il monitoraggio di alcune View e prevenire la loro visualizzazione durante il Session Replay, oltre che oscurare il testo visualizzato.

## Performance Monitoring

Alcune metriche monitorate da LogRocket sono:

- utilizzo di CPU e memoria
- numero di crash
- velocità di rete
- tempo di start dell'app (distinguendo anche tra cold, warm e hot starts)
- i frozen frames
- rage clicks e interazioni frustrate dell'utente

## Visualizzazione

Le dashboard sono il metodo per visualizzare, filtrare, organizzare in grafici e riprodurre i dati ottenuti.

## Session Replay

Il sistema è in grado di riprodurre le sessioni utente registrate, permettendo di passare da una Activity all'altra, visualizzando gli eventi in una time bar e visualizzando tutti i dati della sessione (utente, versioni di SDK, del dispositivo, informazioni sulle performance).

## **Compatibilità**

Sono supportate tutte le versioni di API dalla 21 in su.

# Chapter 15

## FIREBASE



### 15.1 Panoramica

#### Servizi offerti

Firebase è una piattaforma che fornisce numerosi servizi alle applicazioni per dispositivi mobili e web, tra cui il monitoraggio delle performance, l'analisi del comportamento dell'utente, meccanismi per l'autenticazione degli utenti, il salvataggio di dati su cloud e il controllo dei crash.

Originariamente fondata da una compagnia indipendente nel 2011, fu successivamente acquistata da Google nel 2014.

Permette di sviluppare, testare, distribuire e controllare applicazioni native per Android e iOS o applicazioni web.

Questo è lo strumento utilizzato nello sviluppo dell'applicazione, come verrà descritto più avanti.

#### Prezzi

Le funzionalità di base, che includono Performance Monitoring, Google Analytics e Crashlytics (utilizzati nell'applicazione), sono gratuite, mentre sono disponibili alcuni servizi aggiuntivi su sottoscrizione di abbonamenti mensili.

### 15.2 Android

#### Compatibilità

La piattaforma supporta tutte le versioni di Android dalla 4.4 e successive, con API di livello 19 o superiore.

#### Installazione e collegamento alla console

L'installazione può avvenire manualmente, tramite l'inserimento di file di configurazione all'interno del progetto, o in maniera automatica tramite l'integrazione con Android Studio. In questo modo, l'applicazione può essere collegata alla console di Firebase, dove sarà possibile analizzare i dati raccolti.

Accedendo alla pagina [console.firebaseio.google.com](https://console.firebaseio.google.com) con un account Google si può creare un

progetto da collegare al progetto Android Studio che si sta utilizzando: qui sono anche disponibili tutte le informazioni sul collegamento, come la generazione di una chiave SHA-1 da inserire nel progetto per garantirne la sicurezza e il file google-services.json da copiare nella cartella 'app' che completerà il collegamento tra applicazione e console web.

## Perfomance monitoring

Questo strumento è utilizzato per raccogliere informazioni dettagliate sulle prestazioni dell'app. Esso si basa sulle tracce: una traccia è essenzialmente un report di ciò che avviene tra due punti nell'applicazione. I dati raccolti da una traccia sono detti metriche (numeriche) o attributi (stringhe, utili per filtrare i dati raccolti).

Dopo averlo integrato al progetto esso prenderà automaticamente diversi dati, dando inoltre la possibilità di aggiungere delle tracce custom.

**Tracce automatiche** Un primo tipo di traccia raccolta in automatico è quella di durata (ovvero in cui viene misurato il tempo tra due punti nell'applicazione). In automatico vengono tracciati:

- tempo di avvio dell'applicazione: misura i secondi che intercorrono tra il momento in cui l'utente apre l'app e il momento in cui questa è in grado di ricevere il tocco dell'utente;
- tempo in foreground: misura i secondi in cui l'app è in foreground;
- tempo in background: misura i secondi in cui l'app viene messa in background.

Un secondo tipo è quello delle tracce di screen rendering, che conta la percentuale di fotogrammi che richiedono più di un determinato tempo per il rendering (assumendo un tempo di aggiornamento di 60Hz). Le metriche, raccolte in automatico per tutte le Activity e i Fragment, sono:

- rendering lento: percentuale dei fotogrammi dello schermo che impiegano più di 16 ms per il rendering;
- frame bloccati: percentuale dei fotogrammi dello schermo che impiegano più di 700 ms per il rendering.

Questa sezione è particolarmente importante se l'app è distribuita con Google Play, come ricordato nella sezione Android Vitals.

Il terzo tipo di traccia automatica (solo se si utilizzano determinate librerie) è quella per le richieste HTTP/S, che raccoglie dati sul tempo di risposta, la quantità di dati scambiati e il rate di successo delle richieste verso un determinato endpoint. Questi dati possono essere aggregati sulla base di un pattern all'interno dell'URL, anche definito dallo sviluppatore.

Gli attributi passati in maniera predefinita sono:

- versione dell'app,
- sistema operativo (API level),
- tipo di dispositivo,
- fornitore di servizio internet,
- nazione.

**Tracce personalizzate** Oltre a quelle già citate, gli sviluppatori possono inserire all'interno del codice nuove tracce in base alle loro esigenze, raccogliendo metriche come la durata (raccolta in automatico quando si crea una traccia) o altre metriche custom.

Ad ogni nuova traccia è possibile inserire diverse metriche e fino a 5 attributi custom.

**Visualizzazione nella console** All'interno della console, nella sezione Performance Monitoring, è possibile visualizzare i dati raccolti. È possibile personalizzare la dashboard in modo da visualizzare immediatamente le metriche di maggiore interesse e la loro evoluzione nel tempo.

Per ogni traccia, è presente anche una pagina di troubleshooting che aiuta lo sviluppatore ad analizzare e comprendere i cambiamenti delle metriche raccolte.

E' inoltre possibile visualizzare i dati suddividendoli in sessioni utente, avendo informazioni anche sull'utilizzo della CPU e della memoria all'interno della sessione.

Infine, la piattaforma permette di definire degli alerts se determinate metriche dovessero superare delle soglie prestabilite.

## Crashlytics

**Utilità** Questo strumento aiuta a monitorare crash, errori che non portano all'arresto anomalo dell'applicazione e errori di ANR che si verificano nel programma, raggruppandoli in modo intelligente e stabilendone la priorità, anche in base al numero di utenti colpiti, e permettendo l'invio di alerts.

Grazie a Crash Insight si ha inoltre la possibilità di identificare i problemi più comuni, raggrupparli in issues (e, all'interno della stessa issue, suddividerli in varianti), permettendo di identificare più velocemente le criticità dell'app e attuare un debug mirato.

E' disponibile anche un plugin per il monitoraggio del codice nativo incluso mediante NDK. I dati raccolti includono anche il numero di utenti e di sessioni senza crash, in modo da determinare la stabilità dell'applicazione.

**Dati aggiuntivi** La risoluzione dei problemi viene migliorata grazie a informazioni aggiuntive raccolte da Crashlytics tra cui:

- una lista dello stack con tutti i thread in esecuzione al momento dell'arresto anomalo (e nel caso di ANR, anche qual è la causa del problema nel thread in questione);
- informazioni sul dispositivo (tipo, sistema operativo, versione dell'app, ...);
- stato del dispositivo (memoria libera, spazio disponibile sul disco, orientamento, ...).

E' inoltre possibile aggiungere:

- coppie chiave valore per definire lo stato dell'applicazione
- messaggi di log custom
- ID utente
- breadcrumbs su ciò che è avvenuto precedentemente al crash (tramite l'integrazione con Google Analytics).

## Google Analytics

È uno strumento di analisi del comportamento degli utenti che cattura qualsiasi tipo di evento all'interno dell'applicazione, e segmenta il pubblico in base a diversi attributi.

Risulta utile per capire come si compone la popolazione che usa il programma, in modo da rendere la pubblicità e l'utilizzo dei dati più significativi.

**User properties ed eventi** Le due principali informazioni prese da questo strumento sono le proprietà utente, che descrivono la base demografica degli utilizzatori, e gli eventi, che invece registrano ciò che avviene all'interno dell'app, come azioni, chiamate di sistema o errori.

All'interno della piattaforma è anche disponibile una sezione dedicata solamente al monitoraggio degli e-commerce, che permette ad esempio di stabilire su quali prodotti applicare

sconti, e una sezione per misurare il ricavo dalle campagne pubblicitarie e l'ammontare degli acquisti in app.

Tra le informazioni utente predefinite si trovano:

- informazioni personali (età, genere, interessi, ...),
- dati del dispositivo (modello, lingua, sistema operativo, ...),
- geolocalizzazione dell'origine dell'attività (nazione, città, ...).

Gli eventi spaziano in molti ambiti:

- applicazioni (primo avvio, aggiornamenti, arresti, disinstallazioni, ...),
- schermo (transizioni, scroll, refresh, ...),
- utenti (sessioni, app in primo piano, ...).
- pubblicità (click, richieste, ...),
- web (reindirizzamenti, download, ...).

**Dati personalizzati** Come per le tracce, i programmati hanno la possibilità di settare all'interno della propria applicazione degli eventi o delle proprietà utente per conto proprio.

**Visualizzazione nella console** All'interno della console è possibile visualizzare gli eventi raccolti, quante volte e da quanti utenti è stato scatenato, raggruppare l'audience in base a qualsiasi tipo di attributo, osservare l'adozione, l'engagement e la stabilità dell'ultima release, e visualizzare in tempo reale ciò che sta accadendo all'interno dell'applicazione (per valutare eventuali campagne di marketing).

**Debug** Un problema riscontrabile con i dati raccolti da Google Analytics è che non c'è certezza di quando essi vengano inviati: la piattaforma assicura solo che entro le 24 ore successive essi siano ricevuti, e afferma che i dati vengono solitamente aggregati per essere inviati una volta ogni ora, in modo da ridurre il consumo di batteria e l'impiego della rete. Per gestire il debug di questa sezione, dunque, è possibile utilizzare

- la sezione RealTime, che mostra gli eventi e le user properties di tutti gli utenti raccolte negli ultimi 30 minuti, ma che ha un tempo di aggiornamento dell'ordine dei minuti (è quindi più indicata per valutare il risultato di diverse campagne piuttosto che per il debug);
- il tool DebugView, per avere un riscontro dell'ordine di pochi secondi all'interno di un solo dispositivo.

Tramite DebugView, abilitando la modalità debug su un telefono o un browser contenente l'applicazione, è possibile vedere quasi in tempo reale gli eventi e le proprietà inviate da questo dispositivo, in modo da capire se l'strumentazione è stata effettuata correttamente.

# **Part VI**

## **Sviluppo dell'applicazione StepByStep**

# Chapter 16

## StepByStep



### 16.1 Spiegazione dell'applicazione

#### Panoramica

L'App StepByStep è stata sviluppata con lo scopo di permettere agli utenti di poter tracciare le proprie attività di camminata. L'applicazione consente di registrare su una mappa il percorso effettuato, fornendo informazioni sui metri percorsi, il tempo impiegato e la velocità media mantenuta. Per ogni giornata sarà disponibile un riassunto delle attività svolte: si potranno vedere il totale dei passi effettuati, le calorie bruciate e la somma dei metri percorsi. L'avanzamento di questi dati potrà essere confrontato in relazione a degli obiettivi giornalieri impostati.

#### Configurazione Iniziale

La prima volta che si utilizza l'app dopo l'installazione, verrà visualizzata una finestra di dialogo che chiederà all'utente di concedere all'applicazione i permessi di utilizzo degli strumenti di telemetria. Se questi non accetta, l'app potrà comunque essere utilizzata dall'utente e, in tal caso, ad ogni riavvio dell'applicazione, questa finestra verrà riproposta. Se i servizi di telemetria vengono accettati, l'utente non potrà più revocare il consenso. Nel caso in cui l'utente non vorrà più concedere l'invio dei dati di utilizzo di StepByStep agli sviluppatori dovrà disinstallare l'applicazione.

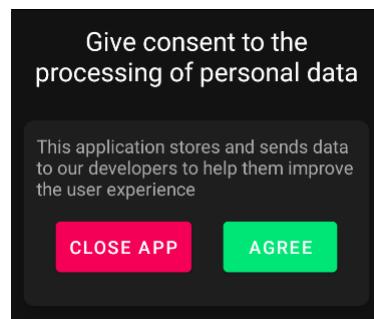


Figure 16.1: PermissionDialog

## Navigazione

L'applicazione è composta da cinque sezioni principali: Summary, Goals, Activity, Workouts e Settings. E' possibile navigare da una sezione all'altra attraverso una bottom navigation bar.

### Sezione Summary

La sezione centrale “Summary” permette all’utente di avere un riscontro diretto con i propri progressi.

**Today** Nella pagina “Today” è possibile disporre del riassunto dei workout che sono stati fatti durante la giornata. Si può vedere in modo rapido e veloce lo stato attuale di avanzamento dei passi, delle calorie e dei metri effettuati rispetto agli obiettivi impostati per quella specifica giornata. Se gli obiettivi sono particolarmente ambiziosi, l’utente potrà raggiungerli con più di un’attività.

**Week** Nella seconda pagina “Week” si può fare un confronto diretto tra i progressi fatti durante la settimana. E’ presente un grafico dove si vedono comparati il numero di passi. Se si seleziona una barra di avanzamento si possono vedere in basso i dettagli del giorno scelto.

**All** La terza pagina “All” permette all’utente di visionare i propri progressi nel corso del tempo. Si trova infatti una comoda lista nella quale sono presenti tutte le giornate in cui sono stati svolti allenamenti.

**Nota** Il calcolo delle calorie bruciate e dei passi effettuati è basato sull’altezza e sul peso impostati dall’utente, ed utilizzano una formula per il calcolo valida per le camminate.



Figure 16.2: Summary

## Sezione Goals

Attraverso questa sezione si possono aggiornare gli obiettivi per la giornata corrente (che influiranno anche sugli allenamenti futuri). Il totale dei passi, delle calorie e dei metri percorsi in una giornata sono confrontati con i valori impostati in questa sezione. L'utente può modificare i propri obiettivi quante volte desidera, lo stato di avanzamento, infatti, si aggiusterà di conseguenza automaticamente. Se viene superata la mezzanotte l'utente non avrà più modo di modificare gli obietti per la giornata trascorsa.

L'applicazione è stata dunque concepita per settare i propri obiettivi giorno per giorno senza poter interferire negli obiettivi di giorni passati.

**Nota** Se si imposta un obiettivo questo è valido per oggi e per tutti i giorni futuri fino alla prossima modifica.

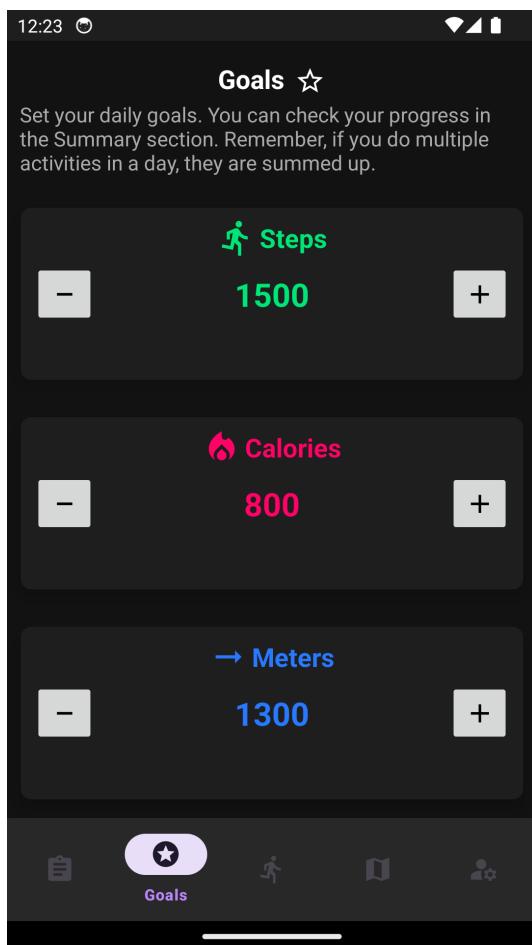


Figure 16.3: Goals

## Sezione Activity

Attraverso questa sezione è possibile registrare le proprie attività di allenamento. All'utente viene permesso di mettere in pausa la propria attività e, successivamente, di continuare da dove si era fermato.

Al termine, è possibile salvare il proprio percorso o, se non si desidera conservarlo, scartarlo.

**Nota** La prima volta che si utilizza l'app e si seleziona la sezione "Activity" viene richiesto all'utente di dare il consenso all'accesso alla propria localizzazione per permettere all'applicazione di tracciare il percorso effettuato sulla mappa. Qualora venga dato il permesso, l'applicazione avrà la facoltà di disporre di tali dati anche se l'utente sta utilizzando le altre sezioni (Settings, Goals, Summary, Workouts). Tuttavia, nel caso di StepByStep, la localizzazione dell'utente verrà utilizzata e salvata solamente quando si sta registrando un allenamento (ossia nel periodo di tempo che intercorre tra la selezione del tasto "START" e "FINISH").

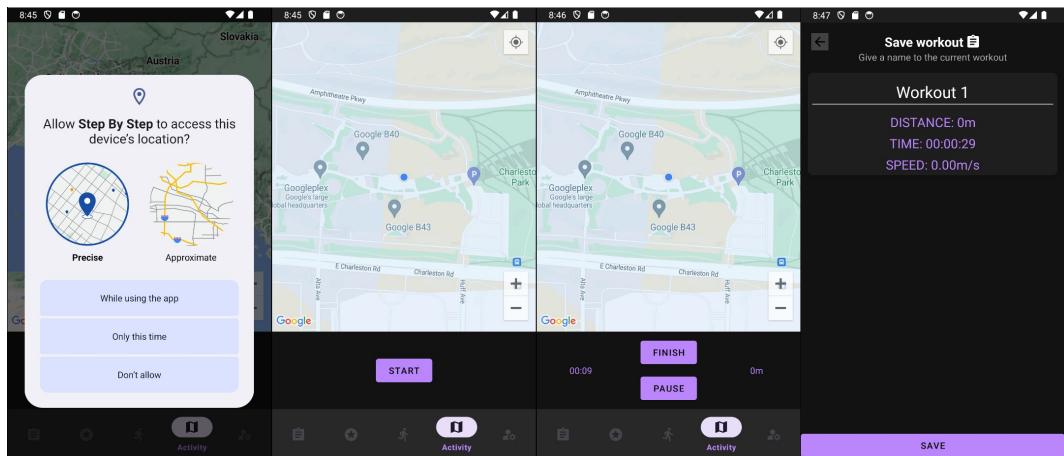


Figure 16.4: Activity

## Sezione Workout

In questa sezione sono presenti tutte le attività di allenamento che si sono registrate con la sezione Activity.

E' possibile inoltre aggiungere degli allenamenti in modo manuale selezionando il tasto “+” in basso a destra. Chiaramente, in questo caso, l'allenamento non disporrà del tracciato. Si tratta comunque di una modalità utile all'utente con cui potrà tenere aggiornati i propri dati nell'app con gli allenamenti che ha svolto in autonomia.

Se si seleziona un allenamento si potrà accedere al relativo percorso nella mappa (se disponibile). Inoltre, se si desidera, sarà possibile modificarne il nome.

**Nota** Inizialmente, gli obiettivi sono impostati di default a zero. Qualsiasi workout inserito manualmente prima della definizione dei primi obiettivi avrà automaticamente obiettivi nulli. Infatti, essendo qualcosa avvenuto nel passato, l'utente non può aver raggiunto un obiettivo che è non ancora stato definito.

In sintesi, l'applicazione è stata concepita per essere utilizzata principalmente per registrare allenamenti giornalieri, ma si è comunque liberi di inserire workout privi di obiettivi e utilizzare l'app semplicemente come diario dei propri allenamenti.

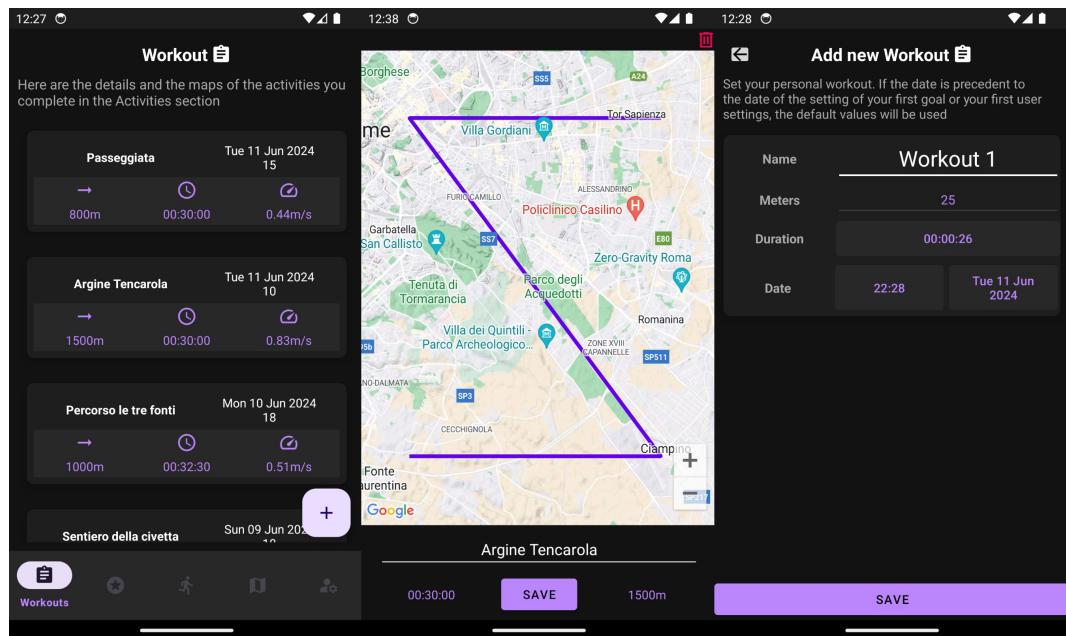


Figure 16.5: Workouts

## Sezione Settings

In questa sezione si possono aggiornare i propri dati utente. L'impostazione corretta di questi dati è utile per poter disporre in modo accurato delle calorie bruciate e dei passi effettuati.

**Nota** I dati che l'applicazione usa di default sono: 30 anni, 180 centimetri e 60 kilogrammi. Anche questi dati non possono essere modificati in maniera retroattiva: una volta definiti, verranno utilizzati per il calcolo di calorie e passi per tutti i giorni successivi, fino ad un nuovo cambiamento delle impostazioni.

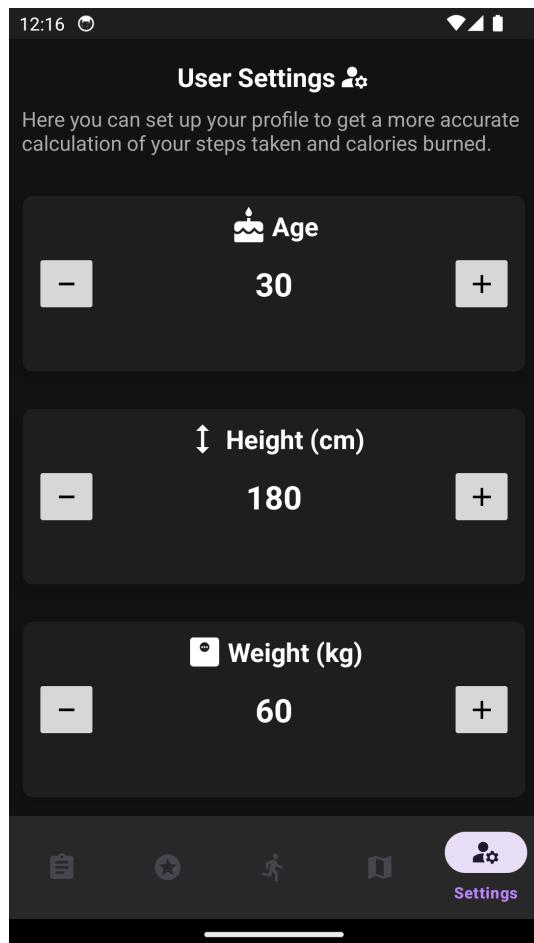


Figure 16.6: User Settings

# Chapter 17

## Telemetria in StepByStep

Per il monitoraggio dell'applicazione nel progetto presentato è stato utilizzato il framework [Firebase](#).

### 17.1 Performance Monitoring

#### Tracce automatiche

Alcune tracce vengono raccolte in automatico dal SDK Performance Monitoring, e sono visualizzabili nella console Firebase collegata al progetto. Quelle che ci interessano di più per valutare le prestazioni della nostra app sono:

- il tempo di avvio dell'app,
- il numero di frame bloccati,
- lo slow rendering.

Mentre sono molto utili per valutare l'utilizzo della nostra applicazione:

- il tempo in foreground delle activity (suddiviso per Fragment o Activity),
- il tempo in background (in questo tempo è incluso il tempo in cui il service è in foreground, ma nessuna activity è in foreground).

Non avendo effettuato richieste di rete, non sono state verificate le possibilità offerte dalla strumentazione automatica di questo tipo di tracce.

Per tutte queste tracce è possibile visualizzare i grafici degli andamenti e filtrare i dati in base a utenti, dispositivi, versioni dell'app, ...



Figure 17.1: Console con grafici delle tracce

## Tracce personalizzate

Sono state aggiunte nuove tracce con due tipi principali di scopi, in particolare:

- **today\_summaries\_trace, week\_summaries\_trace e all\_summaries\_trace** sono tracce che misurano il tempo nel quale l'utente sta interagendo con il relativo fragment, in modo tale da vedere a quale tipo di allenamenti è più interessato (giornalieri, settimanali o totali).

Queste tracce non sono necessarie, in quanto vengono già prese in automatico dalla piattaforma (è il tempo in foreground dei corrispondenti Fragment), ma l'utilizzo di queste tracce è stato inserito come test della correttezza dei tempi presi.

Questo tipo di traccia (incluse dunque quelle prese in automatico) è utile per capire dove andare ad investire in interfaccia utente, se cambiare o modificare delle funzionalità (ad esempio, se viene utilizzata molto l'Activity di aggiunta di un workout, si può pensare di posizionarla come Fragment a cui si può accedere dalla bottom bar principale), e anche che tipo di pubblico accede al nostro prodotto (se guardano spesso i workout settimanali utilizzano di frequente la nostra applicazione, se guardano solo quelli totali la usano meno, se guardano solo quelli giornalieri probabilmente il Fragment Summary andrebbe ripensato);

- **insert\_goal, insert\_workout, insert\_info, change\_workout, delete\_workout** tracciano il tempo impiegato nell'inserimento o modifica dei dati contenuti nel database. Queste tracce iniziano con l'inizio della richiesta di modifica, e finiscono quando il thread corrispondente termina la modifica del database.

Il loro scopo è dunque quello di evidenziare criticità all'interno del database, o se effettivamente le query descritte debbano essere ripensate.

Tracce personalizzate ↓	Durata	Cambiamento 7 giorni
<b>week_summary_trace</b> Numero di esempi: 7	2,28 s	+0%
<b>today_summary_trace</b> Numero di esempi: 60	33,01 s	+0%
<b>insert_workout</b> Numero di esempi: 43	345 ms	+0%
<b>insert_info</b> Numero di esempi: 16	256 ms	+0%
<b>insert_goal</b> Numero di esempi: 22	373 ms	+0%
<b>delete_workout</b> Numero di esempi: 7	33 ms	+0%
<b>change_workout</b> Numero di esempi: 3	28 ms	+0%
<b>all_summaries_trace</b> Numero di esempi: 6	23,93 s	+0%
<b>_app_in_foreground</b> ⓘ Numero di esempi: 208	1,7 min	+41%
<b>_app_in_background</b> ⓘ Numero di esempi: 140	28,04 s	+514%
<b>_app_start</b> ⓘ Numero di esempi: 116	2,01 s	+35%

Figure 17.2: Lista delle metriche personalizzate con gli andamenti

## 17.2 Crashlytics

### Crash

Grazie allo strumento Crashlytics è possibile registrare tutti i casi di crash che si verificano durante l'esecuzione dell'applicazione, raggruppandoli per tipo di errore e controllando la percentuale di utenti che ne è stata affetta.

Per effettuare alcuni dei test, è stato necessario usare dei metodi per lanciare eccezioni in maniera programmatica (naturalmente non presenti nell'applicazione finale).



Figure 17.3: Percentuali di sessioni non affette da un arresto anomalo

Problemi	Versioni	Eventi ↓	Utenti
☒ Arresto anomalo   📱 androidx.fragment.app   🗃 FragmentManager.java:1056 ✨ Problema recente FragmentManager.getFragment java.lang.IllegalStateException - Fragment no longer exists for key f#0: unique id a22d2855-1e8f-4894-bb2d-945...	1.0 – 1.0		1
☒ Arresto anomalo   📱 it.unipd.footbyfoot.fragments.su...   🗃 TodaySummaryFragment.... ✨ Problema recente TodaySummaryFragment.<init> java.lang.IllegalStateException - Fragment TodaySummaryFragment{b62fc79} (1c6c4880-effb-44a7-b979-a8ae9...	1.0 – 1.0		2
☒ Arresto anomalo   📱 it.unipd.footbyfoot.fragments.settings   🗃 SettingsFragment.kt:87 ✨ Problema recente SettingsFragment.onCreateView\$lambda\$6 java.lang.RuntimeException - Crash controllato	1.0 – 1.0		3
☒ Arresto anomalo   📱 it.unipd.footbyfoot.database.work...   🗃 WorkoutDao_Impl.java:3... ✨ Problema recente WorkoutDao_Impl\$11.call java.lang.IllegalStateException - Room cannot verify the data integrity. Looks like you've changed schema but fo...	1.0 – 1.0		3
☒ Arresto anomalo   📱 it.unipd.footbyfoot.fragments.settings   🗃 SettingsFragment.kt:67 ✨ Problema recente SettingsFragment.onCreateView java.lang.NullPointerException - findViewById(...) must not be null	1.0 – 1.0		2

Figure 17.4: Lista di alcuni crash avvenuti

## Informazioni addizionali

Come già sottolineato nella sezione sullo strumento Firebase, ad ogni crash vengono aggiunte delle informazioni addizionali, come lo stack dell'eccezione, informazioni sul dispositivo e, dato che nell'applicazione è stato integrato anche Google Analytics, breadcrumbs sugli eventi e i logs precedenti al crash.

The screenshot shows the 'Analisi dello stack' (Stack Analysis) tab in the Firebase Crashlytics interface. At the top, there are tabs for 'Riepilogo eventi', 'Chiavi', 'Log e breadcrumb', 'Dati', and 'Implementazioni'. Below the tabs are three buttons: a menu icon, 'TXT' (selected), and a download icon. The main area displays a stack trace for a 'Fatal Exception: java.lang.IllegalStateException'. The stack trace starts with the method 'androidx.fragment.app.FragmentManager.getFragment (FragmentManager.java:1056)' and continues through several layers of fragment and view restoration logic, ending with 'androidx.fragment.app.FragmentManager.executeOpsTogether (FragmentManager.java:1943)'. The entire stack trace is preceded by a bolded arrow icon.

Figure 17.5: Lista dello stack al momento del crash

The screenshot shows the 'Dati' (Data) tab in the Firebase Crashlytics interface. At the top, there are tabs for 'Riepilogo eventi', 'Analisi dello stack', 'Chiavi', 'Log e breadcrumb', 'Dati' (selected), and 'Implementazioni'. Below the tabs are navigation arrows. The main area is divided into three columns: 'Dispositivo' (Device), 'Sistema operativo' (Operating System), and 'Arresto anomalo' (Anomaly). The 'Dispositivo' column lists: Brand: Xiaomi, Modello: Mi Note 10 Lite, Orientamento: Verticale, RAM disponibile: 2.62 GB, Spazio libero su disco: 42.27 GB. The 'Sistema operativo' column lists: Versione: Android 12, Orientamento: Verticale, Con accessi di amministratore: No. The 'Arresto anomalo' column lists: Data: 3 giu 2024, 11:11:03, Versione app: 1.0 (1).

Figure 17.6: Stato del dispositivo al momento del crash

## 17.3 Google Analytics

Ricordiamo che Google Analytics raccoglie due tipi di dati: proprietà utente, che sono stringhe associate ad un utente e sono utili per suddividere il pubblico in partizioni, e gli eventi, che indicano ciò che effettivamente avviene all'interno dell'applicazione.

### Dati automatici

Sono raccolti in maniera automatica eventi di user engagement, di vista dei vari Fragment e Activity, di inizio sessione, installazione e disinstallazione dell'applicazione, e così via. Le proprietà utente automatiche sono ad esempio la localizzazione, la lingua, il sistema operativo, il modello del dispositivo, e così via.

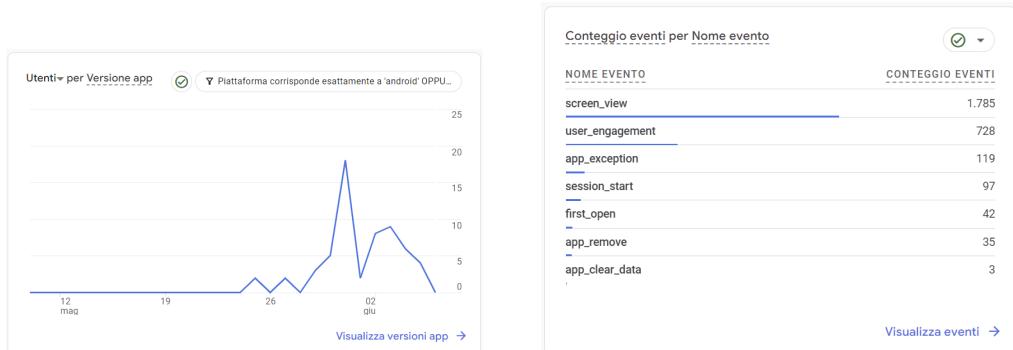


Figure 17.7: (destra) andamento degli accessi, (sinistra) eventi di base

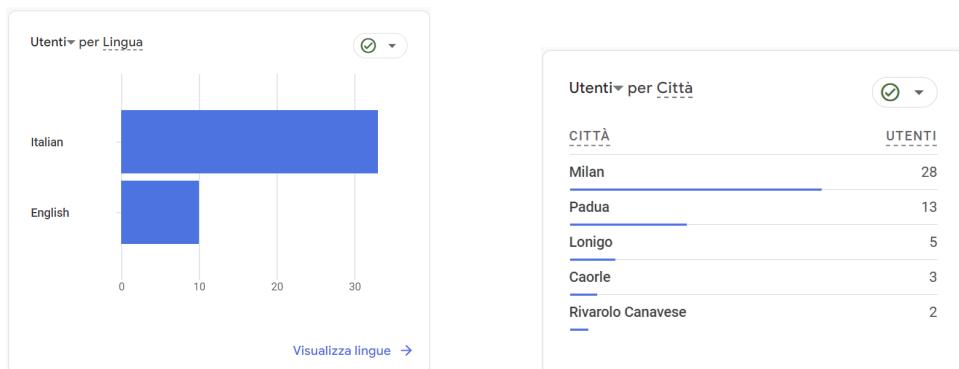


Figure 17.8: (destra) lingua impostata, (sinistra) posizione approssimata

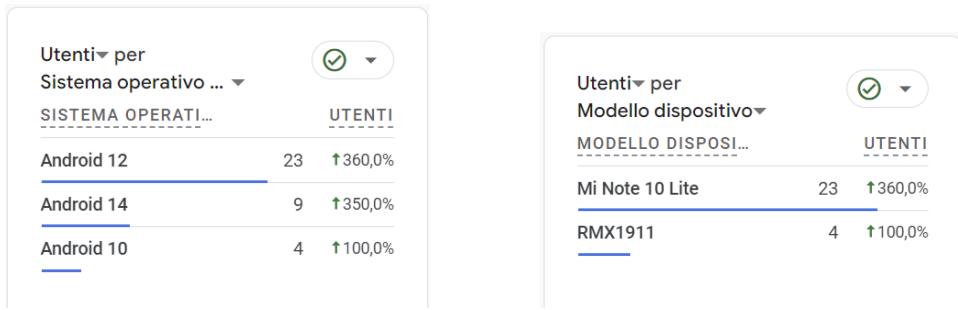


Figure 17.9: (destra) sistema operativo, (sinistra) modello del dispositivo

## Proprietà utente personalizzate

I dati raccolti in maniera automatica sono utili, ma per aumentare la comprensione delle abitudini dei consumatori sono state aggiunte le seguenti proprietà utente:

- DeclaredAge: età dell'utente (se si collegasse un account Google sarebbe raccolta in automatico, e utilizzerebbe il nome riservato Age);
- Height: altezza dell'utente (utile nel calcolare i passi);
- Weight: peso dell'utente (utile nel calcolare le calorie consumate);

Queste proprietà sono utili per comprendere quale tipologia di pubblico adotti la nostra applicazione, se è in forma o meno, e in quel caso andare ad aggiungere funzionalità mirate per questo tipo di utenti. Queste informazioni possono rientrare nelle informazioni sensibili, quindi si tratta di un caso al limite della telemetria malevole, anche se questi dati sono comunque utilizzati all'interno della nostra applicazione.

Altre proprietà utente raccolte sono:

- Workouts counter: numero di allenamenti salvati (sempre per considerare l'utilizzo che l'utente fa della nostra applicazione);
- Average speed: velocità media degli allenamenti salvati (utile nel controllare se l'applicazione è utilizzata per corse o camminate, e quindi magari in futuro permettere di differenziare tra i diversi tipi di allenamenti, o aggiornare la formula di calcolo di calorie e passi);
- Steps goal: obiettivo giornaliero dei passi;
- Distance goal: obiettivo giornaliero della distanza;
- Calories goal: obiettivo giornaliero delle calorie;

Conoscendo gli obiettivi giornalieri degli utenti, oltre ad avere una panoramica più precisa di chi siano e per cosa utilizzino l'app (solo per tenersi in forma o per migliorarsi magari in vista di una gara), saremmo in grado di impostare degli obiettivi di default migliori e magari andare a suggerire agli utenti se e quando cambiare questi obiettivi.

Definizioni personalizzate				
Dimensioni personalizzate	Metriche personalizzate	Metriche calcolate	Cerca	Crea dimensione personalizzata
Nome dimensione ↑	Descrizione	Ambito	Proprietà utente/Parametro	Ultima modifica
Average speed	Average speed of the user	Utente	AverageSpeed	10 giu 2024
CaloriesGoal	Goal set for the calories	Utente	CaloriesGoal	10 giu 2024
Declared Age	Age declared in the application	Utente	DeclaredAge	10 giu 2024
DistanceGoal	Goal set for the distance	Utente	DistanceGoal	10 giu 2024
Height		Utente	Height	10 giu 2024
StepsGoal	Goal set for the steps	Utente	StepsGoal	10 giu 2024
Weight		Utente	Weight	10 giu 2024
WorkoutsCounter	Number of workouts made	Utente	WorkoutsCounter	10 giu 2024

Figure 17.10: Alcune proprietà dell'utente per sessione

## Eventi personalizzati

Oltre agli eventi di default, sono stati aggiunti degli eventi con diversi scopi:

- **added\_workout** e **workout\_saved** sono due eventi che corrispondono rispettivamente all'aggiunta di un workout tramite l'Activity AddWorkout (accessibile dal simbolo + in basso a destra del Fragment Workouts) e al salvataggio di un workout tramite il Fragment Activity, dopo aver tracciato il corrispondente workout con l'applicazione stessa. Per entrambi gli eventi viene incluso il numero di workout salvati in entrambi i modi da quell'utente.

La provenienza dei workout della nostra applicazione è utile per capire se l'utente la utilizza come un diario dove salvare i dati dei suoi allenamenti, magari registrati mediante un'altra applicazione, o se, una volta salvati i dati già presenti nel suo storico, utilizza la nostra applicazione per registrare e salvare i workouts.

Una metrica aggiuntiva dell'evento added\_workout è il numero di giorni che passano da quando il workout è stato effettuato a quando viene salvato: se aggiunge in blocco molti workout vecchi probabilmente può essere utile aggiungere un modo per inserire un numero elevato di workout vecchi assieme, o per importarli da altre applicazioni.

- **workout\_not\_saved** è un evento che registra tutti gli allenamenti che sono stati registrati con la nostra applicazione ma che per qualche motivo non sono stati salvati. Per questi workout passiamo anche il tempo e la distanza percorsi, in modo da capire se l'allenamento è stato iniziato per sbaglio (e sarebbe quindi opportuno aggiungere un ulteriore fase di controllo allo start del workout) o se è stato fatto back senza salvare in maniera accidentale (e quindi si dovrebbe aggiungere un controllo prima di non salvare il workout)
- **workout\_deleted** è l'evento che si scatena quando l'utente elimina un workout già salvato tramite l'apposita funzione nella Activity di info del workout. In essa vengono passati il tempo e la distanza dell'allenamento, oltre alla prima coordinata salvata, se presente. Questo perché un motivo potrebbe essere l'assenza di punti all'interno della mappa, o magari la visualizzazione sbagliata di tale allenamento (il livello di zoom predefinito, per quanto regolabile, non è stato testato su ogni possibile utilizzo dell'utente)
- **first\_point** è l'evento che si scatena ogni volta l'utente finisce un workout dopo averlo registrato con la nostra applicazione: esso registra la prima posizione dell'allenamento, e il suo utilizzo è quello di capire in maniera più precisa rispetto a quanto già fatto da

Firebase la localizzazione degli utenti, magari in ottica di un inserimento di pubblicità mirate

- **notification\_while\_background** (ex `click_notification`) è l'evento che viene inviato ogni qualvolta l'utente clicca sulla notifica generata dal service mentre l'app (o meglio, le Activity dell'app) sono in background. Questo evento contiene l'indicazione se il service fosse o meno in pausa: l'utilità è quella di ottenere un'indicazione su quanto e in che caso venga utilizzata la notifica, per magari andare in futuro ad aggiungere funzionalità come la presenza di pulsanti nella notifica per fermare o mettere in pausa il workout senza entrare nell'applicazione, o fornire indicazioni più dettagliate sullo svolgimento dell'allenamento
- **settings\_update** è l'evento scatenato dall'update dei settings utente (age, height e weight), e contiene il numero di incrementi e decrementi fatti dall'utente per ogni categoria. Il numero di incrementi è utile, nel caso l'utente aumenti o diminuisca di molto ogni volta una certa categoria, per andare ad aggiungere dei pulsanti per aggiungere o rimuovere più di 1 alla volta alla proprietà (magari fare un -10 e +10 per l'altezza e il peso). Il numero di volte in cui questo evento è generato può essere un indicatore, se l'utente imposta solo una volta tali impostazioni all'installazione dell'app e poi le utilizza solo molto raramente, del fatto che potrebbe essere sensato far inserire subito all'utente le sue generalità, per poi spostare questo Fragment in una sezione delle impostazioni, in modo da liberare spazio per inserire un'altra funzionalità.
- **goals\_update** è l'evento scatenato dall'update dei goals (di calorie, passi o distanza), e contiene anch'esso il numero di incrementi e decrementi. Oltre ad essere utile per gli stessi motivi di `settings_update`, questo evento permette di comprendere quanto spesso gli utenti cambino i goals, e quindi se utilizzino l'app per stimolarsi a fare sempre di più, nel qual caso potrebbe essere un'idea quella di inserire un suggerimento sull'aumento dei goals con cadenza prestabilita.

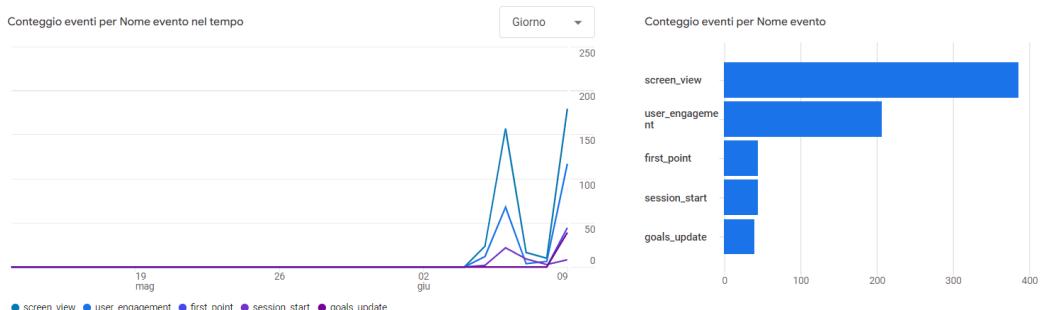


Figure 17.11: Grafico di alcuni eventi

Nome evento	↓ Conteggio eventi		Totale utenti	Conteggio eventi per utente		Entrate totali
	950 100% del totale	30 100% del totale		31,67 Uguale alla media		
1 <a href="#">screen_view</a>	386	30		12,87	0,00 \$	
2 <a href="#">user_engagement</a>	207	21		9,86	0,00 \$	
3 <a href="#">first_point</a>	44	7		6,29	0,00 \$	
4 <a href="#">session_start</a>	44	29		1,52	0,00 \$	
5 <a href="#">goals_update</a>	39	7		5,57	0,00 \$	
6 <a href="#">added_workout</a>	37	7		5,29	0,00 \$	
7 <a href="#">settings_update</a>	36	8		4,50	0,00 \$	
8 <a href="#">click_notification</a>	32	1		32,00	0,00 \$	
9 <a href="#">workout_saved</a>	30	6		5,00	0,00 \$	
10 <a href="#">first_open</a>	29	29		1,00	0,00 \$	
11 <a href="#">app_remove</a>	22	22		1,05	0,00 \$	
12 <a href="#">workout_not_saved</a>	15	5		3,00	0,00 \$	
13 <a href="#">workout_deleted</a>	11	3		3,67	0,00 \$	
14 <a href="#">app_exception</a>	8	3		2,67	0,00 \$	

Figure 17.12: Eventi più frequenti

# **Part VII**

## **Sitografia**

## Introduzione

Vengono riportati tutti i link relativi alle pagine web da cui provengono le informazioni e le immagini dei primi capitoli

- [1] <https://www.treccani.it/enciclopedia/telemetria/>
- [2] <https://it.wikipedia.org/wiki/Telemetria>
- [3] <https://www.internet4things.it/iot-library/telemetria-cose-a-cosa-servi-principali-applicazioni>
- [4] [https://www.splunk.com/en\\_us/blog/learn/observability-vs-monitoring-vs-telemetry.html](https://www.splunk.com/en_us/blog/learn/observability-vs-monitoring-vs-telemetry.html)
- [5] <https://aws.amazon.com/it/compare/the-difference-between-monitoring-and-observability/>
- [6] <https://www.flworld.it/tecnica-f1-cosa-e-la-telemetria-e-come-funziona/>
- [7] <https://www.nurse24.it/dossier/diabete/cgm-sistema-monitoraggio-continuo-glicemia.html>
- [8] [https://www.ilmessaggero.it/animali/orso\\_m49\\_scappato\\_trento\\_ultime\\_notti-5370688.html](https://www.ilmessaggero.it/animali/orso_m49_scappato_trento_ultime_notti-5370688.html)
- [9] <https://dzone.com/articles/everything-software-developers-need-to-know-about>
- [10] <https://www.techtarget.com/whatis/definition/telemetry>
- [11] [https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index\\_en.htm](https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_en.htm)
- [12] <https://www.zerounoweb.it/mobility/aumenta-numero-degli-italiani-internet-soprattutto-mobile/>
- [13] <https://www.nytimes.com/2019/01/03/technology/weather-channel-app-lawsuit.html>
- [14] <https://www.nytimes.com/2019/12/22/us/politics/totok-app-uae.html>
- [15] [https://www.kaspersky.it/about/press-releases/2023\\_kaspersky-segnala-piu-di-340000-attacchi-con-una-nuova-mod-malevola-di-whatsapp](https://www.kaspersky.it/about/press-releases/2023_kaspersky-segnala-piu-di-340000-attacchi-con-una-nuova-mod-malevola-di-whatsapp)
- [16] <https://thehackernews.com/2024/05/android-15-introduces-new-features-to.html>
- [17] <https://arcticwolf.com/resources/glossary/malicious-apps/>
- [18] [https://link.springer.com/chapter/10.1007/978-3-030-90022-9\\_12](https://link.springer.com/chapter/10.1007/978-3-030-90022-9_12)

## Strumenti

Per tutti gli strumenti è stato messo un singolo link alla documentazione che rappresenta un punto da cui partire per ottenere tutte le informazioni raccolte. Dato che la documentazione è ampia, si è preferito non appesantire l'insieme di URL. Sono stati aggiunti link anche per gli strumenti non oggetto di capitoli, ma citati.

- [19] Spiegazione logs, tracce, span, eventi: <https://medium.com/dzerolabs/observability-journey-understanding-logs-events-traces-and-spans-836524d63172>
- [20] Spiegazione metriche: <https://last9.io/blog/understanding-metrics-events-logs-traces-key-pillars-of-observability/>
- [21] Spiegazione RUM: <https://www.dynatrace.com/news/blog/what-is-real-user-monitoring/>
- [22] APM: <https://www.dynatrace.com/news/blog/what-is-apm-2/>
- [23] Android Performance Tuner: <https://developer.android.com/games/sdk/performance-tuner?hl=it>
- [24] Android Vitals: <https://developer.android.com/topic/performance/vitals?hl=it>
- [25] Elenco strumenti APM:  
<https://nandbox.com/5-best-mobile-app-monitoring-tools-to-track-app-performance/>  
<https://www.techrepublic.com/article/apm-monitoring-tools/>  
<https://stackify.com/application-performance-management-tools/>  
<https://www.techradar.com/news/best-apm-tool>
- [26] Firebase: <https://firebase.google.com/docs/perf-mon?hl=it>
- [27] Sentry: <https://docs.sentry.io/platforms/android/>
- [28] LogRocket: <https://docs.logrocket.com/docs/introduction-to-logrocket-mobile>
- [29] Datadog: [https://docs.datadoghq.com/integrations/rum\\_android/](https://docs.datadoghq.com/integrations/rum_android/)
- [30] Dynatrace: <https://www.dynatrace.com/technologies/android-monitoring/>
- [31] AppDynamics: <https://docs.appdynamics.com/appd/4.5.x/en/end-user-monitoring/mobile-real-user-monitoring/instrument-android-applications>
- [32] Raygun: <https://raygun.com/documentation/language-guides/android/>
- [33] Instana: <https://www.ibm.com/docs/en/instana-observability/current?topic=applications-android-api>
- [34] Dotcom Monitor: <https://www.dotcom-monitor.com/>
- [35] Scout APM: <https://www.scoutapm.com/>
- [36] Stackify Retrace: <https://stackify.com/retrace-application-performance-management/>
- [37] Application Insights: <https://learn.microsoft.com/it-it/azure/azure-monitor/app/app-insights-overview>
- [38] LogicMonitor: <https://www.logicmonitor.com/>

- [39] ManageEngine Applications Manager: [https://www.manageengine.com/products/applications\\_manager/](https://www.manageengine.com/products/applications_manager/)
- [40] Microsoft SCOM: <https://learn.microsoft.com/it-it/system-center/scom/manage-operations-guide-overview?view=sc-om-2022>
- [41] FogLight: <https://www.quest.com/foglight/>
- [42] JenniferSoft: <https://jennifersoft.com/en/>
- [43] CA APM: <https://techdocs.broadcom.com/us/en/ca-enterprise-software/it-operations-management/application-performance-management/10-7.html>
- [44] AppOptics APM: [https://documentation.solarwinds.com/en/success\\_center/appoptics/content/kb/apm\\_tracing/using\\_apm.htm](https://documentation.solarwinds.com/en/success_center/appoptics/content/kb/apm_tracing/using_apm.htm)
- [45] Sumo Logic: <https://www.sumologic.com/>
- [46] NewRelic: <https://newrelic.com/platform/application-monitoring>
- [47] Riverbed Aternity: <https://support.riverbed.com/content/support/software/steelcentral-ap/aternity/mobile.html>
- [48] Instabug: <https://www.instabug.com/platforms/android>