Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Università Degli Studi di Padova
Programmazione di Sistemi Embedded

Telemetry

(Un)Health

Enrico Trettenero | Francesco Pizzolato | Leonardo Callegari

Anno Accademico 2023 - 2024

Indice

Indice **1**

1 Introduzione **3**

1.1 Gli obiettivi della Telemetry 6

1.2 Privacy, Tiktok e politica 11

1.3 Breve storia del concetto di Telemetry 14

2 Android: Permessi e Autorizzazioni **17**

2.1 Evoluzione permessi Android 17

2.2 Implementazione permessi 25

3 Telemetria in Android **29**

3.1 Android Performance Tuner (APT) 29

3.2 AppDynamics 30

3.3 Dynatrace 31

3.4 New Relic 33

3.5 Sentry 35

3.6 Datadog 36

3.7 Dotcom-Monitor 36

3.8 Instana 37

3.9 Logic Monitor 38

3.10 Raygun 39
3.11 LogRocket 40
3.12 Firebase 42

4 La nostra applicazione, (Un)Health 53

4.1 Idea generale 54
4.2 Organizzazione delle interazioni utente 55
4.3 UI Design 64
4.4 Design Pattern 66
4.5 Database 66
4.6 API 67
4.7 Note 68

Bibliografia 71

Capitolo 1

Introduzione

L'obiettivo che questo report si propone di raggiungere consiste nell'approfondimento di tematiche estremamente importanti nell'ottica dello sviluppo di un'applicazione mobile, soprattutto per quello che riguarda la gestione dei dati riguardanti gli utenti, lo stato dei dispositivi utilizzati e la sicurezza delle informazioni che vengono condivise nei sistemi distribuiti mediante la rete.

Questi aspetti si possono riassumere nel concetto di Telemetria, che tuttavia presenta sfaccettature estremamente varie, talvolta non sempre chiare al grande pubblico di utenti che utilizzano quotidianamente le applicazioni con l'obiettivo di usufruire dei servizi tangibili che esse dichiarano di mettere a disposizione. Tutti questi servizi sono molto spesso offerti in maniera ingannevolmente gratuita, con un costo che spesso non è monetario ma quanto più relativo ai dati raccolti. La cosa importante in questo contesto è il fatto che spesso l'utente è inconsapevole dei dati che sta fornendo all'applicazione oppure, in altre situazioni, l'utente sa di fornire i propri dati, ma non è a conoscenza dei dettagli, relativamente a quali sono effettivamente i dati che vengono raccolti, inviati e successivamente elaborati.

L'obiettivo che ci siamo posti non è quello di creare un qualche tipo di al-

larmismo o manifestare teorie del complotto contro le grandi aziende che forniscono i servizi che usiamo costantemente, ma piuttosto è quello di rendere consapevoli i lettori sia degli aspetti positivi che il concetto di Telemetry porta con sé, ma anche degli aspetti negativi relativamente all'utilizzo (potenzialmente) malevolo dei dati che vengono collezionati da queste aziende.

Per citare un esempio, una ricerca condotta dalla società "Snoopfall[1]" (che opera nel settore della sicurezza informatica) ha mostrato che le prime dieci applicazioni per l'utilizzo della torcia, scaricabili dal Google Play Store, chiedono una serie di permessi che si estendono molto oltre le funzionalità che l'applicazione si propone di offrire all'utente che ne fa uso. In particolare, lo studio evidenzia che:

> «[...] *possono (le applicazioni), ad esempio, accedere ad informazioni personali e riservate del dispositivo e del possessore del dispositivo; controllare gli spostamenti dell'utente grazie alla triangolazione dei dati del GPS e della rete cellulare; accedere alla fotocamera, scattare foto e girare filmati; leggere, scrivere e cancellare file; fare chiamate; accedere alla rubrica; leggere o scrivere messaggi di posta elettronica, SMS e messaggistica istantanea; avere pieno accesso alle risorse di Rete e alle connessioni (attive e disponibili).*»

Tutti ragionevolmente possiamo renderci conto che questi dati non hanno alcuna correlazione con la funzionalità principale di un'applicazione il cui obiettivo è quello relativo alla torcia. Inevitabilmente la domanda sorge spontanea: qual'è il motivo che induce le aziende a raccogliere una quantità così

[1] Informazioni aggiuntive relative a questa ricerca possono essere collezionate qui.

estesa e dettagliata di informazioni? La risposta è tutt'altro che banale e si collega al concetto di costo che un utente deve pagare per utilizzare un applicazione, che in questo caso si riferisce all'insieme di tutti i dati che lo riguardano e che più o meno consapevolmente fornisce con l'unica finalità di utilizzare l'applicazione.

Flashlight Apps	Super-Bright LED Flashlight	Brightest Flashlight Free	Tiny Flashlight + LED	Flashlight	Flashlight	Brightest LED Flashlight	Color Flashlight	High-Powered Flashlight	Flashlight HD LED	Flashlight: LED Torch Light
Permissions										
retrieve running apps	✓					✓		✓		
modify or delete the contents of your USB storage	✓	✓				✓		✓		
test access to protected storage	✓	✓				✓		✓		
take pictures and videos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
view Wi-Fi connections	✓	✓				✓		✓	✓	
read phone status and identity	✓	✓			✓	✓		✓		
receive data from Internet	✓					✓		✓		
control flashlight	✓	✓	✓			✓	✓	✓	✓	
change system display settings	✓					✓		✓		
modify system settings	✓					✓		✓		
prevent device from sleeping	✓							✓		
view network connections	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
full network access	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
approximate location (network-based)	✓	✓						✓		
precise location (GPS and network-based)	✓	✓						✓		
disable or modify status bar	✓	✓								
read Home settings and shortcuts	✓	✓		✓						✓
install shortcuts	✓	✓		✓						✓
uninstall shortcuts	✓	✓		✓						✓
control vibration	✓		✓							
prevent device from sleeping		✓	✓	✓		✓			✓	✓
write Home settings and shortcuts				✓						✓
disable your screen lock				✓						✓
read Google service configuration					✓				✓	

Figura 1.1: *Analisi dei permessi di 10 applicazioni della torcia presenti in Google Play Store*

Un altro studio dell'università statunitense *Carnegie Mellon* ha analizzato termini e condizioni di utilizzo delle 100 applicazioni più scaricate dal Google Play Store, scoprendo, ad esempio, che applicazioni come Backgrounds HD Wallpaper, Brightest Flashlight, Dictionary.com, Google Maps, Horoscope, Mousetrap, Pandora, Shazam e Talking Tom Virtual Pet sono le più esigenti da un punto di vista delle richieste di accesso a dati personali e funzionalità. Ma, se da un lato appare del tutto normale che Google Maps chieda accedere al GPS del dispositivo oppure che Shazam voglia poter attivare il microfono

quando è in uso, dall'altro appare invece meno chiaro il perché Talking Tom richieda gli stessi permessi.

Ancora una volta è bene precisare che l'obiettivo non è tanto quello di criticare grandi aziende su come i nostri dati vengano sfruttati, mantenuti, utilizzati e/o venduti, ma è piuttosto quello di attirare l'attenzione del lettore per costruire la giusta consapevolezza di fronte ad uno strumento, quello della Telemetry, che è poco conosciuto lato utente e troppo spesso mascherato in un modo che lo mostra solo per quello che di positivo lo riguarda.

1.1 Gli obiettivi della Telemetry

Telemetry, per come è stato concepito e sviluppato, permette di tracciare la salute di un sistema da remoto e sfruttare i dati raccolti per operare elaborazioni che hanno lo scopo di potenziare l'infrastruttura aziendale, informatica o economica. Essa permette di monitorare, mantenere e ottimizzare i propri sistemi nel corso del tempo. Questo strumento è in continua evoluzione e viene sempre più impiegato per ricevere un feedback che sia utile al miglioramento di uno specifico servizio, ad esempio tramite il tracciamento dei dati di utilizzo di un'applicazione.

Telemetry letteralmente significa "remote transmission of measurement data". In altre parole, si riferisce a processi di comunicazione automatizzata di varie fonti di dati con i seguenti scopi:

- Migliorare l'esperienza utente
- Monitorare lo stato e la salute di un'applicazione
- Misurare la qualità e le prestazioni di un'applicazione

- Tracciale le analytics di un servizio web o di un social network
- Mantenere e garantire la sicurezza dei dati che vegono raccolti e impiegati per lo sviluppo e il miglioramento di un'applicazione

L'ultimo punto è forse quello più controverso. Spesso accade che i dati raccolti non vengono gestiti come dichiarato. Altre volte succede che vengono utilizzati in maniera malevola, diversamente da quanto viene presentato all'utente.

Per chiarire meglio il ruolo della Telemetry all'interno di un'applicazione, possiamo paragonarla ad un insieme di sensori di una sorgente remota che misurano dei dati fisici come ad esempio temperatura, pressione, umidità o dati di carattere elettrico come tensioni e/o correnti. Questi dati vengono poi serializzati per formare un flusso di informazioni che viene trasmesso, mediante l'utilizzo di un mezzo fisico che può o meno essere cablato a seconda dei casi, e poi ricevuto da un ricevitore che li decomprime e decodifica con l'obiettivo di organizzare tutti i dati in delle opportune strutture dati che permettano di effettuare elaborazioni più o meno complesse, con la finalità di produrre statistiche che siano utili, relativamente alle caratteristiche del sistema in esame.

In generale, la telemetria permette quindi di produrre ed elaborare informazioni strutturate all'interno di uno specifico ambiente di analisi. L'obiettivo è sia quello di studiare il comportamento delle componenti hardware e software del sistema applicativo considerato, ma anche di ricevere feedback diretto da parte degli utenti, relativamente ai servizi offerti. Il team di sviluppo può poi fare riferimento a queste informazioni per fare scelte di tipo implementativo, relativamente all'aggiornamento di funzionalità esistenti che possono essere migliorate oppure per funzionalità non presenti ma che vengono richieste da-

gli utenti. Alcune delle questioni che la telemetria cerca di affrontare sono riportate di seguito.

- Gli utenti utilizzano una specifica funzionalità nelle modalità previste nella fase di progettazione e sviluppo?
- In che modo gli utenti interagiscono con i servizi che l'applicazione mette a disposizione? E quanto frequentemente avvengono queste interazioni? Oppure ancora, quale è mediamente la durata di una specifica interazione?
- Mediamente, quanto è il tempo di utilizzo dell'applicazione? In quali contesti viene impiegata maggiormente?
- Quali sono le impostazioni e funzionalità che vengono utilizzate con maggiore frequenza da una larga fetta di utenti? Quali sono le preferenze relativamente alle configurazioni dell'applicazione? (Ad esempio orientamento dello schermo, modalità di input, scelta del tema e dei colori, scelta dei caratteri e della dimensione del testo, ...)
- Quali sono le condizioni che determinano i crash del sistema? Quali sono le funzionalità più vulnerabili e che generano i problemi maggiori dal punto di vista tecnico o dell'esperienza d'uso?

Senza l'utilizzo della Telemetry tutte queste informazioni non potrebbero essere raccolte, e questo potrebbe risultare in un problema nel momento in cui un sistema è distribuito su larga scala a milioni di utenti, come accade per le applicazioni mobile, che conseguentemente determina interessi economici non indifferenti.

Quasi sempre le applicazioni sono sistemi dinamici in cui la cosa importante è effettuare aggiornamenti correttivi mirati e aggiungere nuove funzionalità,

per avere dei vantaggi nel mercato rispetto alla concorrenza. E la telemetria aiuta enormemente in questo senso, anche relativamente alla velocità di sviluppo.

Troppo spesso però la diffidenza degli utenti, che non vogliono investire tempo per informarsi, diventa un ostacolo. Ed allora accade che, se si volesse fare un paragone letterario, il concetto di telemetria viene associato ad una sorta di "Big Brother" del romanzo 1984 di George Orwell e dunque viene visto come uno strumento di controllo delle masse. La conseguenza è il blocco o il rifiuto di alcuni dei servizi che gli applicativi mettono a disposizione e che nella maggior parte dei casi hanno finalità benevole. Ciò costituisce un problema non indifferente per chi sviluppa software in maniera costruttiva, senza l'obiettivo di ricavare dati per effettuare della profilazione utente o per venderli al miglior offerente e guadagnarci, perché rallenta notevolmente il processo di risoluzione dei bug di un'applicazione e il processo di evoluzione naturale di un'applicazione che si propone di estendere le proprie funzionalità nel corso del tempo.

Non c'è purtroppo una soluzione vera e propria alla questione appena presentata, però è sicuramente più utile ricavare pochi dati in maniera costruttiva, grazie agli strumenti della Telemetry, piuttosto che non ricavarne nessuno e non progredire nello sviluppo. Prima di integrare uno strumento di Telemetry all'interno di un prodotto software, è bene chiedersi quali sono le informazioni che si vogliono ottenere dall'attività degli utenti e dai dispositivi. Collezionare dati, se gestito in maniera superficiale o riduttiva, può portare a soluzioni errate dal punto di vista implementativo; per non parlare poi dell'aspetto legale associato alla privacy e alla sicurezza degli utenti. Ad esempio, secondo uno studio del Lecture Notes of the Institute for Computer Sciences, Social Infor-

matics and Telecommunications Engineering (LNICST)[2] sia iOS che Google raccolgono telemetria anche quanto l'utente tenta di impedirlo. Google raccoglie notevolmente più dati rispetto ad Apple. Nei primi 10 minuti dall'avvio, lo smartphone Pixel invia circa 1 MB di dati a Google, contro i 42 KB inviati da un iPhone ad Apple. Quando i dispositivi sono inattivi, il Pixel invia circa 1 MB di dati ogni 12 ore, mentre l'iPhone invia solo 52 KB. Google raccoglie circa 20 volte più dati rispetto ad Apple. Delle stime del 2020 suggeriscono che negli USA ci siano 113 milioni di utenti iPhone e 129 milioni di utenti Android. Questo significa che Apple colleziona circa 5,8 GB di dati ogni 12 ore, mentre Google circa 1,3 TB. Entrambi i sistemi si connettono ai loro server di backend ogni 4 minuti e mezzo, anche quando il dispositivo non è in uso.

Fortunatamente, analizzando un'altra ricerca nata dalla collaborazione di alcune università della Gran Bretagna[3], il cui obiettivo era quello di analizzare una serie di smartphone con sistemi operativi differenti e i dati che venivano trasmessi al provider del sistema operativo, è stato notato che sebbene la trasmissione verso i server remoti proprietari sia massiccia, nei sistemi operativi principali la comunicazione almeno è cifrata. Un'altra ipotesi che è stata avanzata da tali ricercatori riguardo uno scenario piuttosto pessimista per quanto concerne il trattamento dei dati dei consumatori è la seguente: tutte le componenti software coinvolte nei dispositivi hanno la capacità di scambiare dati e, combinando ID persistenti e temporanei, e diversi tipi di telemetria, creano il quadro più completo possibile delle abitudini e delle preferenze degli utenti.

[2]Informazioni aggiuntive relative a questa ricerca possono essere collezionate qui.

[3]Informazioni aggiuntive relative a questa ricerca possono essere collezionate qui.

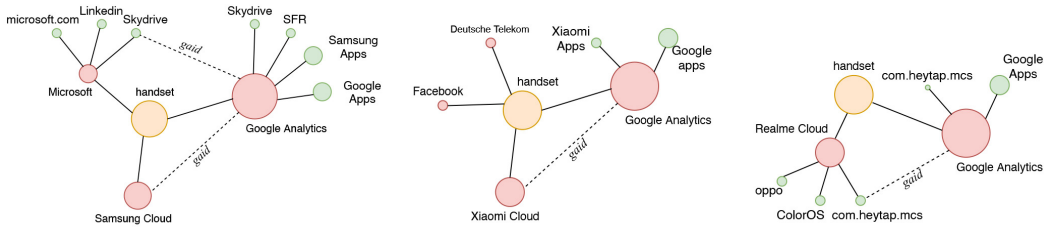
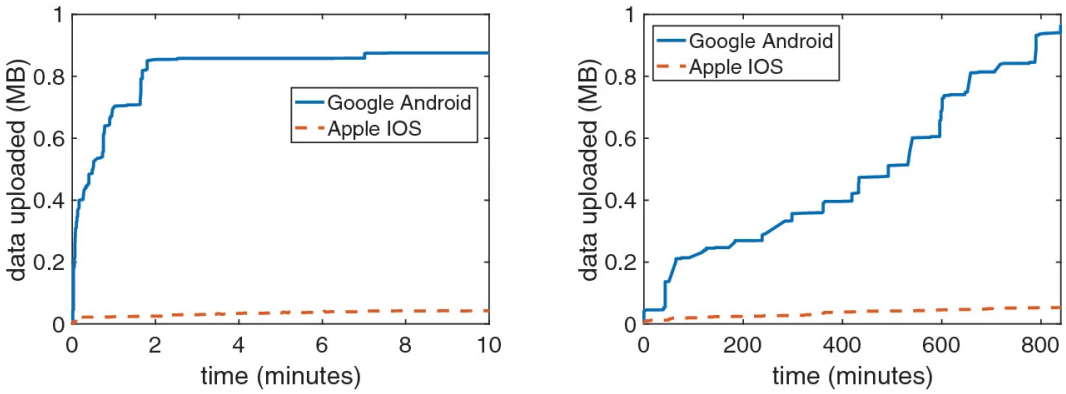
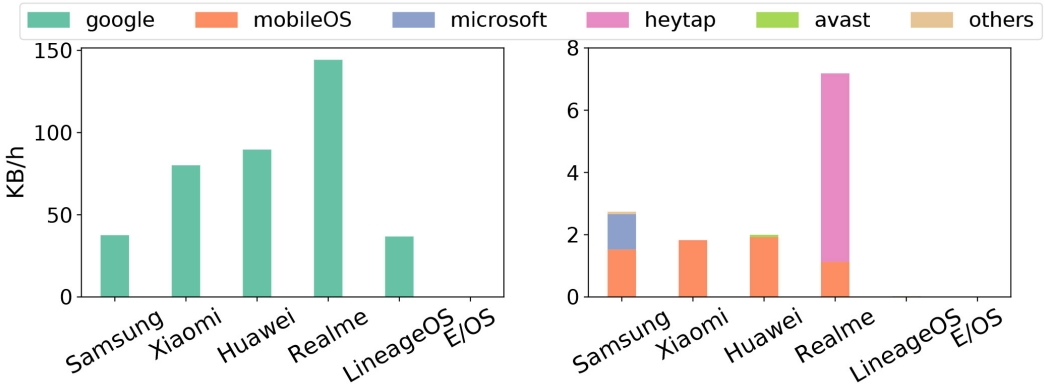


Figura 1.2: *I ricercatori speculano sulla possibilità di combinare le serie di dati per creare un profilo completo del proprietario dello smartphone (gaid sta per Google Advertising ID)*



(a) Startup (b) Idle



1.2 Privacy, Tiktok e politica

Uno degli argomenti più caldi degli ultimi mesi, soprattutto oltreoceano, e di cui siamo venuti a conoscenza in un modo o nell'altro tramite social o tele-

giornali, è stato l'enorme dibattito riguardante il ban dell'applicazione TikTok dal mercato digitale statunitense, che ha destato particolare indignazione e caos tra le folle digitali di utenti che ogni giorno, evidentemente, si riversano su questa piattaforma. TikTok, apparentemente, sembra essere un'innocua (e demenziale, anche se noi stessi siamo i primi a farne uso) applicazione, nata con il nome di "Musically" per chi ormai e' vecchio come noi, che inizialmente permetteva la condivisione di balletti e che poi si è espansa per coprire anche altre tematiche come sport, meme, musica, cucina e molto altro, per sfociare molto spesso nel *cringe*, termine utilizzato dai millenials per manifestare una sorta di sensazione mista tra l'imbarazzo e il fastidio. Tra i contenuti fruibili sulla piattaforma, un esempio che calza a pennello è quello delle "*live NPC*", in cui delle persone si riprendono mentre si comportano come dei robot in attesa di donazioni che, purtroppo (contro ogni logica), arrivano spesso. Oltre ad essere potenzialmente pericoloso per il nostro corpo a causa degli eccessivi rilasci di dopamina che questi social causano e ad abbassare la soglia dell'attenzione degli utenti, TikTok pare esser diventato un cliente molto scomodo a livello socio-politico; non solo per l'influenza economica su alcuni mercati, come quello musicale, che ha visto TikTok lanciare dei trend mondiali, ma anche per quanto riguarda la privacy degli utenti, non essendo a conoscenza di come e dove i dati vengono immagazzinati. Secondo uno studio di una società australiana di cybersecurity [4] l'utilizzo del social network comporterebbe una continua condivisione delle informazioni con un server cinese appartenente a una delle prime cento aziende al mondo che si occupano di sicurezza informatica e dei dati, la *Guizhou Baishan Technology Co.* Il procedimento di condivisione dei dati è stato rilevato da un attento studio sulle versioni 25.1.3

[4]Informazioni aggiuntive relative a questa ricerca possono essere collezionate qui.

per Android e 25.1.1 per iOS, grazie a un'analisi statica e dinamica del codice. Tutto ciò avviene semplicemente accettando le condizioni d'uso al primo accesso alla piattaforma. A seguire riportiamo la lista delle informazioni a cui il colosso cinese può accedere:

- Geolocalizzazione
- Calendario
- Contatti
- Connessione (Informazioni relative al Wi-Fi, l'operatore telefonico, persino il Mac Address)
- Iscrizioni a servizi, prodotti, siti online, abbonamenti
- Tracciamento della clipboard, ovvero l'interazione con la tastiera degli utenti con il proprio smartphone

Tutto ciò, se elencato in maniera esplicita e *"brutale"* come fatto precedentemente, può sembrare un vero e proprio incubo, ma non e' assolutamente distante da ciò che altre applicazioni fanno, come descritto precedentemente nel report. Purtroppo, però, la mancata consapevolezza degli utenti riguardo la protezione dei propri dati e il divertimento garantito da TikTok vincono sulla privacy. La colpa non è certamente degli utenti, però si potrebbe aprire un enorme dibattito sulla stesura delle condizioni d'uso. Sarebbe certamente più utile, intuitivo e di maggior impatto trovare una comunicazione immediata e diretta sulle informazioni a cui un servizio accede, piuttosto che scrivere centinaia di pagine sulle condizioni e termini d'uso a cui nessuno presta la minima attenzione. Ciò, ovviamente, dovrebbe essere imposto a livello legale, altrimenti nessuna azienda lo farebbe perché significherebbe suicidarsi a livello

imprenditoriale ed economico.
Sembrano quindi scontate e banali le motivazioni che spingono gli Stati Uniti a eliminare TikTok dal proprio mercato. Rischiare di condividere informazioni in ambito professionale, economico piuttosto che militare/strategico alla seconda potenza mondiale dietro proprio agli Stati Uniti non sembra essere molto allettante per la sicurezza del paese agli occhi della politica a stelle e strisce.

E' fondamentale perciò, nel momento in cui si manipolano informazioni provenienti dagli utenti, che la protezione di tali sia sempre assicurata e che dati personali non arrivino a terzi, quali hacker. Bisogna quindi agire con estrema cautela per evitare gravi conseguenze che possano ledere la privacy dei consumatori. Sicuramente grandi sistemi come Apple e Google avranno delle infrastrutture molto stabili e sicure, ma ciò deve valere anche per le aziende più piccole che sviluppano applicazioni Android ma non solo.

1.3 Breve storia del concetto di Telemetry

Lo sviluppo di soluzioni e framework associati alla Telemetry è un aspetto relativamente recente legato allo sviluppo di software, nato circa a partire dagli ultimi anni del 1990 e i primi del nuovo millennio. Tutto è nato dall'esigenza di monitorare l'evoluzione di tecnologie informatiche che stavano diventando sempre più stratificate e complesse, in particolare con l'avvento delle prime applicazioni web, dei server e di un utilizzo sempre più massiccio di database che diventano costantemente più complessi e artificiosi. Inizialmente gli sviluppatori si affidavano principalmente a strumenti rudimentali e personalizzati per valutare le prestazioni delle loro applicazioni. Ad esempio, la gestione dei log

veniva implementata manualmente e si analizzavano statistiche estremamente basilari come il tempo di risposta di una query o l'utilizzo di risorse come CPU e memoria. Talvolta venivano eseguiti addirittura test manuali su pool di utenti. Con l'avvento del secondo millennio la complessità delle applicazioni ha raggiunto livelli senza precedenti, in particolare grazie all'espansione del Cloud Computing: con l'aumento della complessità delle applicazioni distribuite è diventato sempre più difficile monitorare le prestazioni in modo efficace, in quanto il comportamento dei sistemi si stava sempre più allontanando dalla gestione diretta degli sviluppatori e gli strumenti di base non erano più sufficienti ad analizzare le performance dei software. Le funzionalità principali degli strumenti di APM (Application Performance Monitoring) si concentravano sulla registrazione delle transazioni end-to-end da client a server, seguendo tutto il workflow e stabilendone l'efficienza. Il primo leader nel settore degli APM è Wily Technology grazie alla release nel 1998 di Introscope, primo APM definibile commerciale, che è stao seguito da DynaTrace (fondata nel 2005), pioniera nell'uso della tecnologia di *"PurePath"* per il tracciamento delle transazioni: questa tecnologia permetteva un tracciamento granulare e dettagliato delle transazioni end-to-end, consentendo agli sviluppatori e agli amministratori di sistema di vedere ogni singolo passo di una transazione e identificare rapidamente i colli di bottiglia. Infine, troviamo New Relic, di cui parleremo successivamente nel report, fondata da da Lew Cirne (precedentemente fondatore di Wily Technology), rivoluzionaria nel campo delle APM grazie ad un approccio Software-as-a-Service (SaaS), che ha reso New Relic l'APM più accessibile e facile da usare per una vasta gamma di aziende, incluse quelle senza risorse IT. Queste aziende posero le basi per le evoluzioni avvenute nel decennio successivo. Con l'esplosione delle applicazioni mobili e del software come servizio (SaaS), la necessità di monitorare le prestazioni delle applica-

zioni è diventata ancora più critica. Gli utenti esigevano esperienze veloci e senza intoppi su una vasta gamma di dispositivi e piattaforme. Le soluzioni APM si sono evolute per supportare una vasta gamma di tecnologie, inclusi cloud pubblici, containerizzazione, microservizi e architetture serverless. Sono emerse nuove tecniche di monitoraggio, come l'analisi dei dati in tempo reale e l'intelligenza artificiale per l'analisi dei problemi delle prestazioni.

Ad oggi, come vedremo successivamente nell'analisi di varie piattaforme di monitoring delle performance, l'accento è posto su due aspetti principali: automazione ed esperienza utente. La prima grazie a strumenti di intelligenza artificiale sempre più dettagliati, curati e maturati nel tempo, mentre la seconda grazie a strumenti di analisi sempre più personalizzati e soggettivi, che si avvicinano alle esigenze percepite dall'essere umano. Inoltre, un aspetto che abbiamo potuto toccare con mano è stata la facilità con cui questi framework possono essere integrati. Nel nostro caso, quindi attraverso la piattaforma FireBase, è stata sufficiente una configurazione delle dependency del progetto e della dashboard offerta dal software per poter visualizzare tutte le varie statistiche. Questa semplicità d'uso è un altro aspetto che rende sicuramente vincente un prodotto di questo tipo, e che fa felici noi ingegneri informatici e sviluppatori di applicazioni.

Capitolo 2

Android: Permessi e Autorizzazioni

2.1 La storia del sistema di permessi e autorizzazioni di Android

Nel corso del tempo il concetto di telemetry si è evoluto. Gli strumenti a disposizione degli sviluppatori sono sempre di più. Con lo sviluppo continuo di questi strumenti, che permettono di raccogliere sempre più dati, Android si è evoluto di pari passo, nel tentativo di proteggere gli utenti, iniziando ad apportare modifiche che hanno cambiato radicalmente le modalità con cui le applicazioni possono raccogliere i dati degli utenti che ne fanno uso.

Con lo sviluppo continuo del reparto hardware, i dispositivi hanno acquisito sempre più strumenti nel corso del tempo. Oggi un dispositivo può utilizzare l'accelerometro, il giroscopio, il GPS e i servizi di localizzazione associati, il microfono e gli speaker, e anche un magnetometro che svolge la funzione di bussola. Insomma con i dispositivi che abbiamo a disposizione oggi è difficile

perdersi. E se da un lato questa è una cosa estremamente positiva, dall'altro bisogna assumere consapevolezza che tutti gli strumenti che abbiamo a disposizione hanno bisogno di dati per funzionare. Hanno bisogno dei dati dell'utente. E se da un lato è molto comodo che il telefono ci possa dare indicazioni sul ristorante più vicino alla nostra posizione corrente, dall'altro bisogna essere consapevoli che questo tipo di informazioni sono raccolte e utilizzate di continuo dalle applicazioni che le sfruttano per fornire i loro servizi. Questo non significa che l'utilizzo di questi dati sia necessariamente malevolo, ma piuttosto che i dati sono spesso raccolti senza che l'utente ne sia consapevole. In tanti casi l'utente è a conoscenza della raccolta di questi, ma non li ritiene abbastanza importanti per iniziare a informarsi a riguardo.

L'aumento nel numero e nella tipologia dei sensori di un dispositivo mobile, che conseguentemente ha determinato un aumento nei dati raccolti dagli applicativi, richiede sempre una maggiore attenzione sia da parte degli sviluppatori che degli utenti finali.

Prima che Android lanciasse la versione 5.1 (Lollipop) di Android alla fine del 2014, gli applicativi richiedevano all'utente i permessi di accedere alle funzionalità del dispositivo al momento dell'installazione. Per l'utente questa richiesta di permessi era nascosta nel processo di installazione dell'applicazione. Ed ecco che con questo sistema, non c'era modo di revocare i permessi una volta che l'applicazione era installata (se non disinstallando l'applicativo, che però avrebbe impedito di sfruttarne le funzionalità), oppure di scegliere con cura quali di quelli dovessero essere abilitati e quali rifiutati. Al tempo il problema aveva un'importanza minore rispetto ad oggi, perché erano incredibilmente meno i dati che un'applicazione raccoglieva.

Con lo sviluppo delle versioni successive di Android, si è capito che la tematica dei permessi e delle autorizzazioni cominciava ad assumere una tale

importanza da richiedere delle modifiche sostanziali al sistema che li gestiva. Per questo motivo, a partire da Android 6 (Marshmallow), che è stato reso disponibile nella seconda metà del 2015, le applicazioni devono controllare ogni permesso che richiedono a tempo di esecuzione (runtime). Questo modello offre agli utenti maggiore visibilità e controllo sulle autorizzazioni, semplificando al contempo i processi di installazione e aggiornamento automatico per gli sviluppatori di app. Nella situazione in cui uno o più permessi non siano abilitati per l'applicativo, una richiesta viene fatta all'utente per abilitarli. Inoltre l'utente ha la possibilità di revocare i permessi in maniera selettiva in ogni momento.

Questo cambiamento radicale rispetto alle versioni precedenti ha fornito all'utente un controllo molto maggiore su quelli che sono i permessi che vengono abilitati per gli applicativi. Come conseguenza, l'utente ha cominciato ad acquisire più consapevolezza sui dati che un'applicazione era in grado di raccogliere. Nonostante questi cambiamenti, spesso accadeva che gli utenti abilitavano i permessi richiesti senza tanto indugiare, con l'unico scopo di utilizzare l'applicativo quanto prima. Alla fine un utente installa un'applicazione per il servizio che offre. E se ritiene di aver bisogno di quel servizio, allora chiedere all'utente di abilitare i permessi, o dargli la possibilità di revocarli, assume un'importanza relativa, poiché l'utente li abiliterà la maggior parte delle volte senza porsi tante domande. Questo comportamento è lo stesso che si verifica quando un utente accetta i termini e le condizioni di utilizzo di un'applicazione. L'utente accetta i termini perché è un passaggio che gli viene chiesto di fare per usare le funzionalità dell'applicazione, e non perché gli abbia letti e abbia consapevolezza di quello che accetta.

Nell'ottica di aiutare gli utenti, ulteriori miglioramenti sono stati inseriti nelle più recenti versioni di Android. Per evitare che le applicazioni abusino

dei permessi abilitati da parte dell'utente in maniera malevola, la versione di Android 10 del settembre 2019 mette a disposizione dell'utente la possibilità di abilitare i permessi relativamente ai servizi di localizzazione solamente quando l'applicazione che li richiede viene utilizzata (cioè mentre è in esecuzione). In questo modo la raccolta dei dati è limitata alla finestra temporale di utilizzo dell'applicazione. Inoltre, sono state realizzate una serie di features con l'obiettivo di massimizzare la privacy dei consumatori, tra cui:

- **Accesso allo spazio di archiviazione esterno limitato all'ambito di file di app**

  Di default viene concesso l' accesso limitato all' unità di archiviazione esterna. Queste app possono accedere ai seguenti tipi di file all'interno di un dispositivo di archiviazione esterno senza dover richiedere autorizzazioni utente correlate allo spazio di archiviazione:

  - File nella directory specifica dell'app
  - Foto, video e clip audio creati dall'app dal media store

- **Limitazioni relative all'avvio di attività in background**

  Questo cambiamento del comportamento consente di ridurre al minimo le interruzioni per l'utente e di mantenere un maggiore controllo su ciò che viene mostrato sullo schermo.

- **Rimozione dell'affinità dei contatti**

  La piattaforma non tiene traccia delle informazioni sull'affinità dei contatti. Di conseguenza, se una applicazione esegue una ricerca nei contatti dell'utente, i risultati non vengono ordinati in base alla frequenza di interazione.

- **Randomizzazione degli indirizzi MAC**
- **Accesso limitato ai dati degli appunti**
- **Limitazione dell'accesso ai dettagli e ai metadati della videocamera**

  Android 10 cambia le informazioni predefinite restituite dal metodo getCameraCharacteristics(). In particolare, la tua app deve disporre dell'autorizzazione CAMERA per accedere a metadati potenzialmente specifici del dispositivo inclusi nel valore restituito di questo metodo.
- **Riconoscimento dell'attività fisica** Android 10 introduce l'autorizzazione di runtime `ACTIVITY_RECOGNITION` per le app che devono rilevare il numero di passi dell'utente o classificarne l'attività fisica, ad esempio camminata, ciclismo o spostamento in un veicolo. Questa funzionalità è progettata per offrire agli utenti visibilità su come vengono utilizzati i dati dei sensori del dispositivo nelle Impostazioni.

Con l'arrivo di Android 11 nel settembre 2020, un'ulteriore spinta è arrivata sul fronte permessi e autorizzazioni. A partire da questa versione, Android ha impostato la revoca automatica di tutti i permessi sensibili per applicazioni che non sono utilizzate. In questo modo, se un utente non utilizza un'applicazione per un periodo di tempo prolungato, non c'è il rischio che tale applicazione continui a raccogliere dati. Inoltre, è stata introdotta la possibilità di abilitare i permessi un'unica volta (one-time permission, attraverso l'opzione "Solo questa volta"). Questo significa che l'applicazione chiederà all'utente il permesso ogni volta che è necessario e fino a che l'utente seleziona di abilitare

il permesso un'unica volta, esso verrà revocato alla chiusura dell'applicazione e richiesto nuovamente quando necessario. Inoltre, sono state effettuate delle modifiche per quanto riguarda le autorizzazioni riguardanti l'accesso ai numeri di telefono del proprio dispositivo attraverso nuove autorizzazioni e classi per accedere alle API riguardo la gestione di essi.

Nelle versioni 12 e 13 di Android la gestione dei permessi per microfono e fotocamera è stata migliorata e permette all'utente di vedere un indicatore apposito mentre microfono e/o fotocamera vengono utilizzati. La libertà di revocare i permessi di utilizzo di questi ultimi è totale e l'utente può farlo nelle impostazioni delle singole applicazioni. In particolare: con la versione 13 di Android sono state introdotte:

- Nuova autorizzazione di runtime per i dispositivi Wi-Fi nelle vicinanze `NEARBY_WIFI_DEVICES`
- Autorizzazioni granulari per i contenuti multimediali:
  - `READ_MEDIA_IMAGES` per immagini e foto,
  - `READ_MEDIA_VIDEO` per video,
  - `READ_MEDIA_AUDIO` per file audio,

  con l'obiettivo di consentire un accesso più granulare ai file multimediali creati da altre app, sostituendo l'autorizzazione READ_EXTERNAL_STORAGE.

Negli ultimi tempi, un'attenzione particolare è stata posta sul reparto notifiche e sui permessi che le riguardano, con l'obiettivo di impedire alle applicazioni un utilizzo spropositato. Come abbiamo potuto analizzare, infatti, il

numero dei permessi e autorizzazioni e come essi vengono gestiti ha assunto un peso sempre di maggior importanza e rilievo. Lo dimostra uno studio dell'Universita' del Lussemburgo in collaborazione con *Qatar Computing Research Institute* [18]. Lo studio, nonostante sia del 2016 quindi piuttosto datato in termini di sviluppo della piattaforma Android, rappresenta in maniera efficace la crescita e l'andamento dei permessi attraverso le varie release delle API.

> "Abbiamo condotto un'analisi mirata alla rilevazione di anomalie nel sistema di permessi. Il nostro studio si concentra sui permessi della piattaforma Android e non copre i permessi personalizzati, definiti dalle applicazioni di terze parti per proteggere l'accesso alle proprie risorse. Abbiamo suddiviso i permessi della piattaforma in quattro categorie:
>
> - **Sample:** permessi dichiarati dalle app di esempio incluse nel codice sorgente della piattaforma (presenti dall'API 21).
> - **Test:** permessi dichiarati nei file manifest dei pacchetti sviluppati per scopi di test.
> - **Package:** permessi dichiarati in vari pacchetti che completano il framework, esclusi quelli dei gruppi test e sample.
> - **Core:** permessi dichiarati nel file manifest principale di Android situato nella cartella frameworks/base/core/res.
>
> Alcuni permessi (delle categorie core e package) sono i "veri" permessi usati per il controllo degli accessi, mentre altri sono ausiliari, utilizzati in applicazioni di esempio (sample) o per i test (test). Il nostro studio si concentra sui permessi core e package, poiché sono quelli che influenzano realmente il comportamento del

sistema operativo. Interessante notare 2.1 che, mentre il numero totale di permessi aumenta con ogni nuova release di Android, il numero di permessi con livelli normal e dangerous, che proteggono le funzionalità esposte alle applicazioni di terze parti, rimane abbastanza stabile. Quindi, dal punto di vista degli sviluppatori, il carico cognitivo non è aumentato molto in termini di nuovi permessi (anche se il numero di problemi di compatibilità da gestire continua a crescere a causa delle fluttuazioni nei permessi). Allo stesso tempo, i ricercatori di sicurezza e gli sviluppatori di app di sistema e piattaforme devono affrontare un numero sempre maggiore di permessi. Non solo vengono aggiunti nuovi permessi, ma durante l'evoluzione della piattaforma molti vengono rimossi o cambiano livello di protezione. La maggior parte dei permessi package sono rimossi perché i relativi pacchetti sono stati eliminati dal sistema o la loro funzionalità è diventata closed-source. Alcuni di essi sono diventati obsoleti perché la funzionalità corrispondente è stata resa disponibile a tutte le applicazioni o integrata con altre funzionalità."

Tutt'ora Android si sta impegnando su questo fronte per fornire ulteriori miglioramenti sul reparto autorizzazioni e permessi, con la finalità di garantire

- **Controllo** sui dati che l'utente decide di condividere con le applicazioni,
- **Trasparenza** sui dati che vengono richiesti e raccolti, per creare consapevolezza tra gli utenti e chiarire i motivi per cui le applicazioni richiedono dati specifici,

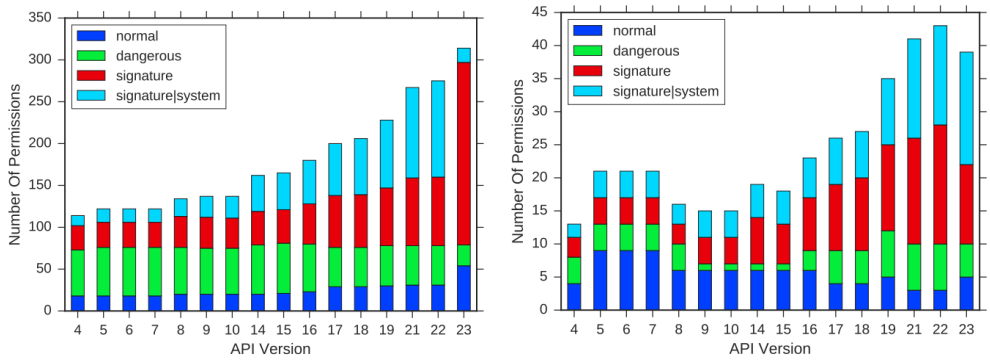
- **Minimizzazione dei dati** che significa ridurre al minimo il numero dei dati che un'applicazione deve utilizzare per funzionare.

2.2 Implementazione linee guida permessi e autorizzazioni nello sviluppo di un'applicazione Android

Tutti i miglioramenti fatti fino a questo momento relativamente alla questione dei permessi e delle autorizzazioni hanno causato un cambiamento anche per quello che riguarda le dinamiche di sviluppo di un'applicazione. Le linee guida di Android forniscono un buon punto di partenza per gli sviluppatori, in relazione alla corretta implementazione delle funzionalità di un'applicazione che richiedano permessi all'utente. La figura 2.2 mostra uno schema che riassume il flusso di lavoro da adottare nel processo di implementazione delle funzionalità di un'applicazione che si interfaccia con il sistema delle autorizzazioni di Android, mentre la figura 2.3 evidenzia la logica di definizione e implementazione dei permessi[1].

Nel momento in cui un'applicazione richiede delle autorizzazioni, esse devono essere dichiarate nel file `manifest` dell'applicazione. La specifica procedura di gestione delle autorizzazioni varia a seconda del tipo di autorizzazione, ed è ad esempio diversa per un'autorizzazione normale richiesta in fase di installazione rispetto a quanto accade per un autorizzazione richiesta a tempo di esecuzione (runtime).

[1] Informazioni aggiuntive si possono recuperare nella documentazione ufficiale Android



(a) (b)

Figura 2.1: Numero di permessi per ogni release della piattaforma: a) core permissions b) package permissions.

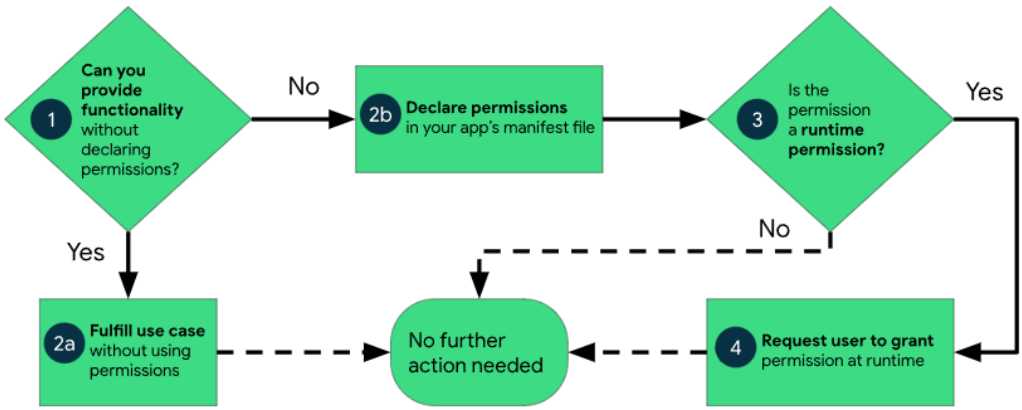


Figura 2.2: Flusso di lavoro ad alto livello per l'utilizzo delle autorizzazioni su Android

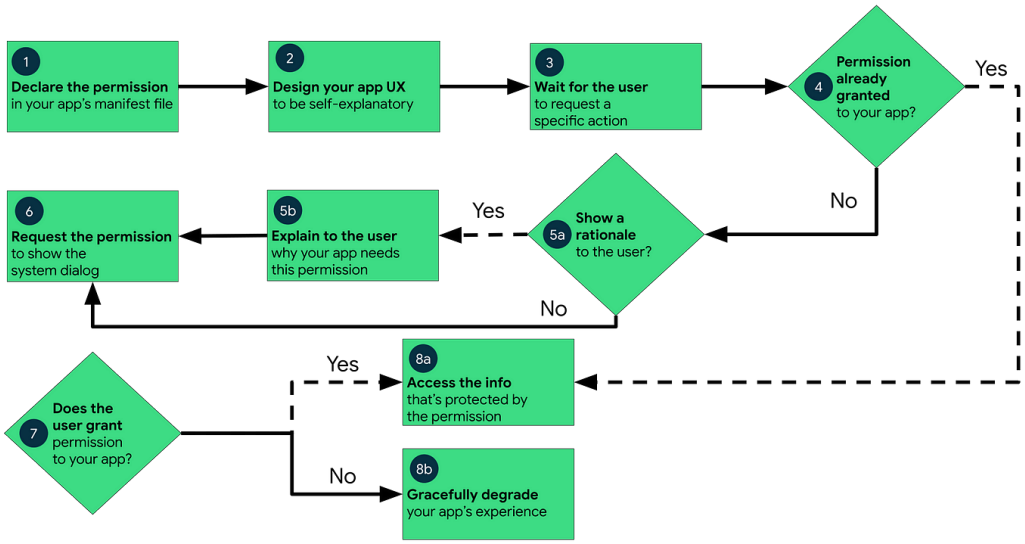


Figura 2.3: Flusso di lavoro a basso livello per la logica di definizione dei permessi Android

Capitolo 3

Telemetria in Android

Nel corso del tempo si sono sviluppati molteplici strumenti che aiutano gli sviluppatori a implementare un sistema di telemetria nelle loro applicazioni. Alcuni di questi strumenti forniscono servizi di telemetry generici mentre altri sono molto più specifici.

La grande disponibilità di questi strumenti permette agli sviluppatori di tracciare qualsiasi informazione ritengano importante con la finalità di migliorare l'applicazione che viene sviluppata, sia per quello che riguarda l'esperienza d'uso degli utenti, sia per quello che riguarda le funzionalità degli applicativi. Allo stesso tempo però, tutte queste possibilità rischiano di rendere difficile il processo di selezione degli strumenti più adatti, a seconda di quelli che sono gli obiettivi e le necessità di un'applicazione specifica. Per questo motivo, è bene capire quali sono questi strumenti e cosa ciascuno di essi offre.

3.1 Android Performance Tuner (APT)

Android Performance Tuner (APT) è uno strumento potente che Google mette a disposizione degli sviluppatori Android e che ha l'obiettivo di aiutarli nello

sviluppo delle applicazioni. Viene molto utilizzato per i giochi, poiché permette di misurare e ottimizzare le frequenze dei fotogrammi, oltre che la fedeltà grafica sugli schermi di dispositivi differenti e il tempo di caricamento, che può variare anche significativamente sulla base dell'hardware di questi dispositivi. Queste misurazioni, ottenibili da quasi tutti i tipi di dispositivi Android, permettono di rilevare problemi di prestazioni nell'esecuzione di un gioco (o anche di un'applicazione che fa particolare affidamento sui parametri elencati in precedenza) ed eventuali colli di bottiglia (bottleneck). In questo modo gli sviluppatori hanno uno strumento concreto per identificare problemi.

La possibilità di raccogliere dati sulle prestazioni da milioni di dispositivi è molto preziosa e spesso è quello che permette una risoluzione rapida dei problemi più gravi.

Con l'obiettivo di facilitare l'utilizzo dei dati raccolti da parte degli sviluppatori, Android Performance Tuner può essere integrato con alcuni dei servizi che Firebase [1] mette a disposizione, come ad esempio uno storage centralizzato che è in grado di raccogliere un gran numero di dati e di facilitare quindi la diagnostica dei problemi e contemporaneamente l'implementazione di miglioramenti mirati per risolvere specifici problemi e/o ottimizzare le risorse utilizzate dalle applicazioni per l'esecuzione delle loro funzionalità.

3.2 AppDynamics

AppDynamics è uno strumento importante di Application Performance Management (APM). Esso mette a disposizione degli sviluppatori una piattaforma

[1] Per maggiori informazioni a riguardo, si veda la sezione dedicata a Firebase riportata alla fine di questo capitolo.

avanzata per monitorare le prestazioni delle applicazioni, corredata di molteplici strumenti di telemetria specificatamente progettati per applicazioni Android. L'obiettivo è quello di permettere agli sviluppatori di mantenere uno standard di qualità e prestazioni sempre ottimo per le loro applicazioni.

Nello specifico, AppDynamics fornisce una panoramica dettagliata delle prestazioni in tempo reale ed è in grado di monitorare con grande accuratezza le risorse utilizzate dalle applicazioni, come ad esempio CPU e memoria. Oltre a questo, AppDynamics implementa un sofisticato strumento per la rilevazione automatica delle anomalie nei comportamenti delle applicazione nelle più svariate situazioni di utilizzo. In questo modo gli sviluppatori hanno modo di capire quali sono i problemi che impediscono il corretto utilizzo dell'applicazione e di apportare modifiche specifiche in tempi brevi, per assicurare un'esperienza utente ottimale.

Nel corso del tempo è stata aggiunta un'integrazione con altri strumenti di DevOps, con l'obiettivo di semplificare l'uso nelle situazioni più disparate.

3.3 Dynatrace

Dynatrace è un altro strumento disponibile in ambito telemetry. Fornisce servizi simili a quelli proposti da AppDynamics. Di fatto, è un diretto concorrente di quest'ultima.

Grazie ai recenti sviluppi in ambito intelligenza artificiale, la piattaforma ha deciso di integrare componenti AI all'interno del loro prodotto, per facilitare l'analisi dei dati raccolti, spesso in numero troppo elevato per permettere un'analisi manuale. Il motore di intelligenza artificiale Davis non si limita ad indagare i problemi e i malfunzionamenti, ma si sforza anche di pro-

porre soluzioni, riducendo in maniera significativa il tempo da impiegare per risolverli.

Uno dei punti di forza di Dynatrace è il suo focus relativamente alla sicurezza. Esso fornisce infatti degli strumenti in grado di scoprire vulnerabilità ed eventuali punti di attacco di un'applicazione in tempo reale. Questo fornisce agli sviluppatori un grande vantaggio perché permette loro di mantenere stabile un'applicazione nel corso delle versioni: non appena una minaccia o una vulnerabilità viene identificata, gli sviluppatori possono immediatamente abilitarsi per risolverla, prima che eventuali dati possano essere compromessi da attacchi malevoli e/o da errori che si verificano in maniera imprevista durante l'esecuzione dell'applicazione. In questo senso, Dynatrace fornisce anche degli strumenti di osservazione delle prestazioni che permettono di fare diagnostica e di raccogliere dati relativamente alle anomalie e ai crash degli applicativi.

Ulteriori servizi di tracking end-to-end sono messi a disposizione degli sviluppatori, con la finalità di tracciare in maniera completa le transazioni, dal front-end al back-end, tenendo traccia di tutti i componenti applicativi che stanno nel mezzo e fornendo così una panoramica completa delle interazioni degli utenti con l'applicazione e dell'applicazione con i servizi di back-end.

Per garantire l'efficienza delle applicazioni sotto tutti i punti di vista, Dynatrace implementa anche degli strumenti di monitoraggio delle infrastrutture che sono in grado di misurare e analizzare le prestazioni delle risorse di rete, oltre a quelle dei server e dei database che si interfacciano con le applicazioni.

Uno dei maggiori punti di forza di Dynatrace è la grande possibilità di personalizzazione: gli sviluppatori hanno la libertà di creare dashboard e finestre di analisi personalizzate, secondo i loro gusti e i loro bisogni, per meglio adattare i servizi a tipi di applicazioni specifiche.

Infine, Dynatrace può essere facilmente integrato in un'applicazione An-

droid grazie all'SDK che viene fornito e che può essere aggiunto al progetto come dipendenza in pochi semplici passaggi.

3.4 New Relic

New Relic APM è un'altra piattaforma di monitoraggio delle prestazioni, una delle più storiche e *"antiche"*, che analizza il comportamento di applicazioni web e non, orientato principalmente alle prime sopra citate. Il software New Relic APM ha un set di funzionalità destinato a raccogliere dati, analizzarli e segnalare metriche di prestazione rilevanti agli amministratori IT e ad altri stakeholder delle applicazioni. Più dettagliatamente, New Relic APM raccoglie metriche sulle transazioni web, tra cui il tempo di risposta lato server web, la velocità espressa in richieste al minuto e gli errori delle applicazioni nel tempo, oltre a metriche su singole richieste HTTP. Lo strumento analizza anche le metriche delle principali applicazioni di database, come SQL, per riportare i tempi di risposta e la velocità, il tempo per query, le query lente e altri dettagli che identificano le istruzioni SQL che potrebbero rallentare un'applicazione web. Supporta Java e ambienti esterni e può raccogliere metriche della macchina virtuale Java (come memoria heap e non-heap, garbage collection, conteggio delle classi, pool di thread, sessioni HTTP e transazioni). Gli amministratori possono definire strumentazioni personalizzate per le applicazioni Java, che consentono all'azienda di cercare attributi specifici delle prestazioni. Le metriche dei servizi esterni raccolgono dettagli sui servizi web; risorse cloud, inclusi servizi nativi come AWS, Azure e altro traffico di richieste basate su rete. New Relic fornisce analisi dettagliate delle vulnerabilità che identificano esattamente le posizioni degli errori e classificano le transazioni e i tipi di warning associati. Gli sviluppatori possono filtrare i risultati per estrarre dettagli

e attributi specifici per ogni traccia. Inoltre, un profiler di thread mostra le aree di attività relative dell'applicazione per individuare possibili colli di bottiglia da risolvere. Le capacità di tracciamento delle distribuzioni catturano istantanee delle prestazioni dell'applicazione prima e dopo per determinare gli effetti dei cambiamenti dell'applicazione o della configurazione.

Come funziona New Relic

New Relic raccoglie e memorizza metriche e dati dalle applicazioni web e altre applicazioni associate all'interno dell'azienda ricevendo i dati tramite la rete. Gli utenti visualizzano i dati tramite una dashboard.

Nello specifico, New Relic utilizza *agents*, che sono piccoli segmenti di codice installati nell'ambiente dell'applicazione. Numerosi agenti gestiscono una vasta gamma di applicazioni, e gli utenti possono trovare agenti per C SDK, Go, Java, Node.js, PHP, Python e altri ancora.

Mostriamo un esempio che illustra illustra come funziona New Relic mediante un'applicazione web . Un *agent* raccoglie una serie di dettagli sulle prestazioni, come il tempo necessario al codice per costruire ogni pagina web e il tempo di caricamento della pagina. Gli amministratori possono esaminare questi dettagli, identificare eventuali fattori che potrebbero ritardare le prestazioni e determinare la causa dei ritardi, mentre la dashboard segnala i problemi in tempo reale. Quando viene scoperto un problema, gli amministratori e gli sviluppatori possono intraprendere azioni correttive, come un aggiornamento del codice, e quindi confrontare le differenze di prestazioni per determinare oggettivamente l'effetto di eventuali modifiche. New Relic ha centinaia di integrazioni per interoperare con strumenti importanti in tutta l'azienda. Ad esempio può anche interagire con una serie di servizi cloud pubblici come AWS,

Google e Microsoft Azure. Le organizzazioni possono utilizzare le API per creare query personalizzate sui dati delle metriche e accedere ai dati su tassi di errore, prestazioni dei server delle applicazioni e altro. Un elenco completo delle integrazioni attuali è disponibile tramite il sito web di New Relic.

3.5 Sentry

Anche Sentry è un'applicazione molto valida per il tracciamento delle prestazioni, degli errori e dei crash di un'applicazione. Uno dei maggiori punti di forza di Sentry è la sua capacità di fornire log e stack trace completi, che aiutano gli sviluppatori a capire esattamente dove e perché si è verificata un'anomalia imprevista. Questi dati sono spesso accompagnati da descrizioni dettagliate relativamente alle attività degli utenti al momento della registrazione di un'anomalia o di un crash, e anche relativamente alle risorse utilizzate dal dispositivo in quel momento. A questo punto tutto il lavoro degli sviluppatori è dedicato alla risoluzione dei problemi e tutto il tempo di identificazione viene evitato.

Sentry implementa anche un sofisticato sistema di analisi che permette di aggregare errori simili in gruppi, con la finalità di facilitare l'identificazione dei problemi che ne derivano. Dei meccanismi di controllo possono essere impostati dagli sviluppatori, che vengono avvertiti nel momento in cui vengono rilevati nuovi errori. Inoltre, per facilitare il processo di sviluppo delle applicazioni, Sentry mette a disposizione una vasta gamma di integrazioni con altri strumenti e piattaforme di sviluppo, tra cui sistemi di gestione del codice sorgente (GitHub, GitLab).

L'integrazione dei servizi offerti da Sentry è semplificata grazie al suo SDK che può essere aggiunto al progetto.

3.6 Datadog

Datadog è un'ulteriore piattaforma in ambito telemetry che permette la raccolta di una vasta gamma di dati da un'applicazione Android. Grazie ai suoi strumenti, esso permette di monitorare con grande facilità le interazioni degli utenti con l'applicazione, con la finalità di fornire un feedback agli sviluppatori che possa aiutarli a indirizzare il loro lavoro nel migliore dei modi per garantire una esperienza d'uso ottimale.

Datadog offre anche un potente sistema di log che facilita la risoluzione dei problemi e che permette agli sviluppatori di dividere i tipi di log in gruppi usando dei tag.

Inoltre, è disponibile un servizio di controllo per il traffico dei dati, che è in grado di misurare il flusso di essi, specialmente in ambienti nativamente progettati per vivere in cloud. Come gli altri strumenti, Datadog offre un'ottima possibilità di personalizzazione.

Anche Datadog è in grado di tracciare in tempo reale e in maniera accurata l'utilizzo delle risorse dei dispositivi, come ad esempio CPU, memoria e I/O.

3.7 Dotcom-Monitor

Dotcom-Monitor è una piattaforma avanzata di monitoraggio delle prestazioni di siti web e di applicazioni web. In ogni caso, può essere utilizzata in ambito Android, con molteplici finalità:

- **Monitoraggio API**: per le applicazioni Android che utilizzano API esterne per fornire i loro servizi, Dotcom-Monitor mette a disposizione un sistema che permette di tenere traccia dello stato dei servizi di backend

su cui l'app si appoggia. In questo modo è possibile identificare eventuali problemi in tempo reale.

- **Test di Carico**: Dotcom-Monitor offre una funzionalità di test per le API che permette di mettere alla prova il carico che queste sono in grado di sopportare. Questo è fondamentale per assicurarsi che le applicazioni riescano a interfacciarsi con i servizi di backend in maniera appropriata anche nei momenti in cui il traffico raggiunge il suo picco.

- **Monitoraggio dello stato delle risorse**: la piattaforma prevede anche un sistema di gestione delle risorse utilizzate da un'applicazione, come ad esempio server e database, che è in grado di segnalare eventuali rallentamenti o anomalie.

- **Monitoraggio della User Experience**: Dotcom-monitor permette di simulare interazioni di utenti con l'interfaccia di un'applicazione Android, con la finalità di capire quali sono le cose che funzionano come previsto e per capire quali potrebbero essere eventuali problemi.

Grazie a tutte le sue funzionalità, Dotcom-monitor rientra nella categoria degli strumenti Application Performance Monitoring.

3.8 Instana

Instana è un'altra piattaforma per il monitoraggio delle prestazioni delle applicazioni (APM) che è progettata per dare agli sviluppatori una visione d'insieme relativamente alle applicazioni Android e all'infrastruttura su cui si basano. Come altri strumenti che abbiamo esplorato, anche Instana utilizza l'intelli-

genza artificiale per potenziare i suoi sistemi di rilevamento delle anomalie e degli errori.

Instana permette anche un'accurata analisi dei tempi di latenza, per capire quali sono i colli di bottiglia nelle iterazioni degli utenti con i vari servizi messi a disposizione dall'applicazione. La piattaforma è progettata per garantire un alto grado di personalizzazione ma soprattutto per essere scalabile, in maniera che si possa facilmente adattare a tutti i tipi di applicazioni, incluse quelle con un significativo utilizzo di risorse.

Le modalità con cui Instana permette di visualizzare le metriche relativamente alle applicazioni sono molteplici e la visualizzazione di sofisticati grafici 3D viene fornita grazie all'utilizzo di potenti algoritmi di Machine Learning.

Come gli altri strumenti, può essere integrato utilizzando il suo SDK.

3.9 Logic Monitor

LogicMonitor è una piattaforma di monitoraggio delle prestazioni IT basata su cloud che offre soluzioni per monitorare infrastrutture, applicazioni e servizi cloud. È progettata per fornire visibilità in tempo reale e approfondimenti sulla disponibilità di sistemi complessi. Come detto precedentemente, questo sistema è orientato più alle infrastrutture piuttosto che alle applicazioni web, infatti le sue funzionalità più sviluppate gravitano attorno alla analisi di server fisici e virtuali associati a sistemi operativi quasi Windows, Linux e Unix, oltre che alle analisi su router, switch, firewall e in generale tutti i dispostivi di rete che potrebbero dover affrontare delle vulnerabilità. Si concentra inoltre sulla protezione di applicazione e servizi web come database, web server, mail server e altre applicazioni aziendali tra cui i più importanti servizi Cloud come AWS, Azuree Google Cloud. L'interfaccia è estremamente professionale e per-

sonalizzabile, difatti risulta essere una delle dashboard meno *"user friendly"*. Inoltre, presenta delle feature molto interessanti legate alla notifica e personalizzazione di allarmi. LogicMonitor invia notifiche istantanee tramite e-mail, SMS, Slack e altre integrazioni quando vengono rilevati problemi,permettendo di configurare soglie di allarme ad hoc per diversi parametri e metriche.

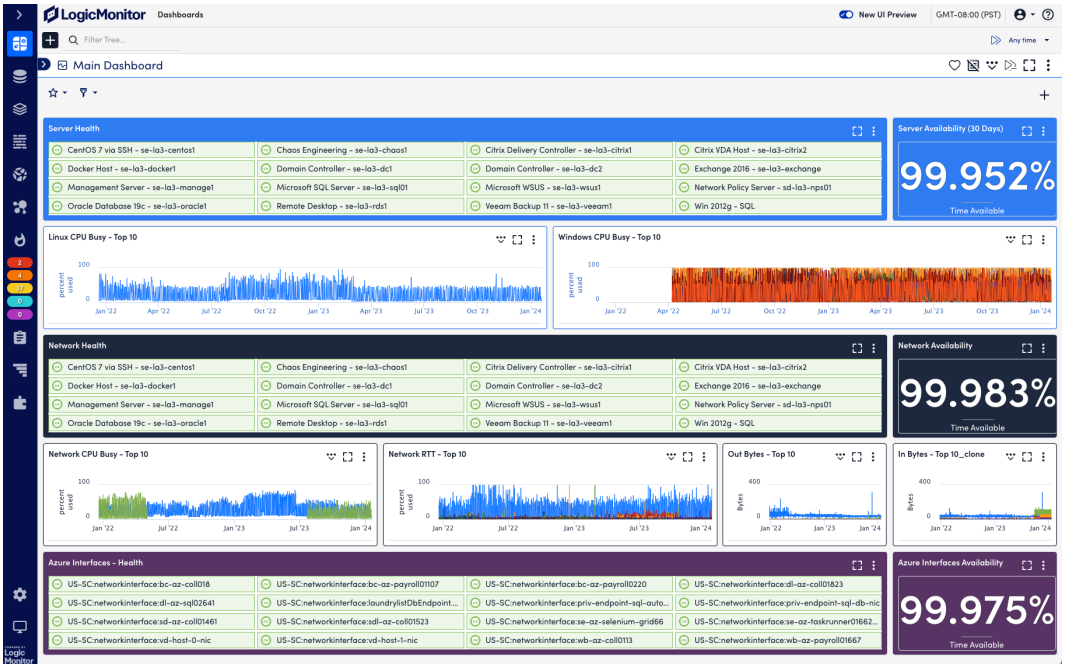


Figura 3.1: Dashboard di LogicMonitor

3.10 Raygun

Raygun è un'altra piattaforma che consente di monitorare le prestazioni di applicazioni Android in tempo reale. Anche questa piattaforma sta investendo molto sull'intelligenza artificiale, tanto da affermare che l'IA sarà il futuro del debugging per la risoluzione degli errori nello sviluppo software. Tra i servizi che offre agli sviluppatori ci sono

- **Tracciamento delle sessioni utente** che aiuta a capire in che modo gli utenti interagiscono con l'applicazione e quali sono i punti critici in cui vengono rilevati anomalie o errori che determinano crash.

- **Notifiche e alert**: Raygun è in grado di integrarsi con strumenti come Slack, Jira e Trello e di segnalare in tempo reale il rilevamento di errori critici, permettendo agli sviluppatori di intervenire tempestivamente e garantire un'esperienza utente ottimale.

- **Diagnostica degli errori**: ogni errore viene corredato di informazioni approfondite e include le condizioni del sistema nel momento in cui si è verificato, come l'utilizzo di CPU e memoria, oltre che ad altri parametri come ad esempio l'I/O, per dare agli sviluppatori una panoramica completa delle problematiche che impediscono il corretto funzionamento dell'applicazione.

Come altri strumenti che abbiamo analizzato, anche Raygun è di facile integrazione grazie agli SDK che vengono messi a disposizione degli sviluppatori.

3.11 LogRocket

Un altro strumento particolarmente interessante e noto è LogRocket. Come i precedenti, si tratta di un ambiente dedicato all'analisi delle prestazioni, monitoraggio di errori e correzione di bug. Rispetto alle altre piattaforme, resta una delle più curate a livello sia di servizi che estetico; non a caso fa parte della classifica dei 100 software migliori del 2024. Oltre alle classiche funzionalità di errori e avvisi, registrazione delle sessioni di rete con i relativi tempi di risposta e tutti gli altri vari servizi tipicio di un APM, LogRocket ha nel suo

armamentario una funzionalità unica e molto interessante: la registrazione delle sessioni. LogRocket registra ogni interazione dell'utente con l'applicazione, inclusi clic, movimenti del mouse, scorrimenti, selezioni di testo, e inserimenti di dati nei campi di input. Ciò avviene in tempo reale, consentendo di riprodurre esattamente ciò che l'utente ha fatto. Inoltre, la piattaforma mantiene una traccia dello stato del Document Object Model (DOM) in ogni momento, in modo che le riproduzioni riflettano accuratamente l'interfaccia utente vista dagli utenti durante le loro sessioni. Grazie al fatto che LogRocket registra lo stato del DOM in ogni momento, gli sviluppatori possono utilizzare queste informazioni per debuggare eventuali problemi che si verificano durante l'interazione dell'utente con l'applicazione. Ad esempio, se un elemento non si comporta come previsto, gli sviluppatori possono esaminare lo stato del DOM per comprendere la causa del problema. Le sessioni registrate permettono di rivedere le sessioni degli utenti in un formato video-like, il che è estremamente utile per comprendere il comportamento degli utenti e i percorsi che li hanno portati a incontrare problemi. Ciò non compromette comunque la privacy degli utenti, infatti LogRocket ha implementato una serie di strumenti che accompagnano tale funzionalità che permettono di oscurare tutte le informazioni private che un utente potrebbe condividere. In questo momento, LogRocket sembra essersi concentrata sull'integrazione dell'intelligenza artificiale nel proprio framework. Essa, difatti, risulta essere la tendenza del presente anche in questo ambito piuttosto specifico dello sviluppo Android. LogRocket, di fatto, è una delle piattaforme più competitive che il mercato possa offrire, ma allo stesso tempo risulta essere anche una delle più costose e impegnative dal punto di vista economico. D'altronde, la qualità si paga, e anche molto, visto che l'abbonamento mensile parte da 300 dollari.

3.12 Firebase

Firebase nasce come piattaforma di supporto per lo sviluppo, gestione e mantenimento di applicazioni sia Android che IOS e Web, creato e mantenuto da Google al momento attuale si tratta del framework più completo presente sul mercato data anche la continua aggiunta di funzionalità, l'implementazione di esso mette a disposizioni svariati strumenti per migliorare l'applicazione sotto molteplici punti di vista, tra questi strumenti troviamo utili ai fini telemetrici:

- Firebase RealtimeDB: si tratta, contrariamente al nome, di un dizionario annidato in formato `[chiave:valore]` che permette di salvare facilmente intere classi di dati associando ad esse una chiave che sia in grado di identificarle in maniera univoca, permettendo l'inserimento e la creazione di uno storico dei vari contenuti locali dell'applicazione.

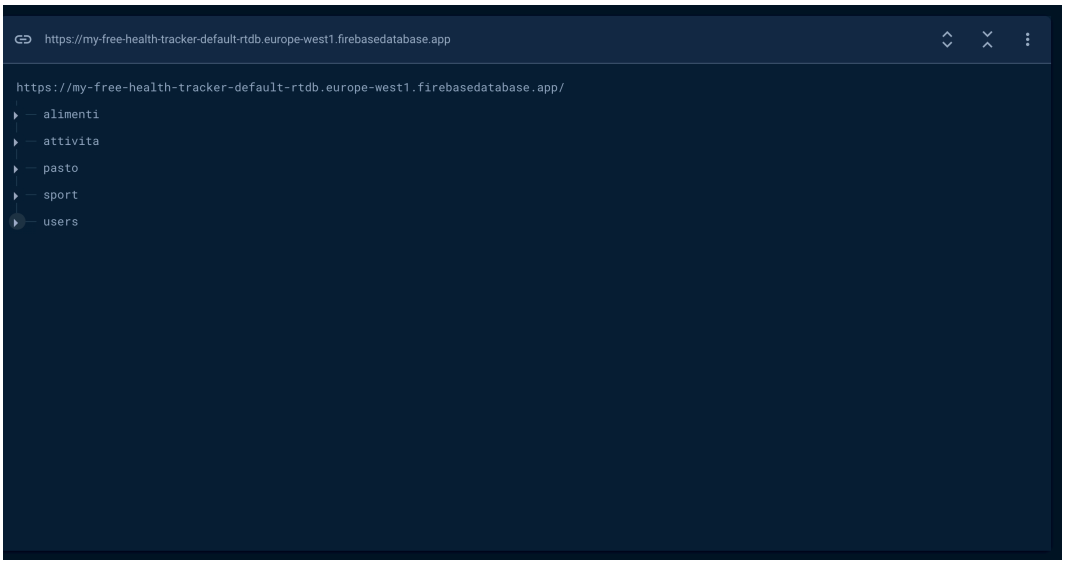


Figura 3.2: Visuale generale di RealtimeDB

Si implementa facilmente una volta dichiarata l'istanza del database su Firebase tramite i comandi `dbRef.child("Chiave").setValue("Valore)`.

Inoltre permette anche di recuperare velocemente ed in maniera asincrona i dati da remoto sia per fini utili all'applicativo utente sia per fini di analisi di essi con strumenti esterni, infatti nonostante RealtimeDB non possa essere definito direttamente uno strumento di telemetria data la sua natura di utilizzo più generica la vera potenzialità deriva dall'analisi ed elaborazione incrociata che si può fare dei dati utente anche grazie alla natura alquanto evasiva di Firebase sui permessi richiesti per connettersi alla rete (ossia nessuno). Da menzionare anche Firebase Data Connect di prossima implementazione (al momento della scrittura in Early Access) il quale è un vero e proprio database basato su PostgreSQL.

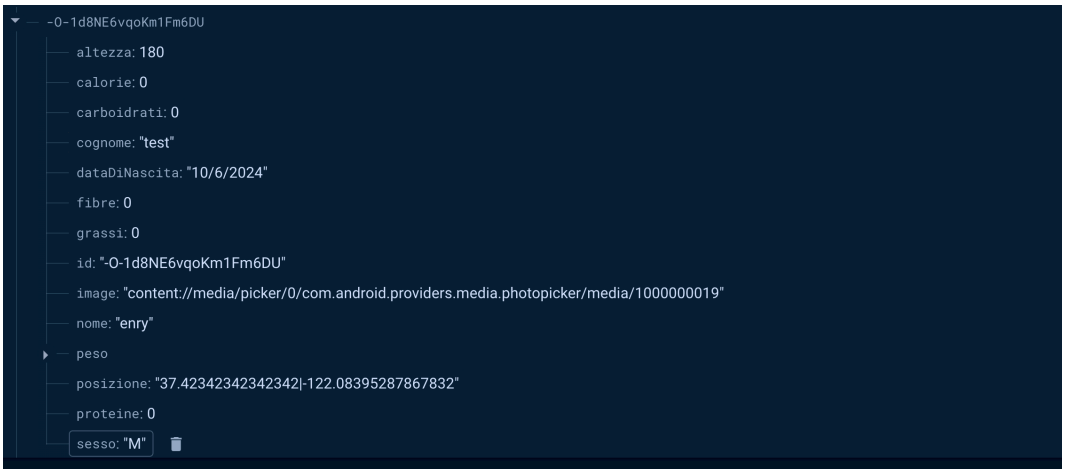


Figura 3.3: Contenuto in tempo reale di RealtimeDB

- Firebase Storage e FireStore: sono molto simili a delle cartelle remote su cui è possibile caricare file anche di grandi dimensioni, utili al fine di salvare i contenuti su un supporto di archiviazione esterno al dispositivo oppure ad analizzare i contenuti che l'utente abilita o crea in app (principalmente Foto e Video), il quale apre chiaramente alla possibilità di un uso illecito del materiale acquisito dall'applicazione a scopi malevoli.

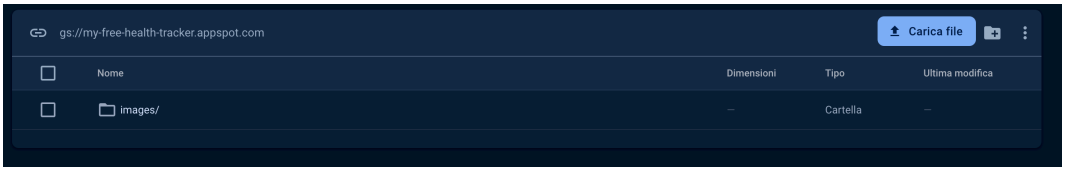


Figura 3.4: Contenuto FirebaseStorage

L'utilizzo risulta molto semplice, è sufficiente richiamare: `FirebaseStorage.getInstance().reference.child(percorso)` per poi inserire il file o il flusso dati con `putFile/Stream(uri)`; codeste funzionalità richiedono preventivamente accesso tramite Firebase Auth oppure tramite l'accesso anonimo se l'autenticazione non è un fattore fondamentale o l'utente deve essere ignaro del collegamento alla rete.

- Firebase CrashAnalytics: Funzionalità fondamentale per lo sviluppo di una applicazione solida e priva di errori, di facile implementazione tramite l'SDK fornita data Google, provvede in caso di arresto anomalo dell'applicazione un resoconto dell'intera istanza attraverso l'errore sollevato, la versione dell'applicazione ed i meta-dati del dispositivo (sistema operativo, hardware). Inoltre la console di comando integra un sommario ed una media sulle anomalie riscontrate più spesso, implementando anche una funzione di avviso in tempo reale in caso di situazioni anomale per consentire un pronto intervento.

- Firebase Performance Monitor: fornisce informazioni su varie operazioni dispendiose a livello di tempo compiute dall'applicazione, è possibile configurarlo su una vasta gamma di situazioni (tempo d'avvio, tempo di connessione alla rete,ecc) e fornisce indicazioni riguardo a che sezione dell'applicazione richiede più risorse o lasciano l'utente più tempo in attesa, situazione da ridurre assolutamente al minimo al fine di aumen-

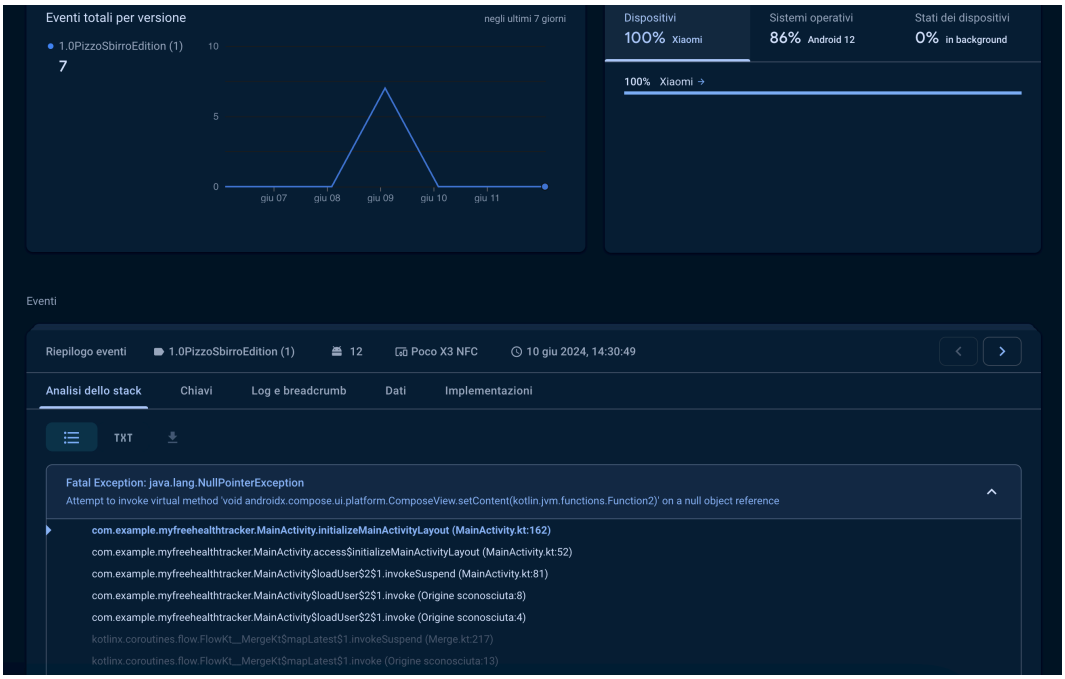


Figura 3.5: Dettaglio di un errore in CrashAnalytics

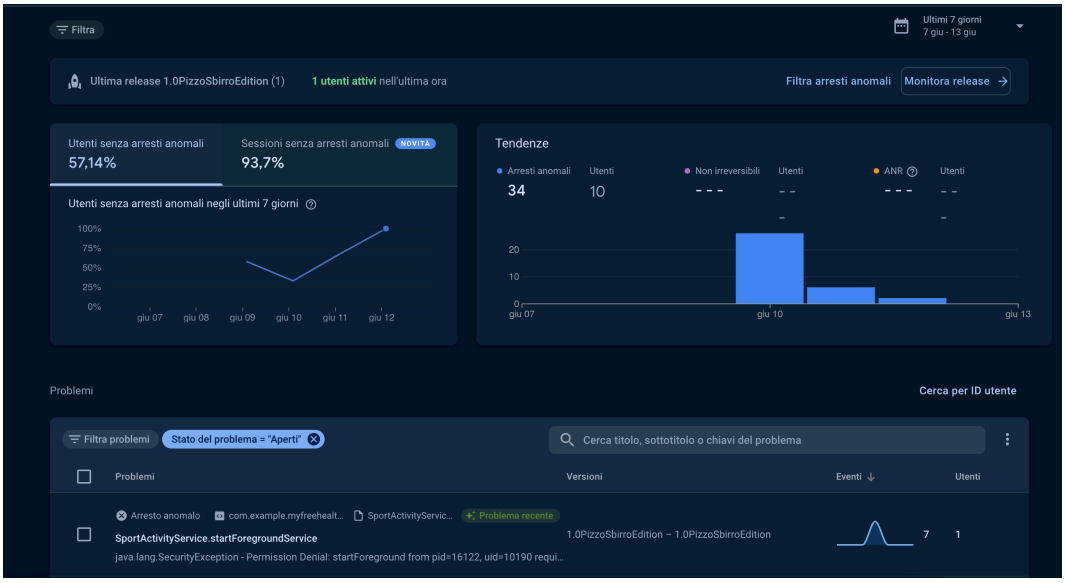


Figura 3.6: Panoramica generale di CrashAnalytics

tare il tempo che l'utente spende sull'applicazione, nonché migliorare l'esperienza d'uso.

Figura 3.7: Panoramica di Performance Monitor

Figura 3.8: Dettagli del tempo di caricamento di ogni sezione dell'applicazione

- Firebase Analytics: Integra un'ampia gamma di strumenti per il monitoraggio dell'applicazione rilasciata, il funzionamento principale si ba-

Richieste di rete | Tracce personalizzate | Rendering schermo | Ultimo aggiornamento: meno di un minuto fa

Cerca in tutti i domini e in tutti gli URL | Tutte le richieste di rete | Crea un pattern URL personalizzato

Tracce richiesta di rete	Risposta	Cambiamento 7 giorni	Operazione riuscita	Cambiamento 7 giorni	Azioni
world.openfoodfacts.org/** Numero di esempi: 13	754 ms	+60%	100%	+0	
images.openfoodfacts.org/** Numero di esempi: 10	573 ms	+55%	100%	+0	

Figura 3.9: Dettaglio del tempo di risposta delle chiamata via Internet

sa sull'acquisizione di eventi personalizzati o meno che Firebase invia al suo sistema di monitoraggio permettendo per ogni utente di tenere traccia dei suoi progressi nell'applicazione, per esempio l'inserimento o acquisto di un prodotto, il cambio di schermata oppure ancora l'avanzamento tra i livelli di un videogioco. Si tratta di una funzione fon-

Eventi esistenti

Nome dell'evento	Conteggio	Variazione %	Utenti	Variazione %	Contrassegna come evento chiave
app_clear_data	69	↑ 6.800,0%	17	↑ 1.600,0%	
app_exception	31	-	5	-	
app_remove	55	↑ 5.400,0%	55	↑ 5.400,0%	
first_open	38	↑ 322,2%	38	↑ 322,2%	
hasCompleteRegistration	13	-	6	-	
hasInsertedActivity	7	-	4	-	
hasInsertedFood	16	-	4	-	
hasInsertedMeal	8	-	3	-	
hasInsertedSport	7	-	4	-	
hasUpdateCalorie	1	-	1	-	
hasUpdatePeso	7	-	2	-	
hasUpdatePosition	66	-	2	-	
login	431	-	8	-	
screen_view	2.945	↑ 607,9%	62	↑ 588,9%	
session_start	304	↑ 328,2%	62	↑ 588,9%	

Figura 3.10: Eventi analizzati

damentale al fine dello sviluppo continuativo dell'applicazione poiché permette di comprendere i punti debole della catena interattiva dell'applicazione, ossia dove l'utente tende a perdere interesse ed uscire dall'applicazione, oltre che allo sviluppo di nuove funzionalità poten-

do comprendere dove concentrare gli sforzi. L'implementazione è molto veloce, oltre all'introduzione delle dipendenze e l'ottenimento dell'istanza tramite il singleton fornito da Firebase è sufficiente richiamare `firebaseAnalyticsRef.logEvent(nomeEvento, bundle)` per registrare un evento.

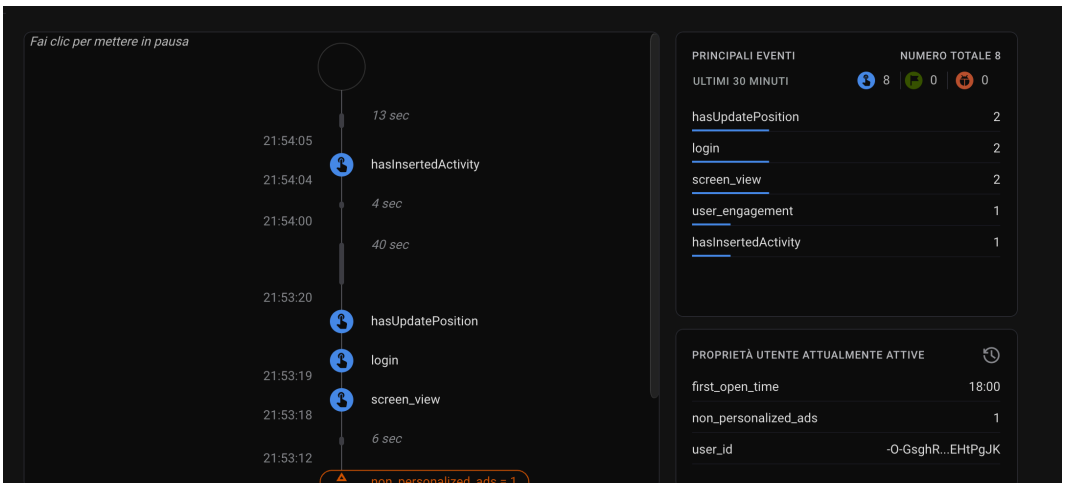


Figura 3.11: Esempio di eventi scatenati nell'uso dell'applicazione

- ReleaseMonitor Potente strumento per il monitoraggio dell'adozione delle varie versioni dell'applicazione, integra anche un veloce resoconto della stabilità di esse, include inoltre come CrashAnalytics i dettagli sulla tipologia di dispositivi su cui sono stati riscontrati errori insieme alla quantità di Ram e Memoria interna disponibile.

- Firebase Console Altro elemento importante al fine della gestione generale di tutte le funzionalità di Firebase è la console di comando che offre una panoramica sullo stato dell'applicazione ed il controllo dei vari strumenti non soltanto utili ai fini telemetrici ma dell'intero processo produttivo dell'applicazione con altre funzioni quali: App Check, App Hosting, Authentication, Machine Learning, A/B Testing, App Distri-

Figura 3.12: Confronto di due versioni tramite Release Monitor

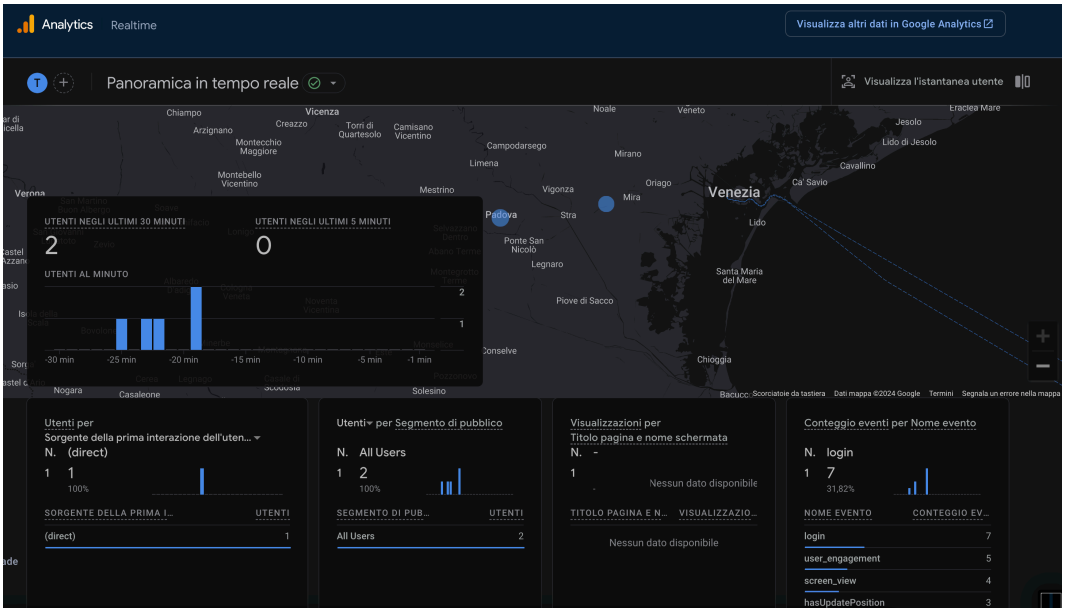


Figura 3.13: Altra panoramica generale dell'applicazione

bution, Test Lab e molte altre che saranno presto aggiunte (come il già' citato database Data Connect).

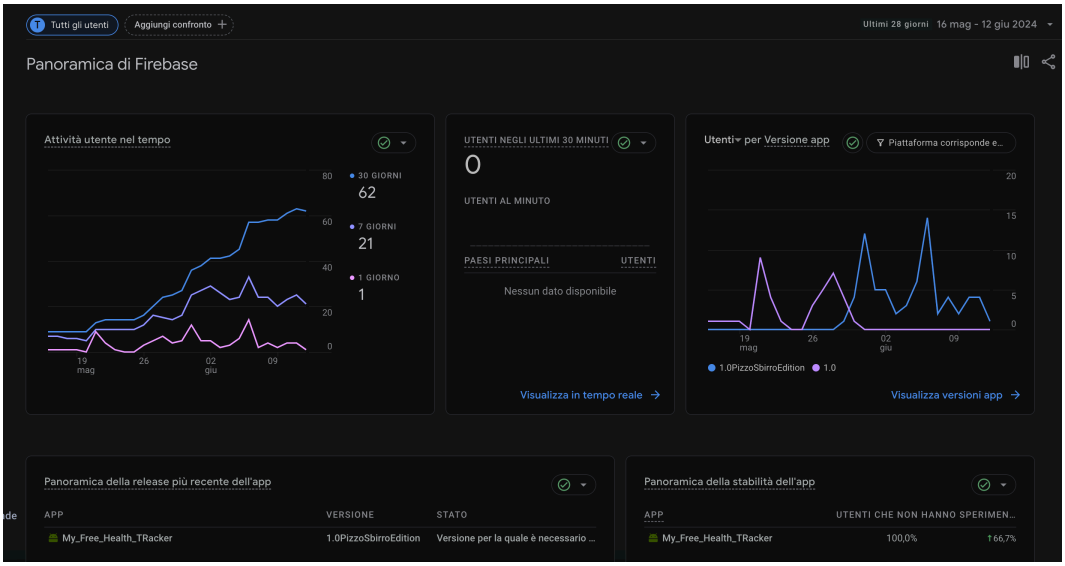


Figura 3.14: Panoramica generale dell'applicazione

Un altro importante punto di forza di Firebase si tratta della semplicità con la quale si può integrare nell'applicazione, nonché la sua robustezza data dalle limitate ma chiare funzioni che rende disponibili allo sviluppatore mentre la controparte più complessa di gestione della concorrenza, funzioni asincrone, connessioni alla rete, ridondanza d'esecuzione e gestione delle funzionalità' in caso di mancanza di requisti (per esempio mancanza di connessione di rete) è completamente nascosta, infatti l'unico punto di accesso è dato dai vari `getInstance()` i quali forniscono un'istanza singleton e mettono a disposizione solo le funzioni necessarie all'uso del particolare strumento.

Importante specificare che tutte le funzioni di Firebase, probabilmente dato il coinvolgimento di Google, non richiedono alcun permesso nonostante forniscano in maniera, a seconda della funzione, accurata o meno la posizione, accesso alla rete, accesso alla memoria interna dell'applicazione ed altro an-

cora. La vera potenza di Firebase sta appunto la capacità di ottenere dati in enorme quantità in maniera assolutamente trasparente all'utente. La vera utilità di questi dati non è nel materiale grezzo e puro che essi forniscono bensì nell'elaborazione di essi, che si ottiene attraverso un'accurata intersezione e analisi; facciamo un esempio con le informazioni fornite dalla nostra applicazione. Grazie a tutti i dati memorizzati sul RealTime Database è possibile incrociare le abitudini dell'utente a fini commerciali, per esempio vedere la tipologia di prodotti consumati a pranzo da utenti di sesso maschile con fascia di età fra i 20 e 30 anni e residenti in Veneto oppure l'analisi del tempo di utilizzo medio per migliorare un algoritmo che stimoli l'utente a passare più tempo sull'applicazione (Instragram ne è un chiaro esempio).

Capitolo 4

La nostra applicazione, (Un)Health

Con la finalità di portare un esempio di implementazione del concetto di Telemetry, abbiamo sviluppato un'applicazione Android che abbiamo sarcasticamente chiamato *"(Un)Health"*, compatibile con SDK $\geq$ 24, ma per una migliore esperienza consigliata per API Level 34 (TargetAPI Level). Diciamo sarcasticamente (Un)Health non perché i nostri fini siano particolarmente malevoli, ma piuttosto per estremizzare il concetto per cui l'applicazione è sviluppata, ovvero la raccolta di dati di utilizzo e dell'utente, oltre che al monitoraggio della stabilità, la performance e l'efficienza del servizio che il nostro software offre. L'aspetto che rende pericolosa una applicazione non è tanto la quantità di dati che vengono raccolti, ma l'utilizzo che se ne fa e la protezione legata ad essi. La nostra applicazione fornisce una dimostrazione applicativa di ciò che potrebbe essere realizzato effettivamente in prodotti commerciali, e ciò ci ha fatto acquisire notevole consapevolezza soprattutto sulla quantità di informazioni che vengono rilevate che, proporzionate alle dimensioni della nostra applicazione, posso essere veramente colossali, essendo comunque il nostro

un esempio circoscritto e non paragonabile ad altri sistemi offerti da grandi aziende.

Per implementare queste funzionalità abbiamo fatto affidamento su strumenti comunemente utilizzati nell'ambito dello sviluppo di applicazioni Android, con l'obiettivo di capire come questi aspetti vengono affrontati nella realtà.

Tuttavia i dati sono molto più di valore quanto più sono sensibili all'utente (abitudini, problemi, ecc) e di conseguenza, cosa c'è di meglio se è proprio l'utente di sua spontanea volontà a fornirceli?

4.1 Idea generale

La nostra applicazione prende ispirazione dalle molteplici già disponibili nell'ambito della salute e del fitness. L'idea di base è quella di fornire all'utente uno strumento di supporto che possa in qualche modo aiutarlo a tenere traccia dei suoi progressi, relativamente alla dieta e allo sport. L'obiettivo è quello di facilitare la visualizzazione di statistiche riguardo la dieta e lo sport, e di visualizzarle in una maniera chiara e diretta con dei grafici opportuni.

La motivazione di questa scelta è data dalla grande quantità di dati che è possibile ottenere da questo tipo di applicazioni. D'altra parte, per permettere di visualizzare statistiche di supporto, è necessario che l'utente fornisca dati relativamente alle sue abitudini alimentari e sportive.

Una premessa da precisare è il fatto che la nostra applicazione probabilmente raccoglie più dati di quelli che un utente si aspetta, anche non strettamente necessari per migliorare l'esperienza utente. In ogni caso, abbiamo cercato di rispettare le linee guida Android relativamente alle autorizzazioni, per ottenere dati come foto e informazioni sulla localizzazione.

Ancora una volta, la finalità è dimostrare che spesso gli utenti autorizzano alcune applicazioni ad accedere a determinate risorse, ma nella pratica non sanno veramente quante siano le informazioni che vengono condivise.

Nelle sezioni successive verranno esposti dettagli di tipo implementativo e di progetto dell'applicazione, corredati di immagini che facilitano la spiegazione.

4.2 Organizzazione delle interazioni utente

L'applicazione è organizzata in due Activity. La prima, `LoginActivity`, si occupa di gestire il processo di registrazione di un utente, richiesto per utilizzare l'applicazione, mentre la seconda, `MainActivity`, ha il compito di ospitare tutte le schermate dell'applicazione, organizzate in Fragment, e contemporaneamente di implementare le barre di navigazione.

L'applicazione consiste in tre schermate principali, che denotano le tre sezioni:

- **Aggiornamenti**: che ha l'obiettivo di visualizzare informazioni di carattere generale, sulla base di dati specifici inseriti in altre sezioni. L'idea è quella di fornire all'utente una panoramica riassuntiva generale ogni volta che l'applicazione viene avviata, prima che l'utente decida di spostarsi sulle rimanenti sezioni per utilizzare gli altri servizi.

- **Sport**: dove l'utente può inserire informazioni dettagliate sulle attività che svolge in ambito sport.

- **Pasti**: dove l'utente può inserire informazioni dettagliate relativamente ai pasti della giornata.

Ciascuna sezione ha un Fragment principale associato ed eventuali Fragment ulteriori per meglio separare le funzionalità legate alla stessa sezione.

L'applicazione è dotata di una barra superiore (toolbar) e di una barra inferiore (bottom navigation bar). Quest'ultima permette all'utente di navigare liberamente da una sezione all'altra. L'implementazione di questi componenti utilizza un approccio dichiarativo (XML) e la navigazione tra i fragment è gestita da un navigation graph, con `layout_main.xml` come nav host. In questo modo lo spostamento da un fragment all'altro è regolato dal nav controller. All'interno di ciascun Fragment sono inseriti i componenti necessari a fornire le funzionalità della sezione cui è associato.

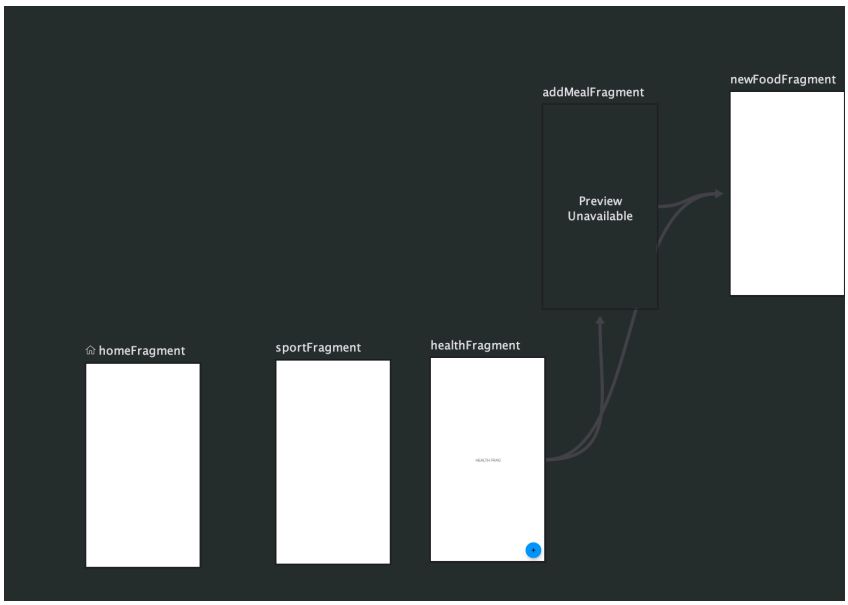


Figura 4.1: NavHost Fragment

Sebbene, come apparirà chiaro in seguito, abbiamo alla fine deciso di implementare gran parte dell'interfaccia grafica usando Jetpack Compose, abbiamo deciso di mantenere il fondamento del layout in XML, per sfruttare la navigazione con il navigation graph, attualmente un po' più semplice di quella disponibile in Compose. Tra le altre cose, volevamo evitare di ricostruire quella

parte e in più volevamo esplorare entrambi i mondi, quello dichiarativo (XML) e quello programmatico (Compose), per capire quali fossero i punti di forza (e i punti deboli) di ciascuno.

Con la finalità di rendere quanto più modulare possibile l'applicazione, abbiamo utilizzato Design Pattern (es. Singleton) che verranno esplorati nel seguito.

Per gestire i dati dell'utente, abbiamo implementato un database locale con Room al quale è possibile accedere dal relativo ViewModel. Questo database contiene informazioni relativamente all'utente ed a tutti i dati che ha inserito nelle varie sezioni. Sebbene questi dati siano raccolti in gran numero, in questo punto dell'applicazione sono comunque dati locali nel dispositivo e nonostante l'utilizzo del RealTime DB di Firebase l'applicazione nasce internamente come disconnessa dalla rete (fatta eccezione per una parte di elementi con il semplice scopo di popolare i menu) e l'utilizzo di quest'ultimo comporta svariati problemi in caso di mancanza di connessione nonché tempi di caricamento di ordini di grandezza superiore rispetto a quelli che si osservano con un database interno.

La vera raccolta dei dati, che vengono poi inviati al cloud, viene fatta grazie a Firebase e al suo servizio Realtime Database che, data la struttura a dizionario, presenta delle entità primarie: Alimenti, Attività, Pasto, Sport e Users, che al loro interno (nel valore) hanno i vari dati inseriti dall'utente, ognuno di essi differenziato da un id univoco. Nel database locale è presente in ogni riga delle tabelle un elemento ridondante che è il codice utente (ottenuto in maniera univoca da Firebase tramite apposito comando) in modo da poter facilmente tenere traccia delle azioni eseguite dall'utente all'interno del database globale.

Per migliorare alcune delle funzionalità che avevamo in mente all'inizio

abbiamo utilizzato delle risorse esterne. Tra queste, librerie per la creazione di grafici, utili per generare una panoramica dei dati inseriti dall'utente, e un'API che fornisce informazioni nutrizionali di alimenti, sulla base del loro codice a barre (EAN-8, EAN-13), oltre ad un service per tenere traccia dello svolgimento dell'attività. Un approfondimento relativamente a questo aspetto viene fatto nel seguito.

Il processo di Registrazione

Il processo di registrazione è il primo passo che l'utente deve affrontare per poter utilizzare l'applicazione. Già in questa fase vengono richieste molteplici informazioni. Le figure 4.2 e 4.3 mostrano le schermate di inserimento dei dati.

Tutte le schermate vengono gestite dalla `LoginActivity` e sono realizzate in Jetpack Compose come funzioni della classe `AuthPage`, che ha il compito di gestire tutto il layout.

I campi di inserimento sono corredati di funzioni di controllo che impediscono all'utente di inserire dati non validi. Nella pratica, l'utente non può avanzare nel processo di registrazione fino a che non inserisce dati validi per i campi della schermata corrente. Per i campi di testo, ci sono vincoli di riempimento, vale a dire che non è possibile lasciare campi vuoti. Inoltre, per i campi di testo che accettano solo valori numerici, non è possibile inserire testo. I valori non interi si rappresentano usando il punto. Per la scelta del genere e della data di nascita vengono utilizzati un radio menu e un date picker, rispettivamente. Naturalmente un utente non può selezionare una data di nascita futura.

L'inserimento della foto è l'ultimo passaggio della registrazione ed è completamente opzionale ma in caso venisse eseguito l'immagine verrà inviata al

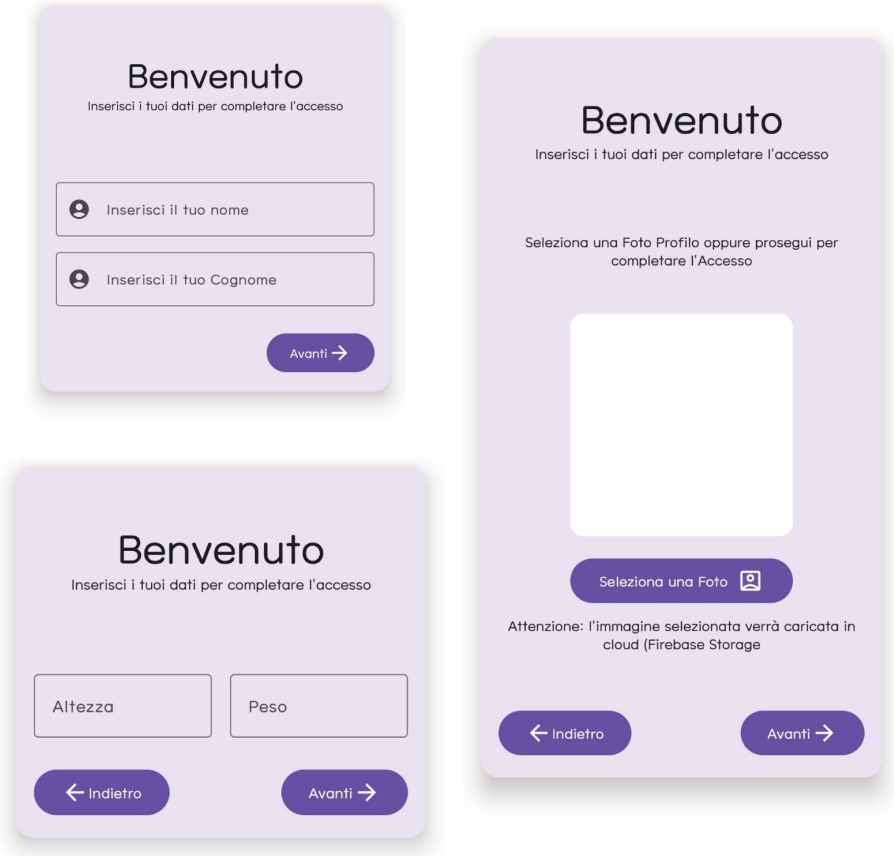


Figura 4.2: Processo di registrazione: inserimento nome, cognome, peso, altezza e foto (opzionale).

Firebase Storage (come riportato ai fini di privacy per la valutazione dell'app). Può essere evitato e in quel caso l'icona dell'utente sarà un'icona predefinita. Una volta che la registrazione viene completata, l'utente viene notificato dell'avvenuto completamento ed è reindirizzato sulla pagina principale dell'applicazione, gestita dalla `MainActivity`, dalla quale può iniziare a usare i servizi forniti e a navigare tra le sezioni.

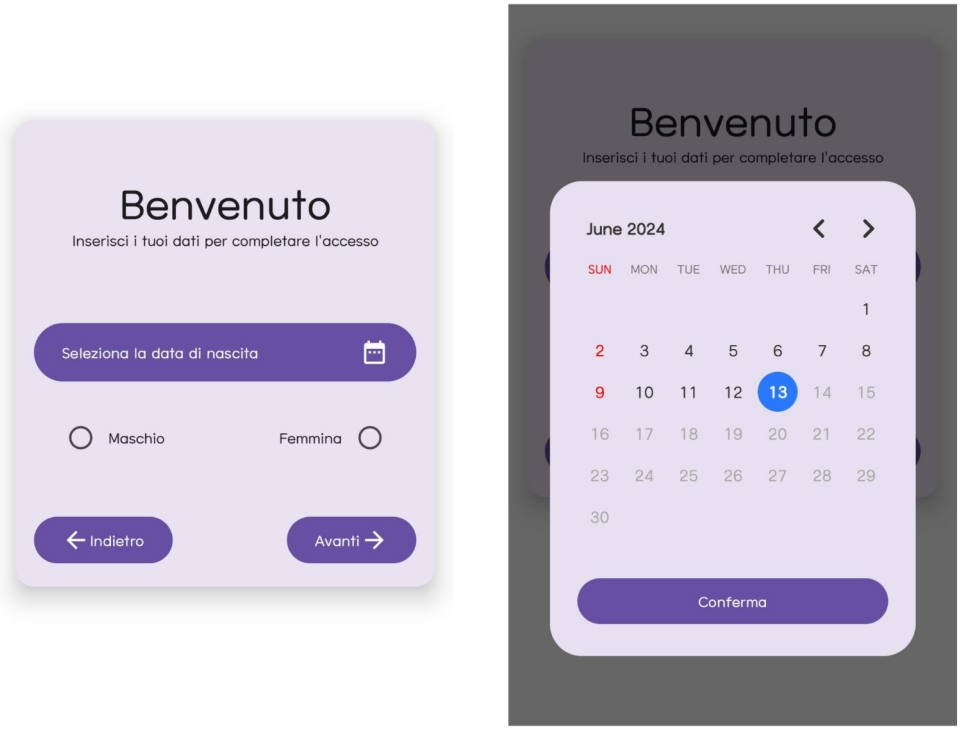


Figura 4.3: Processo di registrazione: inserimento data di nascita e genere.

La sezione Aggiornamenti

La sezione aggiornamenti fornisce una panoramica generale delle informazioni che l'utente ha inserito nelle varie sezioni. Abbiamo voluto concentrare le informazioni relativamente all'alimentazione, piuttosto che quelle relative allo sport, poiché sono molte di più. La figura 4.4 mostra l'interfaccia della sezione aggiornamenti, che è divisa in molteplici sezioni.

La prima sezione fornisce una panoramica generale relativamente alle calorie giornaliere totali accumulate con i vari pasti e alle informazioni sui macro nutrienti che caratterizzano gli alimenti consumati nel giorno corrente. Inoltre, viene mostrato un grafico a barre che visualizza la classifica degli alimenti più consumati dall'utente.

Figura 4.4: Sezione degli aggiornamenti

Subito sotto il riepilogo giornaliero viene visualizzato un grafico che rappresenta il consumo dei macro nutrienti negli ultimi cinque giorni. Ogni linea è associata a ciascun macro nutriente.

Infine, l'ultimo grafico visualizza la traccia degli ultimi sette valori misurati per il peso. Nella stessa sezione compare un bottone che permette di aggiornare il valore del peso, che verrà automaticamente associato alla data in cui viene inserito. Il grafico che lo traccia viene automaticamente aggiornato e in questo modo l'utente può vedere l'andamento del peso nel corso del tempo.

La sezione Sport

La sezione sport consiste nella rappresentazione di un grafico che descrive l'andamento delle calorie totali bruciate negli ultimi tre giorni. Al di sotto, troviamo la lista di tutte le attività inserite dall'utente, di cui viene specificata la durata, la lunghezza e le calorie bruciate. Rispetto alla sezione dedicata ai pasti, che viene approfondita successivamente, è risultato più difficile estrarre ulteriori informazioni dall'utente, in quanto non c'è la possibilità di interagire ad esempio con dispositivi wearable come gli smartwatch, dai quali è possibile ottenere molte più informazioni riguardanti la sessione di allenamento (battiti cardiaci, VO2...) che purtroppo non possono essere calcolate semplicemente con uno smartphone. La sessione di allenamento può essere aggiunta mediante la componente `AlertDialog` di Jetpack Compose, in cui è possibile inserire il tipo di attività e le varie informazioni associate ad essa. Inoltre, si può avviare un allenamento il quale riporterà automaticamente il calcolo della durata attraverso un apposito `Button`: al click, viene eseguito un Service che rimane in esecuzione per tutta la durata della sessione: l'`AlertDialog` viene bloccato a schermo finché non viene fermato il cronometro; successivamente l'attività può essere completata e inserita; oltre al tempo viene inserita in automatico anche la data.

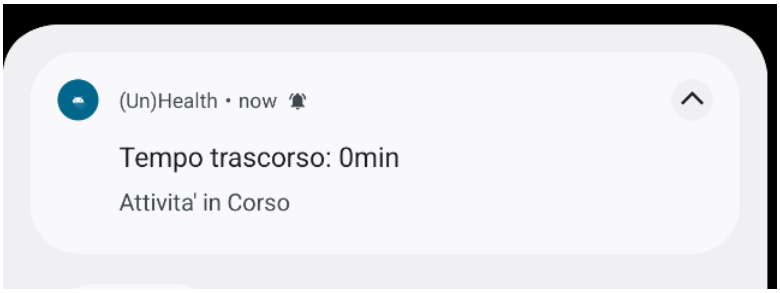


Figura 4.5: Notifica di attività' avviata che si aggiorna ogni minuto

La sezione Pasti

La sezione pasti è la sezione più complessa dell'intera applicazione, poiché gestisce funzionalità multiple:

- **Aggiunta dei nuovi alimenti al database**: l'utente ha la possibilità di aggiungere un nuovo alimento che potrà usare in seguito per comporre i suoi pasti. Nel processo di aggiunta può appoggiarsi al sistema di scansione di un codice a barre, che interroga l'API di FoodOpenFacts e tenta di ricavare i valori nutrizionali associati al codice. Alternativamente, può inserire i dati manualmente, in modo che anche nel caso di fallimento della chiamata API in caso di prodotto non presente nel database (non essendo una repository completa), l'inserimento possa essere effettuato.

- **Aggiunta di un nuovo pasto**: l'utente può creare un nuovo pasto aggiungendo uno o più alimenti tra la lista già presenti in locale, oppure può creare un nuovo alimento, che poi verrà anche aggiunto alla lista degli altri alimenti locali. Alternativamente, può scegliere di inserire un alimento utilizzato in precedenza da un altro utente, disponibile grazie alla raccolta dei dati sul Realtime Database centralizzato di Firebase.

Inoltre, la schermata principale relativa ai pasti mostra un riepilogo dei pasti della giornata corrente. Per fornire all'utente maggiore libertà nella consultazione delle informazioni, una lista di giorni in cima alla schermata gli permette di visualizzare le informazioni relative ai pasti dei giorni precedenti, se queste sono disponibili.

Ogni pasto visualizzato nella lista che popola questa schermata può essere cliccato nell'apposita icona a destra, che causa un'espansione che visualizza le informazioni degli alimenti che lo costituiscono.

4.3 UI Design

In questa sezione verranno elencate brevemente le scelte implementative per quanto riguarda la gestione dell'interazione tra le varie componenti che costituiscono l'interfaccia grafica e l'estetica della nostra applicazione.

Architettura

La componente applicativa risiede in un'unica attività: `MainActivity`. Inizialmente, durante la prima metà del corso, l'applicazione gestiva l'interazione di tre Fragment mediante un `Navigation Graph`, e l'idea era di sviluppare interamente l'applicazione attraverso l'approccio dichiarativo. Facendo alcuni test individualmente, però, ci siamo resi conto che l'utilizzo di file XML ci limitava nella personalizzazione delle entità grafiche, perciò è stata presa la seguente scelta: la navigazione fra i vari Fragment sarebbe stata organizzata con approccio dichiarativo, mentre le componenti esposte nei vari Fragment sarebbero state realizzate in Compose. Questa scelta è stata anche favorita dalla maggior intuitività del grafo di navigazione delle risorse gestito tramite file XML rispetto al corrispondente elemento in Compose. Tutte le componenti che costituiscono la UI sono implementate attraverso approccio programmatico, mediante funzioni annotate con l'annotazione `@Composable`. All'interno delle stesse funzioni viene gestita anche la computazione dei dati che devono essere visualizzati nei vari elementi. Inoltre Compose risulta molto comodo per visualizzare quantità indefinite di dati su varie liste estremamente diverse fra loro.

Design UI

Per quanto riguarda l'approccio al design ci siamo ispirati al trend attuale di applicazioni delle varie aziende di spicco del settore, ossia abbiamo cercato di mantenere uno stile minimalistico per ridurre le distrazioni all'utente finale e rendere più chiara e settoriale ogni sezione. Come gestione tema è stato deciso di implementare `MaterialDesign3` al fine di rendere compatibile l'applicazione con l'ambiente dinamico del dispositivo. Le varie proprietà sono state specificate nella classe `Theme.kt`.

Lingue

L'intera applicazione è stata progetta in ottica di evoluzione e upgrade dell'applicazione stessa, infatti tutte le entità testuali sono fornite esternamente dalle risorse permettendo di implementare in modo molto semplice svariate lingue. Al momento attuale è stata implementata anche la lingua inglese: per utilizzarla è sufficiente cambiare la lingua dalle impostazioni del proprio dispositivo e automaticamente l'applicazione utilizzerà le componenti testuali impostate.

Grafici

Uno dei vantaggi per cui abbiamo privilegiato Jetpack Compose rispetto all'approccio dichiarativo è la modularità e scalabilità. Ciò ha permesso, infatti, di integrare nella nostra UI componenti grafiche molto complesse come LineChart, ColumnChart e PieChart, senza doverci troppo preoccupare della loro implementazione ma semplicemente di gestire l'aspetto estetico associato ai colori, le animazioni, il padding, le label e le griglie. In particolare, sono state utilizzate due librerie presenti nelle repository GitHub ComposeCharts e Tehras.

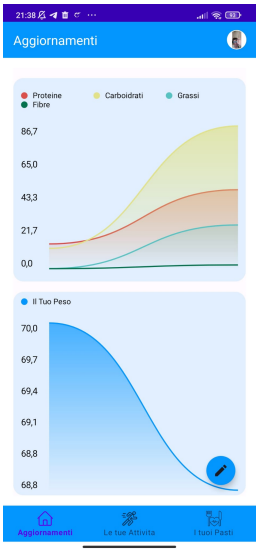


Figura 4.6: Esempi di grafici nella nostra applicazione

4.4 Design Pattern

Al fine di rendere l'applicazione modulare ed in futuro facilmente aggiornabile si è optato per il Design Pattern `MVVM Model View ViewModel` al fine di separare la gestione dei dati dalla presentazione; inoltre data la grande mole di dati che i vari `Fragment` devono computare si è optato per un design pattern di tipo `Singleton` implementando le istanze uniche all'interno di `Application()` per evitare problemi di integrità, inoltre anche Firebase implementa nativamente una gestione `Singleton` dell'istanza, mentre Room Database interno è sempre un `Singleton` generato tramite un design pattern `Factory`. Infine, gran parte del funzionamento dell'interfaccia si basa su un design pattern di tipo `Observer`.

4.5 Database

La nostra applicazione è organizzata con due database. Nonostante possa sembrare una scelta che va contro il principio della ridondanza dei dati, ciò

permette all'applicazione di funzionare allo stesso modo sia quando è connessa ad internet, sia quando non lo è. Il database locale è stato implementato attraverso Room, perciò si tratta si un database basato sul motore SQLite. Il database remoto invece fa parte dei servizi offerti dalla piattaforma Firebase, e quindi non ci addentriamo negli aspetti tecnici del sistema già descritto ampiamente nel capitolo precedente.

4.6 API

(Un)Health, nella sua sezione riguardante l'alimentazione, permette di inserire manualmente degli alimenti specificando le varie proprietà associate ad esso. Abbiamo deciso di automatizzare questa procedura, mantenendo comunque il comportamento precedente. L'idea che abbiamo voluto sfruttare è il fatto di poter scannerizzare il codice a barre di un prodotto alimentare, che lo identifica univocamente, per poter estrarre tutte le statistiche legate ad esso. Per poter implementare ciò, abbiamo utilizzato OpenFoodFacts. OpenFoodFacts è un progetto collaborativo e open source che mira a raccogliere e rendere disponibili informazioni dettagliate su prodotti alimentari di tutto il mondo. Nel concreto, il sistema offre un database collaborativo in cui chiunque può contribuire aggiungendo nuovi prodotti, modificando quelli esistenti e caricando fotografie delle etichette. Questo approccio crowdsourced permette di mantenere il database aggiornato e ampio, anche se non sempre sono presenti tutti gli alimenti e le proprietà, oppure alcuni alimenti presentano delle statistiche non aggiornate. E' uno strumento fondamentale nella analisi dati e nella ricerca in campo alimentare, e permette di fare molta chiarezza sulla qualità e le componenti che costituiscono un alimento. Oltre al database, è disponibile anche un'API per sviluppatori che vogliono integrare i dati nei loro progetti,

oltre ad una serie di SDK che permettono di utilizzare già al massimo del potenziale l'applicazione. Attraverso il client, una volta ottenuto il barcode grazie ad una libreria di scannerizzazione utilizzando la fotocamera, abbiamo effettuato la chiamata all'API FoodOpenFact, in particolare la versione "v0", ottenendo un file JSON contenente tutti i campi del prodotto scannerizzato. Una volta ottenuta la response, essa viene serializzata nelle varie entità specificate dai model corrispondenti grazie al serializzatore "Ktor" specificato nel client, che esegue il parsing del file JSON. Fatto ciò, l'entità `ProductResponse` viene convertita in un oggetto di tipo `Alimento`, che rappresenta l'effettivo alimento da inserire all'interno del nostro sistema. Allo stesso tempo, essa viene convertita in un'istanza di tipo `AlimentoWrapper`. Essa presenta le stesse caratteristiche di `Alimento` tranne che per le proprietà di tipo **`mutableStateOf`**. Ciò permette di mantenere coerente la componente visiva con le entità scannerizzate, automatizzando il riempimento dei campi alla ricezione della response di OpenFoodFacts.

4.7 Note

Lo sviluppo dell'applicazione è stato realizzato su vari dispositivi con varie API Level, per una migliore esperienza come emulatore consigliamo Pixel 8 Pro con API Level 34, ma funziona ugualmente bene su dispositivo fisico Redmi Note 8Pro con Api Level 29 e Poco X3 con Api Level 30 o in generale qualsiasi dispositivo compatibile con gli standard di versione. Il contenuto dei menu "Esistenti da altri utenti" esclude in automatico i dati presenti localmente per non generare confusione.

Attenzione: data l'aggressiva politica di notifiche di Android 14 se non attivata esplicitamente (impostazioni app -> notifiche) la notifica push del

tempo trascorso rimane nascosta ma perfettamente funzionante.

Bibliografia

[1] https://link.springer.com/chapter/10.1007/978-3-030-90022-9_12.

[2] https://www.corriere.it/tecnologia/23_febbraio_27/tiktok-allarme-degli-esperti-sulla-privacy-i-nostri-dati-personali-sono-senz shtml.

[3] https://www.scss.tcd.ie/Doug.Leith/Android_privacy_report.pdf.

[4] https://developer.android.com/games/sdk/performance-tuner?hl=it.

[5] https://www.appdynamics.com/.

[6] https://www.dynatrace.com/.

[7] https://developer.android.com/guide/topics/permissions/overview?hl=it.

[8] https://newrelic.com/.

[9] https://sentry.io/welcome/.

[10] https://www.datadoghq.com/product/.

[11] https://www.dotcom-monitor.com/.

[12] https://www.ibm.com/products/instana.

[13] https://www.logicmonitor.com/.

[14] https://raygun.com/.

[15] https://raygun.com/.

[16] https://logrocket.com/.

[17] https://firebase.google.com/.

[18] Yury Zhauniarovich and Olga Gadyatskaya. Small Changes, Big Changes: An Updated View on the Android Permission System. In *Proceedings of 19th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID 2016, pages 346–367, 2016.