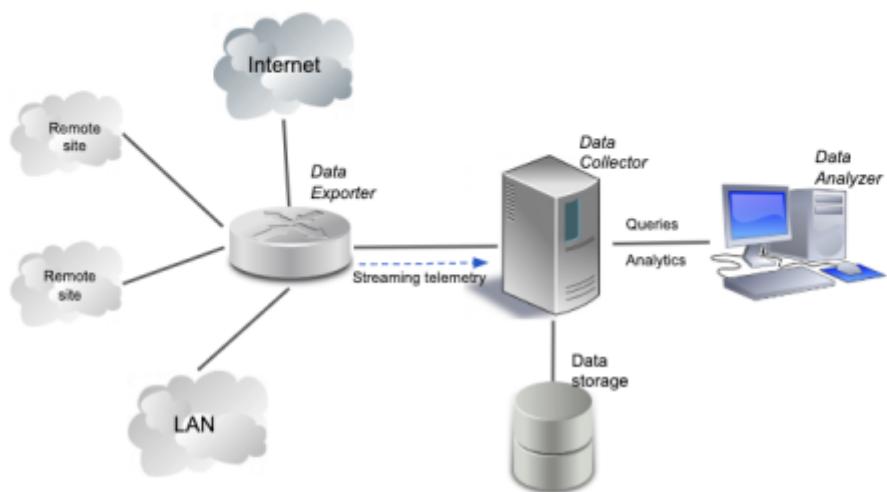


UNIVERSITÀ DEGLI STUDI DI PADOVA

CORSO DI PROGRAMMAZIONE DI SISTEMI EMBEDDED

## Telemetry



Anno Accademico : 2023-2024

## Contenuti

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Usi e benefici della telemetria</b>	<b>5</b>
<b>3</b>	<b>Rischi e usi malevoli della telemetria</b>	<b>10</b>
<b>4</b>	<b>La telemetria in Android</b>	<b>17</b>
4.1	Introduzione alla telemetria in Android . . . . .	17
4.2	OpenTelemetry . . . . .	21
4.3	Firebase Analytics . . . . .	29
4.3.1	Caratteristiche principali di Firebase Analytics . . . . .	29
4.3.2	Utilizzo di Firebase Analytics nella Telemetria . . . . .	30
4.3.3	Implementazione di Firebase Analytics . . . . .	30
<b>5</b>	<b>Privacy e Sicurezza</b>	<b>32</b>
5.1	Conformità alle Normative sulla Privacy . . . . .	32
5.1.1	Informativa sulla Privacy . . . . .	33
5.1.2	Raccolta del Consenso con Google Consent SDK . . . . .	33
5.1.3	Utilizzo di Matomo (ex Piwik) per la telemetria self-hosted . . . . .	34
5.1.4	Metodi per garantire la sicurezza dei dati . . . . .	34
5.2	Sviluppare un app Android in linea con le normative sulla privacy	35
<b>6</b>	<b>Applicazione sviluppata: gioco del memory</b>	<b>39</b>
6.1	Descrizione e Struttura dell'Applicazione . . . . .	39
6.1.1	Screenshot dell'Applicazione . . . . .	41
6.2	Integrazione di Firebase Analytics . . . . .	47
6.3	Dashboard di Firebase . . . . .	50
6.3.1	Firebase DebugView . . . . .	50
6.3.2	Firebase Realtime Overview . . . . .	51
6.3.3	Events Report . . . . .	53
6.3.4	Firebase Overview Report . . . . .	54

## 1 Introduzione

La telemetria consiste nell'utilizzo di sensori per raccogliere informazioni su varie grandezze fisiche, trasmettendo queste in tempo reale o periodicamente ad un sistema di controllo tramite reti di comunicazione.

Nello specifico la telemetria è costituita e caratterizzata da alcuni step fondamentali, ossia:

**Definizione degli obiettivi:** Si determinano quali dati devono essere monitorati e perché.

**Selezione dei sensori:** Si identificano i sensori necessari per la misura dei dati desiderati

**Implementazione dei dispositivi di raccolta dati:** Si installano i dispositivi di acquisizione dati (come data logger, sensori IoT o altri dispositivi di monitoraggio) sui sistemi da monitorare.

**Scelta della tecnologia di comunicazione:** Riguarda la trasmissione di dati dai dispositivi di raccolta dati ai sistemi di elaborazione. Le opzioni includono connessioni via cavo (Ethernet), wireless (Wi-Fi, Bluetooth, cellulari), o reti LPWAN.

**Configurazione della rete di comunicazione:** Una volta scelta la tecnologia di comunicazione, è necessario configurare l'infrastruttura di rete per supportare la trasmissione dei dati. Questo può includere:

- Configurazione della LAN, per applicazioni locali
- Configurazione della WAN: Per applicazioni distribuite su una vasta area geografica, è necessario configurare una Wide Area Network (WAN).

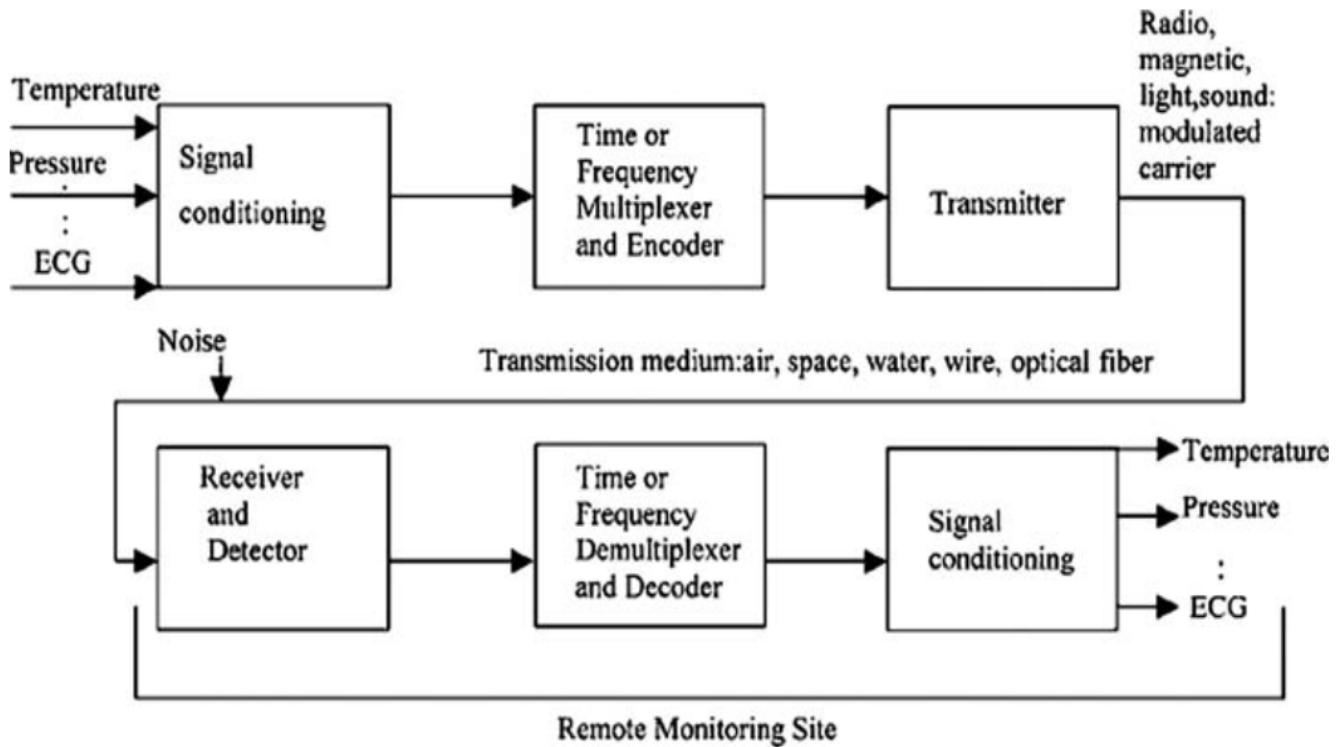
**Implementazione delle analisi e reporting:** Consiste nell'analisi dei dati raccolti, da cui vengono generati dei report o avvisi in base ai parametri monitorati. Le analisi possono implicare vari risvolti, tra cui la generazione di grafici, l'identificazione di anomalie o il monitoraggio predittivo.

**Test e ottimizzazione:** Una volta implementato, testa e ottimizza il sistema di telemetria per garantire che funzioni correttamente. In questa fase vengono effettuati aggiustamenti e miglioramenti in funzione dell'analisi dei dati prodotti a seguito dei test.

**Manutenzione e aggiornamenti:** Si mantiene il sistema aggiornato e si effettua la correzione di bug per garantire la sicurezza e l'affidabilità nel tempo.

**Sicurezza:** La telemetria coinvolge la trasmissione di dati critici da dispo-

sitivi remoti tramite reti di comunicazione, e garantire la sicurezza di questi dati è essenziale per prevenire accessi non autorizzati, manipolazioni malevoli o interruzioni delle operazioni di monitoraggio e controllo. Tratteremo questo aspetto in maniera più approfondita nel capitolo dedicato ai...

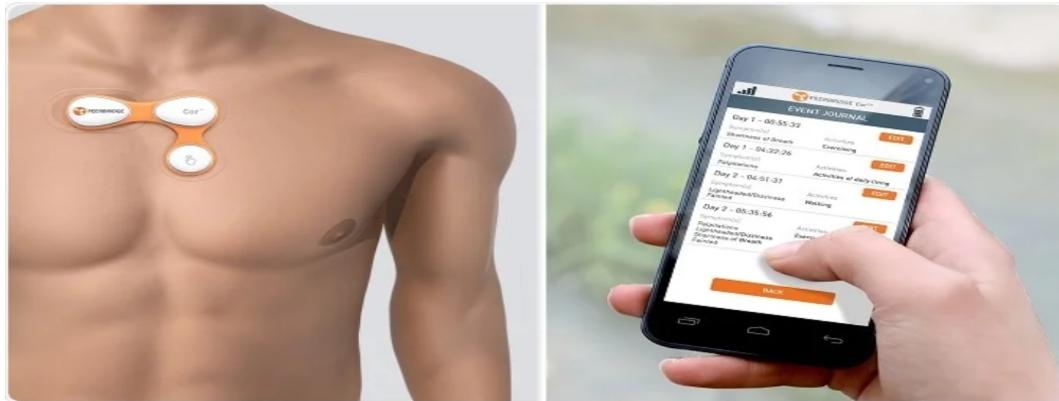


## 2 Usi e benefici della telemetria

La telemetria trova applicazione in svariati settori, tra cui: sanità, automazione...

Di seguito, facciamo una breve trattazione dei principali usi.

### Monitoraggio dei pazienti in ambito sanitario



In sanità, la telemetria è utilizzata per monitorare pazienti a rischio di attività cardiaca anomala, fornendo dati in tempo reale per la diagnosi e la gestione delle condizioni di salute.

Nello specifico, la telemetria wireless dei pazienti include un fase di monitoraggio, in cui i pazienti sono equipaggiati con dispositivi di monitoraggio che rilevano parametri vitali come la frequenza cardiaca, la pressione sanguigna e la temperatura corporea.

I dati raccolti dai dispositivi di monitoraggio vengono trasmessi attraverso il sistema WiFi a tablet e smartphone di cui sono stati dotati medici e infermieri, i quali analizzano i dati ed identificano eventuali anomalie o prendono decisioni mediche più informate. In caso di anomalie, i medici e gli infermieri possono intervenire rapidamente, ad esempio modificando la terapia o inviando un'ambulanza.

## Telemetria industriale

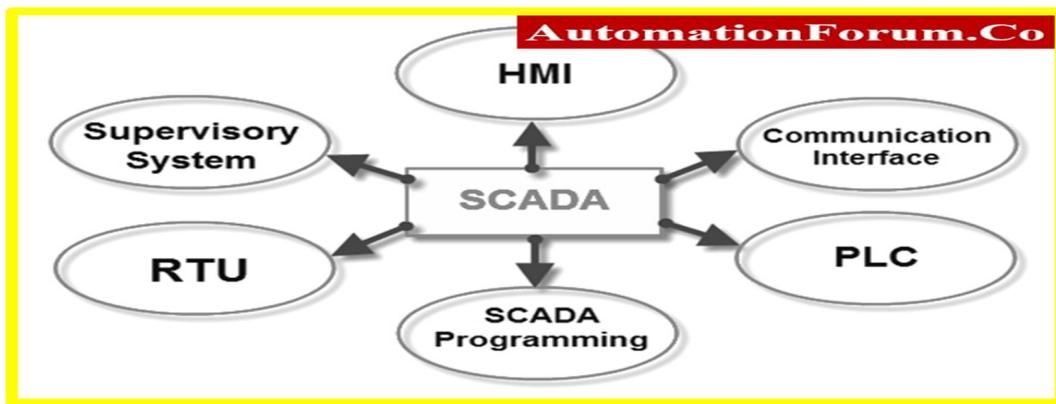
In ambito industriale la telemetria è utilizzata per monitorare le prestazioni delle macchine e degli impianti, aiutando a identificare problemi e a prendere misure preventive per evitare guasti. La telemetria industriale può essere utilizzata per monitorare parametri come la temperatura, la pressione, la velocità e la posizione di macchine e impianti, fornendo informazioni in tempo reale per la gestione e la manutenzione dei macchinari. Nelle applicazioni industriali spesso si utilizzano sistemi SCADA(Supervisory Control and Data Acquisition), ossia sistemi implementati per raccogliere i dati provenienti dai sensori in tempo reale e visualizzarli graficamente sui terminali degli operatori, consentendo loro di monitorare le prestazioni dell'impianto, i parametri critici e i livelli di produzione. Entrando più nel dettaglio, i sensori, attraverso protocolli di comunicazione industriali come Modbus TCP/IP o Profibus , trasmettono le misurazioni a degli RTU(Remote Terminal Unit), ossia dispositivi hardware che acquisiscono dati, elaborano le informazioni localmente e trasmettono i risultati ad un server SCADA.

Il server SCADA è il componente centrale del sistema e gestisce l'acquisizione, l'elaborazione e l'archiviazione dei dati, servendosi di Databases, per analisi storiche, tracciabilità e reportistica.

Un'interfaccia Utente (HMI - Human-Machine Interface) consente agli operatori di interagire con il sistema SCADA attraverso schermate grafiche. Le HMI mostrano informazioni chiare e intuitive sullo stato dei processi industriali e consentono agli operatori di eseguire azioni di controllo e monitoraggio.

Oltre a questi componenti, nei sistemi SCADA, possono essere presenti:

- **PLC**: I PLC (Programmable Logic Controller) sono dispositivi digitali programmabili utilizzati nell'automazione industriale per controllare macchinari e processi di produzione, nei sistemi SCADA hanno un ruolo simile ai RTU.
- **Sistemi di notifica/allarme**



## Localizzazione GPS e tracciamento dei Percorsi

Un esempio dell'uso della telemetria nella localizzazione GPS è nell'ambito dei servizi di consegna e logistica, in cui i veicoli sono dotati di un dispositivo GPS telemetrico che riceve costantemente segnali satellitari per determinare la posizione esatta. Questi dispositivi GPS trasmettono i dati di localizzazione in tempo reale al software di gestione della flotta.

Il software di gestione della flotta di solito opera su un'architettura client-server, dove il software client è utilizzato dagli operatori per interagire con il sistema centralizzato, è generalmente fornita un'interfaccia che offre una visualizzazione grafica della flotta di veicoli su una mappa interattiva, attraverso la quale i gestori della flotta possono monitorare i percorsi seguiti dai veicoli, le rotte pianificate e le fermate effettuate durante le consegne.

Questo è possibile perché i dati telemetrici dei veicoli vengono trasmessi al server centrale, dove vengono elaborati, archiviati e resi disponibili per l'accesso tramite l'interfaccia del software client. Il software comunica con i dispositivi telemetrici sui veicoli, utilizzando API (Application Programming Interface) e protocolli di comunicazione standard, come HTTP. Le API consentono al software di ricevere e inviare dati ai dispositivi telemetrici e di integrarsi con altri sistemi o applicazioni aziendali.



## Agricoltura 4.0

La telemetria agricola utilizza sensori di monitoraggio installati nei campi per raccogliere dati sulle condizioni ambientali, come temperatura, umidità, luminosità, pH del suolo e contenuto di nutrienti. Questi sensori trasmettono i dati in tempo reale a sistemi centralizzati di gestione agricola, consentendo ai coltivatori di monitorare le condizioni dei campi e prendere decisioni immediate sulle operazioni culturali.



Fornendo un esempio di applicazione tipica, si potrebbe immaginare al monitoraggio di umidità in un terreno agricolo.

Questo si realizzerebbe utilizzando sensori di umidità dotati di un modulo LoRa (componente hardware utilizzato per la trasmissione wireless a lungo raggio e a bassa potenza per dispositivi IoT ).

I sensori rileverebbero l'umidità del suolo e invierebbero i dati tramite segnali LoRa a un gateway. A questo punto, il gateway trasmetterebbe i dati ricevuti, tramite protocollo MQTT(un protocollo di messaggistica progettato per dispositivi con restrizioni di risorse) ad un server centrale. L'architettura del software centrale è solitamente basata su una piattaforma cloud, nello specifico, il software è ospitato su un backend cloud che include server virtuali, servizi di archiviazione come database NoSQL (ad es. MongoDB) o SQL (ad es. PostgreSQL), e servizi di calcolo per l'elaborazione dei dati. Le tecnologie cloud come AWS (Amazon Web Services), Microsoft Azure o Google Cloud Platform sono spesso utilizzate per implementare il software centrale.

A questo punto, gli API endpoint e i microservizi consentono l'interazione con il sistema da parte di applicazioni frontend, in particolar modo i microservizi gestiscono funzionalità specifiche, mentre, gli API endpoint espongono i dati elaborati in un formato strutturato e accessibile, che l'utente può visualizzare attraverso dei dashboard web.

### **3 Rischi e usi malevoli della telemetria**

Per quanto riguarda la telemetria ci sono alcuni rischi e potenziali usi malevoli che è importante considerare. Di seguito sono descritti alcuni dei rischi associati alla telemetria e alcuni utilizzi malevoli.

#### **Rischi della Telemetria:**

**Privacy e Sicurezza dei Dati:** Uno dei principali rischi è legato alla privacy e alla sicurezza dei dati raccolti. Se i dati di telemetria non vengono adeguatamente protetti e crittografati, potrebbero essere compromessi da attacchi informatici, portando a perdite di dati sensibili o violazioni della privacy degli utenti.

**Integrità dei Dati:** Se i dati di telemetria vengono manipolati o alterati da terze parti malevole, ciò potrebbe portare a decisioni errate o dannose basate su informazioni errate.

**Disponibilità del Servizio:** Gli attacchi di tipo Denial of Service (DoS) o Distributed Denial of Service (DDoS) possono mirare a sovraccaricare i sistemi di telemetria, rendendo i dati inaccessibili o causando interruzioni nei servizi critici.

**Manipolazione dei Dispositivi:** I dispositivi IoT utilizzati per la telemetria potrebbero essere vulnerabili a attacchi di hacking che mirano a prendere il controllo remoto dei dispositivi stessi. Questo potrebbe consentire a un attaccante di modificare le impostazioni, interrompere il funzionamento o causare danni fisici.

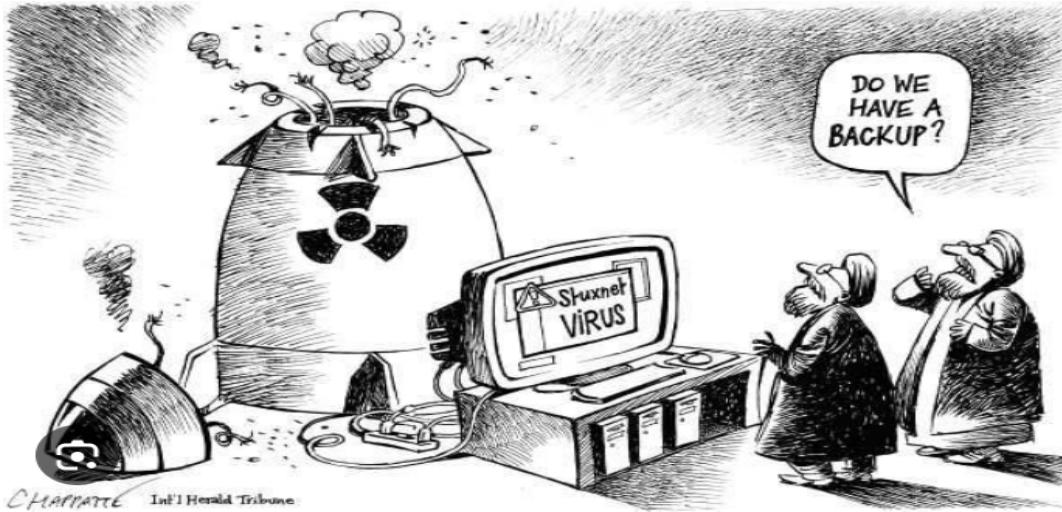
#### **Usi Malevoli della Telemetria:**

**Spionaggio e Sorveglianza Illegale:** La telemetria potrebbe essere utilizzata per scopi di sorveglianza non autorizzata o spionaggio, ad esempio monitorando i movimenti di persone o la loro attività senza consenso.

**Hacking e Attacchi Informatici:** I dati di telemetria raccolti possono essere sfruttati da hacker per identificare vulnerabilità nei sistemi o nelle reti e condurre attacchi mirati come phishing o furto di identità.

**Sabotaggio Industriale:** Un uso malevolo della telemetria potrebbe includere il sabotaggio di impianti industriali o infrastrutture critiche manipolando i dati di telemetria per causare malfunzionamenti o danni fisici.

## Il caso Stuxnet



Un esempio di quanto la sicurezza sia un aspetto critico nella telemetria, è rappresentato dall'attacco informatico noto come Stuxnet, avvenuto nel 2010 e mirato a compromettere il sistema di controllo di impianti industriali, in particolare le centrifughe nucleari utilizzate per l'arricchimento dell'uranio. Stuxnet è stato un sofisticato worm informatico scoperto nel giugno 2010, concepito per attaccare e danneggiare impianti industriali critici. L'obiettivo primario era sabotare le centrifughe utilizzate per l'arricchimento dell'uranio in impianti nucleari in Iran.

Stuxnet è stato distribuito principalmente attraverso dispositivi USB infetti e file eseguibili dannosi scaricati da Internet. L'obiettivo era di introdurre il worm nei sistemi di controllo industriale, che spesso sono offline o isolati dalla rete Internet per motivi di sicurezza.

Stuxnet ha sfruttato diverse vulnerabilità zero-day, cioè vulnerabilità sconosciute o non corrette, per ottenere l'accesso ai sistemi di destinazione. Ad esempio, il worm ha sfruttato una vulnerabilità critica di Windows nota come MS08-067 per compromettere i sistemi Windows target.

Una volta infettato il sistema, Stuxnet mirava specificamente ai PLC prodotti da Siemens utilizzando codici di identificazione unici. Il worm eseguiva una scansione delle reti target alla ricerca di dispositivi PLC Siemens.

Una volta individuato un PLC Siemens, Stuxnet sfruttava una vulnerabilità nel protocollo di autenticazione del PLC per ottenere l'accesso al sistema. Il

worm utilizzava credenziali di default per bypassare le protezioni di sicurezza. Dopo aver ottenuto l'accesso al PLC, Stuxnet caricava e eseguiva il suo codice dannoso all'interno del sistema PLC stesso. Questo codice dannoso era progettato per manipolare i parametri di controllo delle centrifughe, come la velocità di rotazione e altre variabili critiche, causando danni alle apparecchiature e interruzioni nei processi industriali.

Una delle caratteristiche più insidiose di Stuxnet era la sua capacità di mascherare le sue attività dannose. Il worm manipolava i dati dei sensori delle centrifughe per fornire informazioni fuorvianti agli operatori, facendo credere che le macchine funzionassero normalmente mentre erano soggette a manipolazioni dannose.

La compromissione dei PLC da parte di Stuxnet ha avuto un impatto significativo sull'efficienza e sull'integrità degli impianti nucleari coinvolti, dimostrando il potenziale delle minacce informatiche avanzate nei confronti delle infrastrutture critiche.

## Il caso Cambridge Analytica

Per fare un esempio reale di attacco alla privacy dovuto a un uso malevolo/improprio della telemetria possiamo trattare il caso di Cambridge Analytica. Cambridge Analytica era una società di consulenza politica e analisi dei dati che ha ricevuto attenzione globale nel 2018 per le sue pratiche di raccolta e utilizzo dei dati personali di milioni di utenti di Facebook senza il loro consenso.

L'azienda ha ottenuto dati personali da milioni di utenti di Facebook attraverso un'applicazione di quiz psicologici chiamata "thisisyourdigitallife". Gli utenti che hanno interagito con questa app hanno fornito il proprio consenso per partecipare a un quiz che esplorava aspetti della loro personalità.

Gli utenti che hanno completato il quiz hanno condiviso una serie di informazioni personali, come il loro nome, la data di nascita, il luogo di residenza e le loro preferenze.

Inoltre, l'applicazione ha richiesto l'autorizzazione per accedere ai propri dati su Facebook, inclusi i "Mi piace", le informazioni sulle attività online e altre informazioni dettagliate sul profilo dell'utente.

Uno degli aspetti più controversi di questa pratica è stato il fatto che l'appli-

cazione "thisisyourdigitallife" ha anche ottenuto accesso ai dati degli amici degli utenti senza il loro consenso esplicito. Ciò è stato possibile a causa delle politiche di condivisione dei dati di Facebook in vigore all'epoca.

Una volta raccolti i dati, Cambridge Analytica ha eseguito analisi avanzate per creare profili psicografici dettagliati degli utenti. Questi profili sono stati utilizzati per influenzare comportamenti e opinioni politiche attraverso strategie di marketing mirate.

Utilizzando algoritmi di analisi dei dati, Cambridge Analytica ha categorizzato gli utenti in base a tratti psicologici, preferenze personali e atteggiamenti politici.

Questi profili sono stati utilizzati per creare messaggi persuasivi e campagne di marketing politico altamente mirate. Cambridge Analytica ha utilizzato i dati raccolti e gli insight ottenuti attraverso l'analisi per influenzare il comportamento degli elettori durante le campagne elettorali del 2016 negli Stati Uniti.

Ciò includeva la creazione di contenuti politici personalizzati e mirati per influenzare le opinioni degli utenti su temi politici chiave.

Questa vicenda dimostra come un uso malevolo della telemetria possa avere gravi conseguenze per la privacy degli individui e per l'integrità dei processi democratici.



## ”Iridium Hacking”

Come affermato in precedenza una delle fasi principali di un'applicazione telemetrica è la trasmissione di informazioni da sensori a server per l'elaborazione di dati.

Una delle criticità che si possono verificare è quella del Man-in-the-Middle (MitM):

In un attacco MitM, un attaccante intercetta e manipola la comunicazione tra due parti legittime. Nelle applicazioni di telemetria questo potrebbe appunto avvenire tra sensori e server di raccolta dati ma anche tra server e dashboard di visualizzazione.



Nel 2014, un gruppo di ricercatori di sicurezza informatica presso la Sys-Scan360 Security Conference ha dimostrato con successo un attacco MitM al sistema di comunicazione satellitare Iridium.

Iridium è una rete globale di satelliti utilizzata per comunicazioni vocali e dati, incluso il monitoraggio e il controllo remoto di dispositivi e sensori.

I ricercatori hanno utilizzato attrezzature di basso costo, tra cui un ricevitore software-defined radio (SDR).

Un SDR è un dispositivo hardware in grado di ricevere e decodificare segnali radio utilizzando software configurabile.

Il SDR è stato programmato per sintonizzarsi sulle frequenze utilizzate dalla rete Iridium per le comunicazioni ascendenti (uplink) dai dispositivi terrestri ai satelliti.

Queste comunicazioni includono messaggi di controllo e dati di telemetria

inviai dai dispositivi remoti.

Utilizzando il software configurabile del SDR, i ricercatori hanno decodificato e analizzato i segnali radio ricevuti dagli satelliti Iridium.

Questo ha incluso l'identificazione dei pacchetti di dati e dei comandi inviati ai dispositivi connessi.

Quindi, i ricercatori hanno modificato questi pacchetti di dati prima di reinviarli ai satelliti Iridium.

Ad esempio, hanno potuto cambiare i parametri dei comandi di telemetria o inviare comandi di controllo alterati ai dispositivi remoti.

Di seguito vengono indicate le dichiarazioni di Sec, uno dei ricercatori che ha contribuito all'hacking:

”La rete Iridium ha una limitata larghezza di banda dati, solo 2,4 Kb/sec, il suo utilizzo è limitato a specifiche applicazioni.

Attualmente, la rete Iridium viene utilizzata per scopi di tracciamento nella logistica e nei trasporti.

Solo per darti un esempio, il traffico dei pager Iridium non utilizza di default la crittografia e la maggior parte del traffico viene inviato in testo non cifrato. Una presentazione trapelata da Wikileaks nel 2008 afferma che la rete Iridium è difficile da hackerare, ma le cose sono molto diverse.

La complessità dell'interfaccia radio Iridium rende molto difficile lo sviluppo di un dispositivo di monitoraggio Iridium L-Band, probabilmente al di là delle capacità della maggior parte degli avversari determinati.

Il sistema Iridium potrebbe essere violato con una radio software economica come il badge rad1o o l'HackRF.

La maggior parte del traffico dei pager potrebbe essere facilmente intercettato con prodotti disponibili sul mercato e il software utilizzato dai ricercatori. Con solo il badge rad1o e un'antenna PCB integrata, è possibile raccogliere il 22% di tutti i pacchetti che si possono ricevere con un'antenna Iridium adeguata.

Basta caricare il software sul tuo PC, collegare il badge rad1o e puoi iniziare a ricevere i messaggi dei pager Iridium. Quindi, buon hacking!.

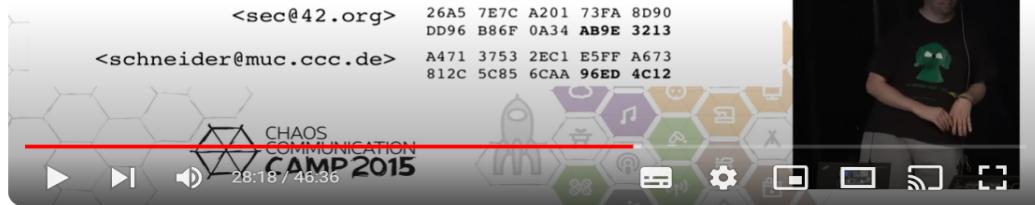
Gli esperti hanno evidenziato che il badge ha una capacità di elaborazione limitata, il che implica che l'analisi del traffico Iridium richiede un software specifico (Iridium toolchain) che potrebbe essere eseguito su un laptop o su un economico Raspberry Pi 2.

Un Raspberry Pi 2 ha esattamente la potenza di calcolo necessaria per el-

borare il traffico.”

Durante la presentazione, Sec ha mostrato una demo live dell’intercettazione del traffico Iridium, mostrando come catturare e decodificare il traffico dei pager Iridium.

- SDR Workshop in 30 minutes (also maybe tomorrow)
  - 17:45 in Hall 13, Bring Laptop
- We have Equipment @ µc<sup>3</sup> Village
  - Network Analyzer, USRP B-200
- Code is on github (BSD Licence)
  - <https://github.com/mucc/iridium-toolkit>
- Iridium System Specification / Iridium Radio Link Protocol Specification
  - We still want it
  - No questions asked



**Sec, schneider: Iridium Hacking**



## 4 La telemetria in Android

### 4.1 Introduzione alla telemetria in Android

La telemetria in Android funziona attraverso una serie di componenti e processi che permettono la raccolta, l'elaborazione e l'analisi dei dati generati dai dispositivi.

Definiamo i principali componenti della telemetria in Android:

- Sensors API: Android fornisce un'API per accedere ai sensori hardware presenti sui dispositivi, come accelerometro, giroscopio, sensore di prossimità, sensore di luce, ecc.
- Usage Statistics e Battery Stats: Android tiene traccia dell'utilizzo delle applicazioni, delle risorse di sistema e della durata della batteria.
- Location Services: Android offre servizi di localizzazione che consentono alle applicazioni di accedere alla posizione geografica del dispositivo.

Network Traffic Monitoring: Android monitora il traffico di rete generato dalle applicazioni e dal sistema operativo. Questo include informazioni sulle connessioni di rete attive, il volume di dati trasmessi e ricevuti, la velocità di connessione, ecc.

Una tipologia di tool largamente diffusa ed utilizzata sono gli APM (Application Performance Monitoring). Sono strumenti che permettono di cogliere eventuali colli di bottiglia nelle performance delle applicazioni, al fine di migliorare le prestazioni.

Le caratteristiche di un APM possono essere riassunte nei seguenti punti:

- permettere di mostrare l'impatto e le performance al fine di semplificare la risoluzione di problemi nelle applicazioni.
- concedere la possibilità di integrazioni con app di terze parti per facilitarne l'utilizzo.
- gestire e rendere osservabile dall'inizio alla fine i comportamenti dell'applicazione o di parti di essa.
- evidenziare all'interno dell'applicazione o di suoi componenti eventuali colli di bottiglia nel flusso d'esecuzione.
- mostrare come eventuali modifiche infrastrutturali impattino sull'applicazione e le sue performance.

Nello specifico gli APM rilevano i dati degli indicatori chiave (KPIs), che riguardano sia i singoli processi che l'esecuzione dell'intero programma. Le metriche fondamentali utilizzate sono:

1. Utilizzo CPU
2. Tasso di richieste
3. Tasso di inattività
4. Frequenza di errori
5. Tempo di risposta

I tool più diffusi sono i seguenti: Dynatrace, Datadog, AppDynamics, Dotcom-Monitor, Instana, Logic Monitor, Scout e Raygun. Ognuno dei tool citati è a pagamento con alcuni che permettono una versione di prova gratuita o una versione molto limitata gratuita.

- Dynatrace sfrutta la potenza dell'AI, per permettere al programmatore di conoscere i processi da ottimizzare. Inoltre predice e risolve problemi nell'app prima che causino gravi disagi all'utente.
- Datadog permette il monitoraggio continuo del profilo, rilevamento automatico degli errori tramite intelligenza artificiale, integrazioni con oltre 400 piattaforme e strumenti, ricerche basate su tag per facilitare il completamento della ricerca
- AppDynamics offre il rilevamento automatico degli errori, analisi delle cause di radice alimentata da intelligenza artificiale, supporto per il monitoraggio dell'ambiente ibrido.
- Dotcome-Monitor permette il monitoraggio delle prestazioni in tempo reale per pagine web, API e applicazioni web, supporto per la creazione di script per transazioni web chiave, integrazione con strumenti popolari come Asana, Salesforce e BMC.

- Instana presenta un automatizzazione del monitoraggio senza necessità di plug-in aggiuntivi, mappatura delle dipendenze, supporto per l’analisi dei guasti potenziata dall’intelligenza artificiale.
- LogicMonitor si basa su OpenTelemetry e OpenMetrics, supportando l’alerting system per tracciamento e il monitoraggio delle attività e metriche
- Scout offre il supporto per un sistema di allerta intelligente, identificazione semplice di query lente e problematiche di prestazioni del backend dell’applicazione.
- Raygun presenta una grande integrazione con strumenti popolari come Bitbucket, GitLab, Jira, GitHub e altri strumenti di sviluppo, consentendo agli sviluppatori di ottenere informazioni azionabili sulle prestazioni del lato server. In generale offre monitoraggio delle prestazioni del lato server

L’utilizzo di strumenti come gli APM sono fondamentali per qualsiasi business in quanto avvicinano le aspettative dell’utente reali con quelle che la società si aspetta. Inoltre restituiscono un feedback fondamentale su eventuali problemi o possibili ottimizzazioni e implementazioni future.

All’interno della famiglia degli APM, esistono dei più specifici strumenti per il monitoraggio dei soli dispositivi android, uno tra questi è APT (Android Performance Tuner). I vantaggi di questo tool sono molteplici, a partire dalla capacità di comprendere i tempi di caricamento e l’impatto che hanno sull’utente, continuando con una misura complessiva sulla qualità dell’esperienza del pubblico fino a un’analisi prestazionale delle applicazioni. Inoltre permette di cogliere i dispositivi in cui certe applicazioni funzionano a pieno regime, mentre arrancano su dispositivi differenti.

A differenza dei tool visti finora, che si integrano con altri applicazioni per lo sviluppo, sono presenti dei software che includono ambedue i servizi. Firebase è una piattaforma di sviluppo di app mobile e web, nonché una piattaforma di sviluppo di applicazioni back-end, fornita da Google. Offre una vasta gamma di servizi e strumenti per gli sviluppatori, che includono:

1. Database in tempo reale: Un database cloud in tempo reale che consente agli sviluppatori di sincronizzare i dati tra gli utenti in tempo reale.

2. Autenticazione: Servizio di autenticazione che consente agli sviluppatori di gestire l'autenticazione degli utenti tramite email, social media o altri metodi.
3. Hosting: Servizio di hosting web che consente agli sviluppatori di distribuire facilmente le loro app web statiche o dinamiche.
4. Cloud Functions: Servizio per l'esecuzione di codice serverless in risposta agli eventi generati dai servizi di Google Cloud Platform o da altre fonti.
5. Storage: Servizio di archiviazione cloud che consente agli sviluppatori di memorizzare e distribuire file multimediali e altri dati generati dall'utente.
6. Messaggistica Cloud: Servizio per l'invio di notifiche push e messaggi diretti ai dispositivi mobili.
7. Analisi: Strumenti per l'analisi delle prestazioni e dell'utilizzo dell'app.
8. Testing e Monitoraggio delle prestazioni: Strumenti per testare e monitorare le prestazioni dell'app e il feedback degli utenti.

## 4.2 OpenTelemetry

OpenTelemetry è un progetto open-source che mira a fornire un set unificato di API, SDK e strumenti per la raccolta di dati di telemetria (tracing, metrics, logs) da applicazioni.

Un obiettivo principale di OpenTelemetry è che sia facilmente implementabile nelle applicazioni o sistemi, indipendentemente dalla loro lingua, infrastruttura o ambiente di runtime.

La memorizzazione e la visualizzazione della telemetria sono intenzionalmente lasciate ad altri strumenti.

### Componenti di OpenTelemetry

**API:** Le API di OpenTelemetry permettono agli sviluppatori di integrare tracing, metrics e logs nelle loro applicazioni.

**SDK:** Implementazioni delle API che forniscono la logica necessaria per raccogliere, elaborare e inviare i dati di telemetria.

Gli SDK includono configurazioni e strumenti per personalizzare il comportamento della raccolta dei dati.

**Exporters:** Moduli che inviano i dati raccolti a vari backend di monitoraggio e osservabilità come Jaeger, Zipkin, Prometheus, New Relic, Datadog, ecc.

**Instrumentation Libraries:** Librerie pronte all'uso che strumentano automaticamente framework e librerie comuni per raccogliere dati di telemetria senza richiedere modifiche significative al codice dell'applicazione.

### Usare OpenTelemetry nella propria app Android

La configurazione di OpenTelemetry nelle applicazioni Android può variare leggermente a seconda della versione di Android, ma le differenze principali riguardano solitamente alcune funzionalità di OpenTelemetry, che possono richiedere versioni minime specifiche dell'API di Android.

Di seguito si presenta una breve guida per la configurazione di OpenTelemetry in app programmate in Kotlin.

## Step 1: Configurazione del progetto

1. Aggiungere nel file build.gradle del modulo app le dipendenze necessarie.

```
implementation platform('io.opentelemetry:opentelemetry-bom:1.25.0')
implementation "io.opentelemetry:opentelemetry-api"
implementation "io.opentelemetry:opentelemetry-context"
implementation 'io.opentelemetry:opentelemetry-exporter-otlp'
implementation 'io.opentelemetry:opentelemetry-exporter-logging'
implementation 'io.opentelemetry:opentelemetry-extension-kotlin'
implementation 'io.opentelemetry:opentelemetry-sdk'
implementation 'io.opentelemetry:opentelemetry-semconv'
```

2. Creare un file denominato: network\_security\_config.xml nella directory app/res/xml e aggiungere il seguente contenuto al file:

```
<!-- To view the complete content of the file, visit https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidJavaDemo/app/src/main/res/xml/network_security_config.xml. -->

<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
<domain-config cleartextTrafficPermitted="true">
    <!-- Replace the domain name with the endpoint that you have obtained as instructed in the "Prerequisites" section of this topic. The endpoint cannot contain "http://", a port number, or a URL path. -->
    <domain includeSubdomains="true">tracing-analysis-dc-hz.aliyuncs.com</domain>
</domain-config>
</network-security-config>
```

3. Modificare il file app/src/main/AndroidManifest.xml aggiungendo il seguente contenuto per consentire all'applicazione di accedere alla rete:

```
<!-- To view the complete content of the file, visit https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidJavaDemo/app/src/main/AndroidManifest.xml. -->

<?xml version="1.0" encoding="utf-8"?>
<manifest ...>
    <!-- Add the following line to grant network access permissions. -->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        ...
        <!-- Add the following line to configure the network for the domain name to which you want to report data. -->
        android:networkSecurityConfig="@xml/network_security_config"
        ...>

        ...
    </application>
</manifest>
```

## Step 2: Inizializzazione di OpenTelemetry

### 1. Creare un utility class OpenTelemetry.

Creare un file OpenTelemetryUtil nella directory in cui risiede il file MainActivity e aggiungere il contenuto seguente al file OpenTelemetryUtil.

```
/*
Visit the following Link to view the complete code:
https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidKotlinDemo/app/src/main/java/com/example/androidkotlindemo/OpenTelemetryUtil.kt
*/

val otelResource = Resource.getDefault().merge(
    Resource.create(
        Attributes.of(
            ResourceAttributes.SERVICE_NAME, "<your-service-name>", // Replace <your-service-name> with the name of your application.
            ResourceAttributes.HOST_NAME, "<your-host-name>" // Replace <your-host-name> with the name of your host.
        )
    )
)

/* Report trace data over HTTP. */
val sdkTracerProvider = SdkTracerProvider.builder()
    .addSpanProcessor(SimpleSpanProcessor.create(LoggingSpanExporter.create())) // Display the trace data in Logs or the command Line. This Line is optional. Comment this Line if you do not need it.
    // Replace <HTTP-endpoint> with the endpoint that you have obtained as instructed in the "Prerequisites" section of this topic.
    .addSpanProcessor(BatchSpanProcessor.builder(
        OtlpHttpSpanExporter.builder()
            .setEndpoint("<HTTP-endpoint>") // Example: http://tracing-analysis-dc-hz.aliyuncs.com/adapter_xxxx@xxxx_xxxx@xxxx/api/otlp/traces
            .build().build()
    ))
    .setResource(otelResource)
    .build();

val openTelemetry: OpenTelemetry = OpenTelemetrySdk.builder()
    .setTracerProvider(sdkTracerProvider)
    .setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInstance()))
    .buildAndRegisterGlobal()

// Obtain the tracer that is used to create a span.
tracer = openTelemetry.getTracer("android-tracer", "1.0.0")
```

### 2. Creare un file denominato: network\_security\_config.xml nella directory app/res/xml e aggiungere il seguente contenuto al file:

```
<!-- To view the complete content of the file, visit https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidJavaDemo/app/src/main/res/xml/network_security_config.xml. -->

<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <!-- Replace the domain name with the endpoint that you have obtained as instructed in the "Prerequisites" section of this topic. The endpoint cannot contain "http://", a port number, or a URL path. -->
        <domain includeSubdomains="true">tracing-analysis-dc-hz.aliyuncs.com</domain>
    </domain-config>
</network-security-config>
```

### 3. Inizializzare OpenTelemetry durante l'inizializzazione dell'applicazione. Chiamare il metodo OpenTelemetryUtil.init() nel metodo onCreate della classe MainActivity

```

/*
Visit the following Link to view the complete code:
https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidKotlinDemo/app/src/main/java/com/example/androidkotlindemo/MainActivity.kt
*/
...

class MainActivity : AppCompatActivity() {

    ...

    override fun onCreate(savedInstanceState: Bundle?) {
        WindowCompat.setDecorFitsSystemWindows(window, false)
        super.onCreate(savedInstanceState)
        // Add the following line to initialize OpenTelemetry.
        OpenTelemetryUtil.init()

        ...
    }
}

```

### Step 3: Creare uno span per il tracciamento.

1. Creare uno span (unità di lavoro che rappresentano operazioni o attività che possono essere tracciate per ottenere informazioni sul comportamento e le prestazioni dell'applicazione).

Nel metodo di monitoraggio degli eventi di tocco dei pulsanti, nel file FirstFragment creare uno span denominato First Fragment Button onClick.

```

/*
Visit the following Link to view the complete code:
https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidKotlinDemo/app/src/main/java/com/example/androidkotlindemo/FirstFragment.kt
*/

binding.buttonFirst.setOnClickListener {
    // Obtain the tracer.
    val tracer: Tracer = OpenTelemetryUtil.getTracer()!!
    // Create a span.
    val span = tracer.spanBuilder("First Fragment Button onClick").startSpan()
    try {
        span.makeCurrent().use { scope ->
            // obtain the trace ID.
            println(span.spanContext.traceId)
            // obtain the span ID.
            println(span.spanContext.spanId)

            findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)
        }
    } catch (t: Throwable) {
        span.setStatus(StatusCode.ERROR, "Something wrong in onClick")
        throw t
    } finally {
        span.end()
    }
}

```

2. Configurare gli attributi e un evento per lo span.

```

/*
Visit the following Link to view the complete code:
https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidKotlinDemo/app/src/main/java/com/example/androidkotlindemo/FirstFragment.kt
*/

// Configure attributes.
span.setAttribute("key", "value")
val eventAttributes =
    Attributes.of(
        AttributeKey.stringKey("key"), "value",
        AttributeKey.longKey("result"), 0L
    )

// Add an event.
span.addEvent("onClick", eventAttributes)

```

### 3. Configurare lo stato dello span.

```

/*
Visit the following Link to view the complete code:
https://github.com/alibabacloud-observability/android-demo/blob/master/AndroidKotlinDemo/app/src/main/java/com/example/androidkotlindemo/FirstFragment.kt
*/

...
try {
    ...
} catch (t: Throwable) {
    // Configure the status of the span.
    span.setStatus(statusCode.ERROR, "Something wrong in onClick")
    throw t
} finally {
    span.end()
}

```

### 4. Creare uno span nidificato.

Gli span nidificati permettono di tracciare una struttura gerarchica di operazioni, dove uno span può contenere altri span come sotto-attività.

```

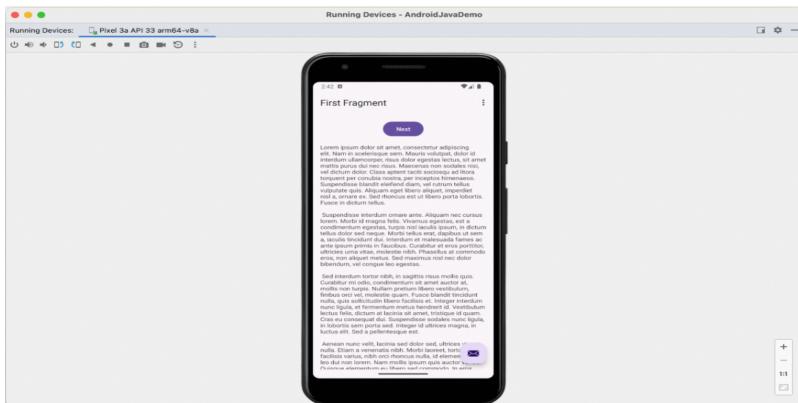
/**
Visit the following Link to view the complete code:
https://github.com/alibabaCloud-observability/android-demo/blob/master/AndroidKotlinDemo/app/src/main/java/com/example/androidkotlindemo/FirstFragment.kt
*/

// Nested span.
fun parentspan() {
    // Obtain the tracer.
    val tracer: Tracer = OpenTelemetryUtil.getTracer()!!
    // Create a span.
    val span = tracer.spanBuilder("Parent Span").startSpan()
    try {
        span.makeCurrent().use { scope ->
            // Obtain the trace ID.
            println(span.spanContext.traceId)
            // Obtain the span ID.
            println(span.spanContext.spanId)
            childspan()
        }
    } finally {
        span.end()
    }
}

// Nested span.
fun childspan() {
    // Obtain the tracer.
    val tracer: Tracer = OpenTelemetryUtil.getTracer()!!
    // Create a span.
    val span = tracer.spanBuilder("Child span").startSpan()
    try {
        span.makeCurrent().use { scope ->
            // Obtain the trace ID.
            println(span.spanContext.traceId)
            // Obtain the span ID.
            println(span.spanContext.spanId)
        }
    } finally {
        span.end()
    }
}

```

## Step 4: Eseguire il progetto per visualizzare i dati di tracing



fonti: [OpenTelemetry Documentation](#) , Use OpenTelemetry to report the trace data of Android applications

## **Vantaggi**

OpenTelemetry offre una serie di vantaggi significativi per il monitoraggio delle applicazioni. Eccone alcuni:

### **Standardizzazione**

OpenTelemetry unifica i principali standard di monitoraggio e tracciamento (OpenTracing e OpenCensus), fornendo un framework comune per la raccolta e l'analisi dei dati di telemetria.

### **Compatibilità e Interoperabilità**

OpenTelemetry supporta un'ampia gamma di linguaggi di programmazione, è integrabile in più piattaforme e garantisce compatibilità e interoperabilità tra diversi strumenti di monitoraggio e analisi.

### **Modularità e Flessibilità**

OpenTelemetry ha un'architettura Pluggable, ossia la modularità di OpenTelemetry consente, come affermato in precedenza, di scegliere e configurare diversi exporter per inviare i dati a vari backend di analisi (come Jaeger, Zipkin, Prometheus, Datadog, etc.).

### **Supporto alla Telemetria Completa**

OpenTelemetry supporta la raccolta di metriche, tracce e log, offrendo una soluzione completa.

### **Open Source**

Essendo un progetto open source, OpenTelemetry beneficia di una vasta comunità di sviluppatori e contributori che migliorano costantemente il framework.

## **Svantaggi**

Sebbene OpenTelemetry offra molti vantaggi, ci sono anche alcuni svantaggi:

### **Complessità di Implementazione**

L'apprendimento e l'implementazione di OpenTelemetry possono risultare complessi e la configurazione può risultare tediosa.

### **Overhead di Performance**

La raccolta e l'elaborazione dei dati di telemetria possono aggiungere overhead alle applicazioni, influenzandone le performance.

### **Dipendenza da Terze Parti**

L'efficacia di OpenTelemetry dipende dall'uso di exporter per inviare i dati a backend di terze parti. Se questi backend hanno limitazioni o problemi, ciò può influenzare l'intera soluzione di monitoraggio, quindi, OpenTelemetry non si utilizza per applicazioni di telemetria self-hosted, quindi applicazioni in cui i dati raccolti sono interamente gestiti nel sistema stesso, anziché essere affidati a servizi terzi.

## 4.3 Firebase Analytics

Firebase Analytics è una parte essenziale della piattaforma Firebase di Google, utilizzata principalmente per il monitoraggio delle app mobili e web. Questo strumento di analisi è particolarmente utile per la telemetria, poiché consente di raccogliere dati dettagliati sull'uso dell'applicazione, il comportamento degli utenti e le prestazioni dell'app stessa.

### 4.3.1 Caratteristiche principali di Firebase Analytics

- **Raccolta Automatica dei Dati:** Firebase Analytics raccoglie automaticamente i dati di utilizzo di base e gli eventi predefiniti senza richiedere configurazioni aggiuntive. Questi dati includono l'interazione con l'app, gli errori, i tempi di caricamento e molto altro.
- **Eventi Personalizzati:** Gli sviluppatori possono definire eventi personalizzati per monitorare specifiche interazioni degli utenti con l'applicazione. Questo è particolarmente utile per tracciare obiettivi aziendali specifici, come l'acquisto di prodotti o la visualizzazione di contenuti particolari.
- **Segmentazione del Pubblico:** Firebase Analytics consente di segmentare il pubblico in base a vari criteri, come la demografia, il comportamento e la tecnologia utilizzata. Questo aiuta a comprendere meglio le diverse tipologie di utenti e a personalizzare le esperienze in base alle loro esigenze.
- **Integrazione con Altri Strumenti Firebase:** Firebase Analytics si integra perfettamente con altri strumenti della suite Firebase, come Firebase Cloud Messaging, Firebase Remote Config, e Firebase A/B Testing. Questa integrazione facilita l'invio di notifiche personalizzate, l'implementazione di configurazioni dinamiche e la conduzione di esperimenti per migliorare l'app.
- **Dashboard Intuitiva:** La dashboard di Firebase Analytics offre una visualizzazione chiara e dettagliata dei dati raccolti, permettendo agli sviluppatori di monitorare le metriche chiave e di prendere decisioni informate basate sui dati.

#### 4.3.2 Utilizzo di Firebase Analytics nella Telemetria

Firebase Analytics è particolarmente efficace per la telemetria nelle applicazioni Android grazie alla sua capacità di raccogliere e analizzare grandi quantità di dati in tempo reale. Di seguito sono riportati alcuni dei principali usi di Firebase Analytics nel contesto della telemetria:

- **Monitoraggio delle Prestazioni:** Firebase Analytics aiuta a monitorare le prestazioni delle applicazioni identificando eventuali colli di bottiglia, errori o rallentamenti. Questi dati sono cruciali per garantire un'esperienza utente fluida e senza interruzioni.
- **Analisi Comportamentale:** Consente di analizzare il comportamento degli utenti all'interno dell'applicazione, comprendendo quali funzionalità sono più utilizzate e quali possono essere migliorate. Questa analisi è fondamentale per l'ottimizzazione continua dell'app.
- **Tracciamento delle Conversioni:** Firebase Analytics traccia le conversioni e altre azioni significative compiute dagli utenti, permettendo di misurare l'efficacia delle campagne di marketing e delle strategie di engagement.
- **Gestione degli Incidenti:** La raccolta di dati sugli errori e sui crash dell'app aiuta gli sviluppatori a individuare rapidamente i problemi e a implementare le correzioni necessarie per migliorare la stabilità dell'app.

#### 4.3.3 Implementazione di Firebase Analytics

L'implementazione di Firebase Analytics in un'app Android è semplice e richiede pochi passaggi:

1. **Aggiunta delle Dipendenze:** Nel file `build.gradle` del modulo dell'app, aggiungere le dipendenze per Firebase Analytics.

```
implementation 'com.google.firebaseio:firebase-analytics:18.0.0'
```

2. **Inizializzazione di Firebase:** Inizializzare Firebase nel metodo `onCreate` della classe `MainActivity`.

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    FirebaseApp.initializeApp(this);  
}
```

3. **Definizione degli Eventi Personalizzati:** Utilizzare il seguente codice per registrare eventi personalizzati.

```
FirebaseAnalytics mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);  
  
Bundle bundle = new Bundle();  
bundle.putString(FirebaseAnalytics.Param.ITEM_ID, "id_a");  
bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, "name_a");  
bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "type_a");  
mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
```

## 5 Privacy e Sicurezza

### 5.1 Conformità alle Normative sulla Privacy

La raccolta dei dati tramite telemetria solleva importanti questioni relative alla privacy e alla sicurezza. In particolar modo, lo sviluppo delle app deve prevedere la conformità della raccolta dei dati rispetto alle leggi ed i regolamenti sulla privacy, come il Regolamento Generale sulla Protezione dei Dati (GDPR) in Europa o il California Consumer Privacy Act (CCPA) negli Stati Uniti.

Per le app soggette a GDPR, sono stabiliti dei principi fondamentali a cui si deve attenere la raccolta ed il trattamento dei dati, cioè:

- **Trasparenza:** I dati personali devono essere trattati in modo lecito e trasparente nei confronti dell'interessato.
- **Limitazione della Finalità:** I dati personali devono essere raccolti per finalità determinate, esplicite e legittime, e non devono essere ulteriormente trattati in modo incompatibile con tali finalità.
- **Minimizzazione dei Dati:** I dati personali devono essere pertinenti e limitati a quanto necessario rispetto alle finalità per le quali sono trattati.
- **Limitazione della Conservazione:** I dati personali devono essere conservati in una forma che consenta l'identificazione degli interessati per un periodo non superiore al necessario rispetto alle finalità per le quali sono trattati.
- **Integrità e Riservatezza:** I dati personali devono essere trattati in modo da garantire un'adeguata sicurezza, compresa la protezione contro il trattamento non autorizzato o illecito e dalla perdita, distruzione o danno accidentale, mediante misure tecniche o organizzative adeguate.

Le app che utilizzano telemetria devono garantire una serie di diritti agli utenti, tra cui:

- **Diritto di Accesso:** Gli utenti hanno il diritto di sapere se i loro dati personali sono stati raccolti e trattati, e di accedere a tali dati.
- **Diritto di Rettifica:** Gli utenti hanno il diritto di correggere dati personali inesatti o incompleti.

- **Diritto alla Cancellazione:** Gli utenti hanno il diritto di ottenere la cancellazione dei propri dati personali in determinate circostanze.
- **Diritto di Limitazione del Trattamento:** Gli utenti possono richiedere la limitazione del trattamento dei loro dati personali in determinate circostanze.
- **Diritto alla Portabilità dei Dati:** Gli utenti hanno il diritto di ricevere i propri dati personali.
- **Diritto di Opposizione:** Gli utenti hanno il diritto di opporsi al trattamento dei loro dati personali per motivi legittimi, inclusa la profilazione.

Nello sviluppo di un'app Android che utilizza telemetria e sia conforme al GDPR, è necessario integrare diverse funzionalità che garantiscano il rispetto dei diritti degli utenti e la protezione dei loro dati personali. Di seguito alcune soluzioni generalmente adottate.

### 5.1.1 Informativa sulla Privacy

L'informativa sulla privacy deve essere di facile comprensione e completa, spiegando quali dati vengono raccolti, come vengono utilizzati, e con chi vengono condivisi. Di seguito si illustra un esempio di implementazione per un activity che mostra l'informativa sulla privacy.

### 5.1.2 Raccolta del Consenso con Google Consent SDK

Google fornisce un SDK per il consenso (Consent SDK) che facilita la raccolta del consenso degli utenti per la pubblicità personalizzata e non personalizzata, conforme al GDPR. Ecco un esempio di come utilizzare questo SDK:

```
ConsentRequestParameters params = new ConsentRequestParameters.Builder().build();
ConsentInformation consentInformation = null;
consentInformation=UserMessagingPlatform.getConsentInformation(context);
consentInformation.requestConsentInfoUpdate(activity, params,
() -> {
    // Consent info updated.
    if (consentInformation.isConsentFormAvailable()) {
        loadForm();
    }
},
formError -> {
    // Handle the error.
}
);

```

### 5.1.3 Utilizzo di Matomo (ex Piwik) per la telemetria self-hosted

Matomo è una piattaforma di analisi open-source che può essere utilizzata per applicazioni di telemetria self-hosted. Questa libreria ti aiuta a inviare dati di analisi dalle app Android alle istanze di Matomo.

### 5.1.4 Metodi per garantire la sicurezza dei dati

È necessario implementare misure tecniche e organizzative adeguate per garantire la sicurezza dei dati personali raccolti. Una delle misure tecniche di maggiore importanza è la crittografia dei dati. Come esempio citiamo la libreria "Signal-Android", dove viene utilizzata la libreria "SignalProtocol" per crittografare i messaggi. Ecco un estratto dal codice di Signal, che è un'app di messaggistica che implementa queste misure per garantire una comunicazione criptata:

```
SignalProtocolStore store = null  
store=new InMemorySignalProtocolStore(identityKeyPair, registrationId);  
SignalProtocolAddress address = new SignalProtocolAddress("+14150001111", 1);  
SessionBuilder sessionBuilder = new SessionBuilder(store, address);  
sessionBuilder.process(preKeyBundle);
```

Oltre alla crittografia dei dati, possono essere implementate anche misure di autenticazione per l'identità degli utenti. Esistono librerie e piattaforme di sviluppo appositamente create a tale scopo. Una di queste è Firebase Authentication, gestito da Google ed atto a semplificare l'implementazione di sistemi di autenticazione. Di seguito alleghiamo un esempio di gestione dell'autenticazione utilizzando Firebase Authentication in un'app Android:

```
FirebaseAuth mAuth = FirebaseAuth.getInstance();  
mAuth.signInWithEmailAndPassword(email, password)  
    .addOnCompleteListener(this, task -> {  
        if (task.isSuccessful()) {  
            // Sign in success  
            FirebaseUser user = mAuth.getCurrentUser();  
        } else {  
            // If sign in fails, display a message to the user.  
        }  
    });
```

## 5.2 Sviluppare un app Android in linea con le normative sulla privacy

Nello sviluppo di un'app Android che utilizza telemetria e sia conforme al GDPR, è necessario integrare diverse funzionalità che garantiscano il rispetto dei diritti degli utenti e la protezione dei loro dati personali, di seguito alcune soluzioni generalmente adottate.

### Informativa sulla Privacy

L'informativa sulla privacy deve essere di facile comprensione e completa, spiegando quali dati vengono raccolti, come vengono utilizzati, e con chi vengono condivisi.

Di seguito si illustra un'esempio di implementazione per un activity che mostra l'informativa sulla privacy.

```
class PrivacyPolicyActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_privacy_policy)  
  
        // Carica l'informativa sulla privacy dal web o dal file locale  
        val privacyPolicyWebView: WebView = findViewById(R.id.privacyPolicyWebView)  
        privacyPolicyWebView.loadUrl("https://www.tuodomnio.com/privacy-policy")  
    }  
}
```

### Raccolta del Consenso con Google Consent SDK

Google fornisce un SDK per il consenso (Consent SDK) che facilita la raccolta del consenso degli utenti per la pubblicità personalizzata e non personalizzata, conforme al GDPR. Ecco un esempio di come utilizzare questo SDK.



```
package com.example.myapplication

import com.google.android.ump.ConsentInformation
import com.google.android.ump.ConsentInformation.OnConsentInfoUpdateFailureListener
import com.google.android.ump.ConsentInformation.OnConsentInfoUpdateSuccessListener
import com.google.android.ump.ConsentRequestParameters
import com.google.android.ump.UserMessagingPlatform

class MainActivity : AppCompatActivity() {
    private lateinit var consentInformation: ConsentInformation

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Create a ConsentRequestParameters object.
        val params = ConsentRequestParameters
            .Builder()
            .build()

        consentInformation = UserMessagingPlatform.getConsentInformation(this)
        consentInformation.requestConsentInfoUpdate(
            this,
            params,
            ConsentInformation.OnConsentInfoUpdateSuccessListener {
                // TODO: Load and show the consent form.
            },
            ConsentInformation.OnConsentInfoUpdateFailureListener {
                requestConsentError ->
                // Consent gathering failed.
                Log.w(TAG, "${requestConsentError.errorCode}: ${requestConsentError.message}")
            }
        )
    }
}
```

Il modulo di consenso viene caricato e, se disponibile, viene mostrato all’utente per raccogliere il consenso.

Il modulo di consenso contiene tutte le informazioni necessarie per aiutare l’utente a comprendere come i suoi dati verranno utilizzati.

Quello di Google Consent SDK non è l’unico metodo possibile per la raccolta del consenso, ma è un esempio esplicativo.

fonte: [User Messaging Platform \(UMP\) SDK - Google Developers](#)

## Utilizzo di Matomo (ex Piwik) per la telemetria self-hosted

Matomo è una piattaforma di analisi open-source che può essere utilizzata per applicazioni di telemetria self-hosted.

```
// Configuration of Matomo Tracker in an Android app
import org.matomo.sdk.Matomo
import org.matomo.sdk.Tracker
import org.matomo.sdk.TrackerBuilder
import android.content.Context

class MatomoAnalytics(context: Context) {
    private val tracker: Tracker

    init {
        // Initialization of the Matomo tracker with the server URL and site ID
        tracker = TrackerBuilder.createDefault("https://your-matomo-server.com/matomo.php",
            .build(Matomo.getInstance(context))
        tracker.optOut = true // Disable data collection by default
    }

    // Enable data collection after user consent
    fun onUserConsentGranted() {
        tracker.optOut = false
    }

    // Disable data collection upon user request
    fun onUserConsentRevoked() {
        tracker.optOut = true
    }

    fun getTracker(): Tracker {
        return tracker
    }
}
```

Questa libreria ti aiuta a inviare dati di analisi dalle app Android alle istanze di Matomo.

fonte: [Matomo Mobile SDK for Android](#)

## Metodi per garantire la sicurezza dei dati

Come già affermato, è necessario implementare misure tecniche e organizzative adeguate per garantire la sicurezza dei dati personali raccolti.

Una delle misure tecniche di maggiore importanza è la crittografia dei dati. Come esempio citiamo la libreria "Signal-Android", dove viene utilizzata la libreria "SignalProtocol" per crittografare i messaggi.

Ecco un estratto dal codice di Signal, che è un app di messaggistica che implementa queste misure per garantire una comunicazione criptata :

```
public class SignalServiceMessageSender {
    public void sendMessage(SignalServiceMessage message) throws IOException {
        SignalServiceCipher cipher = new SignalServiceCipher(localAddress, remoteAddress, se
        SignalServiceCipherMessage encryptedMessage = cipher.encrypt(message.getMessage());

        // Send the encrypted message
        service.sendMessage(encryptedMessage);
    }
}
```

fonte: [Signal Android](#)

Oltre alla crittografia dei dati possono essere implementate anche misure di autenticazione per l'identità degli utenti.

Esistono librerie e piattaforme di sviluppo appositamente create a tale scopo. Una di queste è Firebase Authentication, gestito da Google ed atto a semplificare l'implementazione di sistemi di autenticazione.

Di seguito alleghiamo un esempio di gestione dell'autenticazione utilizzando Firebase Authentication in un'app Android.

```
FirebaseAuth auth = FirebaseAuth.getInstance();
auth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, task -> {
    if (task.isSuccessful()) {
        FirebaseUser user = auth.getCurrentUser();
        // User is signed in
    } else {
        // Authentication failed
    }
});
```

fonte: [Firebase Auth](#)

## 6 Applicazione sviluppata: gioco del memory

### 6.1 Descrizione e Struttura dell'Applicazione

L'applicazione che abbiamo deciso di sviluppare è un semplice gioco di matching Pokémon cards. L'applicazione offre due modalità di gioco: single player e multiplayer.

Nella modalità single player, il gioco è diviso in tre round, rappresentati dalle classi `Round1Fragment.kt`, `Round2Fragment.kt`, e `Round3Fragment.kt`. Nella modalità multiplayer, il gioco è gestito dalla classe `MultiplayerFragment.kt`. Entrambi i tipi di gioco derivano dalla classe astratta `BaseRoundFragment.kt`, che gestisce i frammenti comuni.

La logica del gioco per il singolo giocatore è implementata nella classe `SingleGameLogic`, mentre la logica per il multiplayer è implementata nella classe `MultiplayerGameLogic`.

La struttura del progetto è la seguente:

`app/src/main/java/com/example/matchgame/`

- `ui/`
  - `BaseRoundFragment.kt` - Classe astratta che gestisce i frammenti
  - `Round1Fragment.kt` - Primo round
  - `Round2Fragment.kt` - Secondo round
  - `Round3Fragment.kt` - Terzo round
  - `MultiplayerFragment.kt` - Modalità multiplayer
  - Altri frammenti: `HomeFragment.kt`, `YouLoseFragment.kt`, `YouWinFragment.kt`,  
`Player1WinFragment.kt`, `Player2WinFragment.kt`, `AboutFragment.kt`,  
`DialogFragment.kt`, `InfoFragment.kt`
- `logic/`
  - `IGameLogic.kt` - Interfaccia della logica del gioco, contiene la definizione dei metodi comuni
  - `SingleGameLogic.kt` - Logica di gioco per la modalità single player
  - `MultiplayerGameLogic.kt` - Logica di gioco per la modalità multiplayer

- **models/**
  - `MemoryCard.kt` - Rappresenta una singola carta nel gioco di memoria
- **adapter/**
  - `CardAdapter.kt` - Gestisce la logica per la visualizzazione delle carte nel `RecyclerView`
- **telemetry/**
  - `DataCollector.kt` - Gestisce la trasmissione dei dati alla console Firebase per la telemetria
- **MainActivity.kt** - L'attività principale che funge da punto di ingresso dell'applicazione

### 6.1.1 Screenshot dell'Applicazione

La schermata principale dell'applicazione permette all'utente di scegliere tra la modalità single player e multiplayer. Come mostrato nella Figura 1, i due pulsanti principali consentono di selezionare la modalità di gioco desiderata.

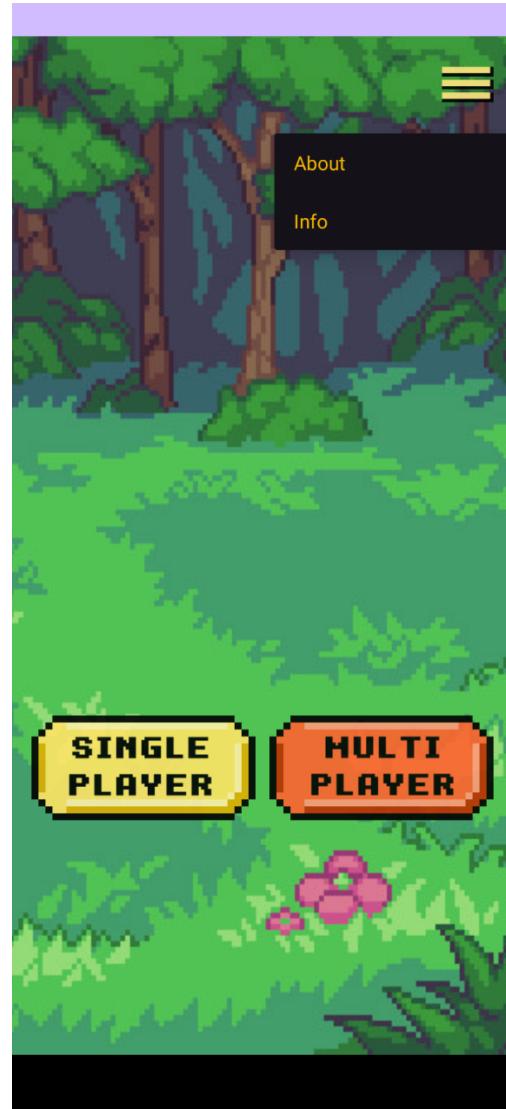


Figura 1: Schermata principale dell'applicazione.

Durante il gioco, se l'utente tenta di tornare indietro, viene visualizzato un menu che permette di scegliere se continuare o uscire dal gioco. La Figura 2 ne mostra un esempio.

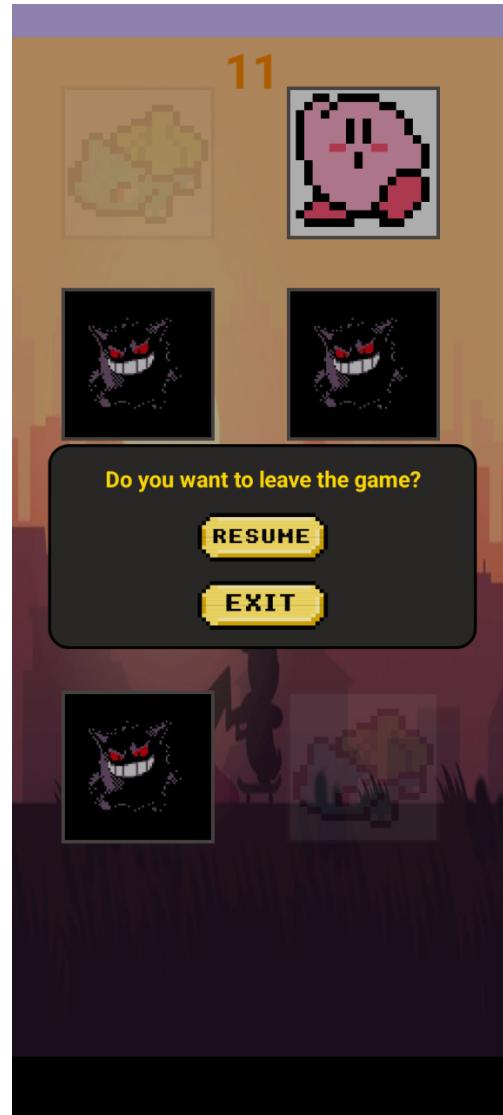


Figura 2: Menu che si attiva quando l'utente tenta di tornare indietro durante i round del gioco.

Nella modalità single player, il gioco è suddiviso in tre round. Le Figure 3, 4 e 5 mostrano rispettivamente il primo, secondo e terzo round. L'utente deve abbinare le carte dei pokémon uguali. Ogni livello aumenta di difficoltà con più carte da abbinare e il timer più restrittivo.

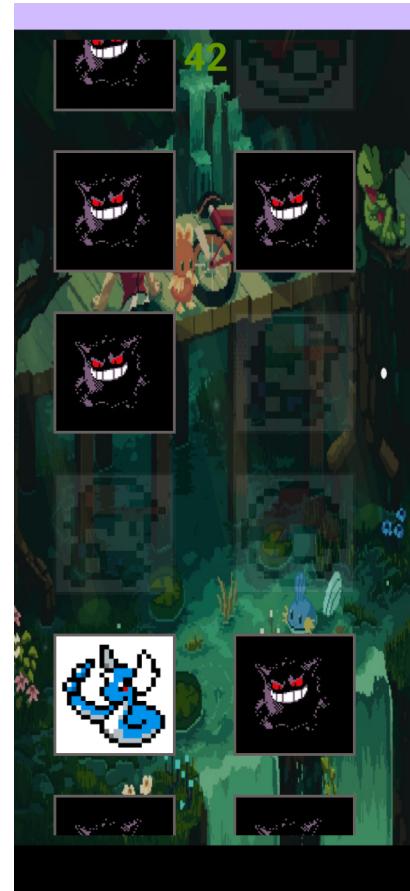
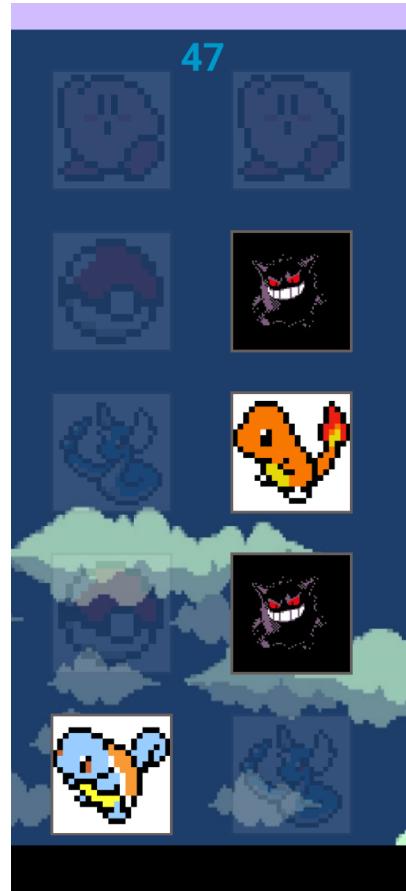
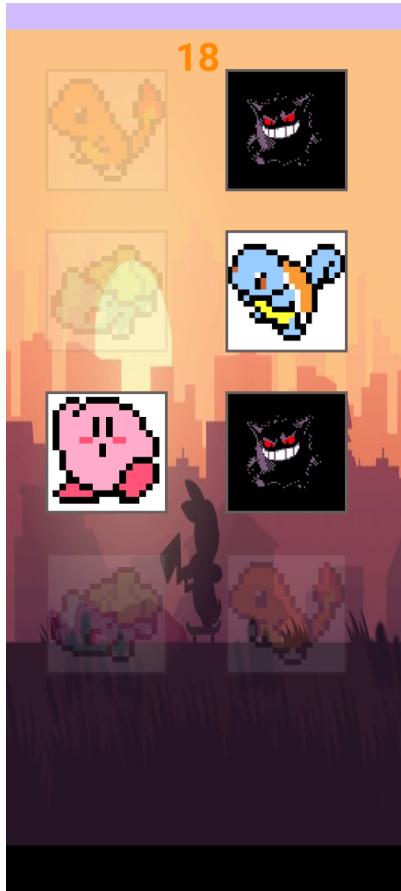


Figura 3: Schermata del primo round nella modalità single player.

Figura 4: Schermata del secondo round nella modalità single player.

Figura 5: Schermata del terzo round nella modalità single player.

Nella modalità multiplayer, i due giocatori si alternano nel tentativo di abbinare le carte. Le Figure 6 e 7 mostrano le schermate per il giocatore 1 e il giocatore 2, rispettivamente. Le carte per il giocatore 1 sono evidenziate in rosso, mentre quelle per il giocatore 2 sono evidenziate in blu. In questa modalità lo scopo diventa quello di realizzare uno score finale superiore all'avversario (il giocatore riceve 1 punto per abbinamento, 0 altrimenti).

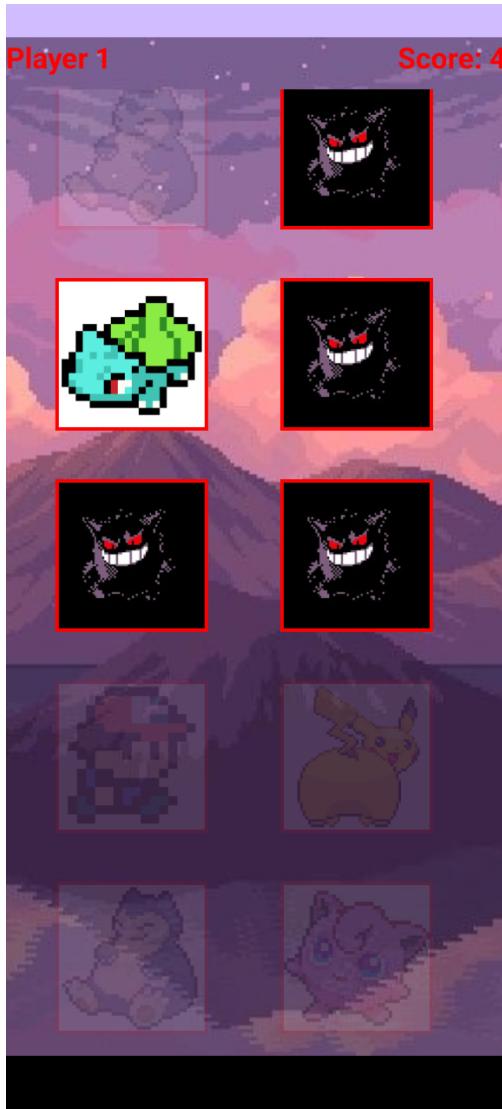


Figura 6: Schermata della modalità multiplayer per il giocatore 1. Le carte sono evidenziate in rosso.

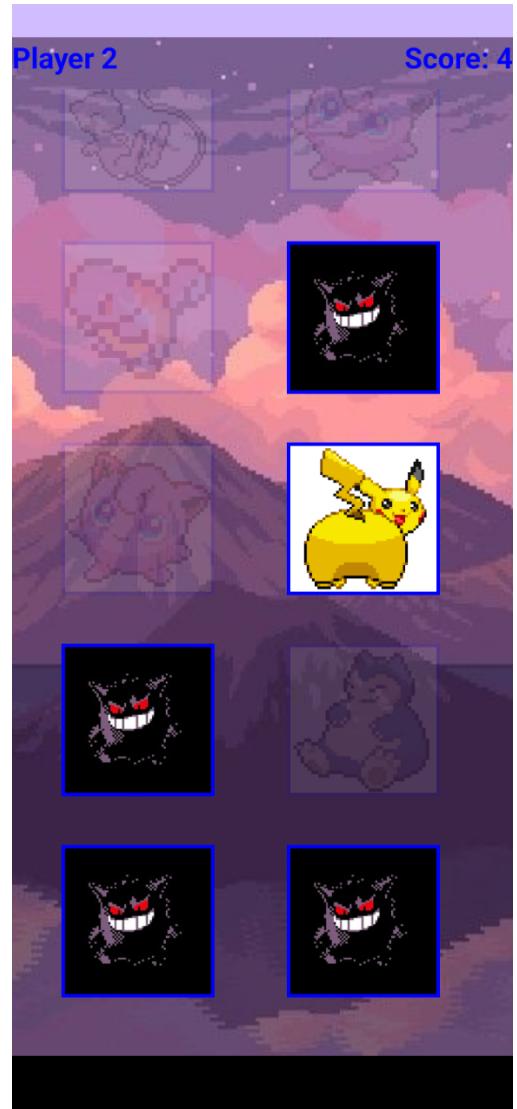


Figura 7: Schermata della modalità multiplayer per il giocatore 2. Le carte sono evidenziate in blu.

L'applicazione supporta anche la modalità landscape, che offre un'esperienza di gioco ottimizzata per schermi più larghi. La Figura 8 mostra la schermata principale in modalità landscape.



Figura 8: Schermata principale dell'applicazione in modalità landscape.

Il menu in modalità landscape, visibile nella Figura 9, consente all'utente di scegliere se continuare o uscire dal gioco, offrendo un'esperienza utente coerente anche in questa modalità.



Figura 9: Menu in modalità landscape.

Nella modalità single player landscape, il gioco è suddiviso in tre round. Le Figure 10, 11 e 12 mostrano rispettivamente il primo, secondo e terzo round in modalità landscape. La modalità di gioco rimane coerente anche per il multiplayer.

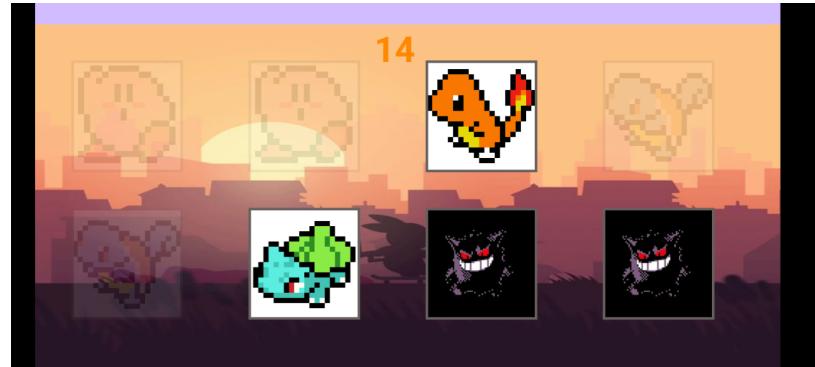


Figura 10: Schermata del primo round nella modalità single player in modalità landscape.

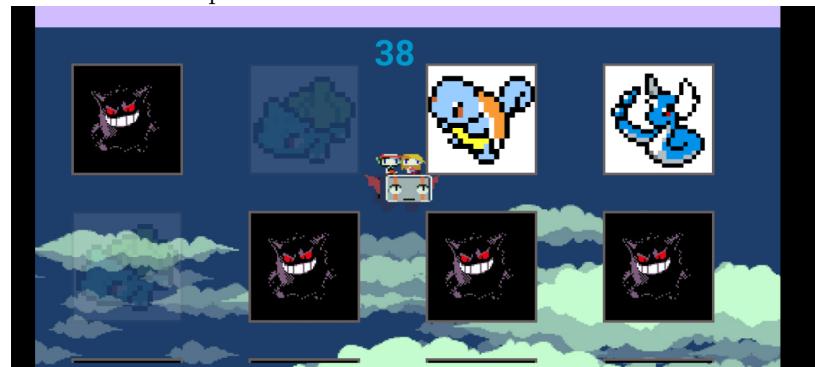


Figura 11: Schermata del secondo round nella modalità single player in modalità landscape.

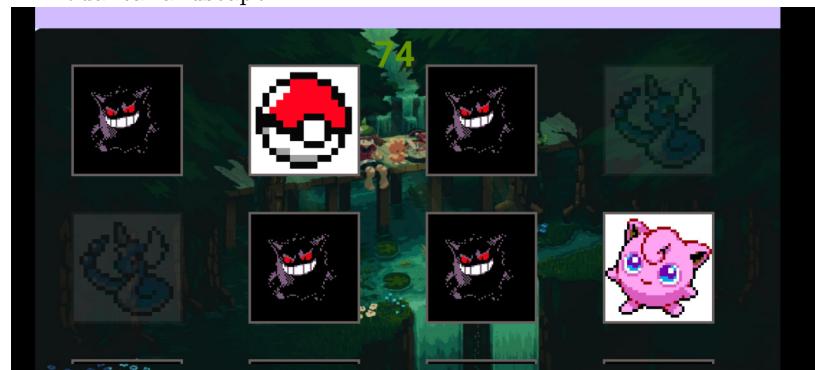


Figura 12: Schermata del terzo round nella modalità single player in modalità landscape.

## 6.2 Integrazione di Firebase Analytics

Per la raccolta e trasmissione dei dati di telemetria, è stato utilizzato Firebase. La classe `DataCollector.kt`, che gestisce l'invio dei dati alla console di Firebase, invia i seguenti dati:

- **Tempo di avvio dell'applicazione:** evento `app_launch_time`, che misura il tempo impiegato dall'applicazione per avviarsi. È importante per valutare le prestazioni iniziali e l'efficienza dell'app.
- **Tempo di click del pulsante per iniziare la partita** (single player o multiplayer): evento `click_play_button`, che traccia il momento in cui l'utente avvia una partita, sia in modalità single player che multiplayer. Utile per comprendere le preferenze degli utenti riguardo le modalità di gioco.
- **Tempo di completamento del livello:** evento `level_completion_time` o `multiplayer_level_completion_time`, che registra il tempo impiegato dall'utente per completare un livello, sia in modalità single player (i tre rounds) che multiplayer. Permette di identificare eventuali ostacoli del completamento di un livello da parte degli utenti. Inoltre offre un feedback della difficoltà percepita dagli utenti.
- **Tempo medio per girare le carte:** evento `round1_average_time_between_clicks`, `round2_average_time_between_clicks`, `round3_average_time_between_clicks`, `multiplayer_average_time_between_clicks`, che monitora il tempo medio per girare le carte nelle diverse modalità del gioco. Insieme ai dati precedenti, permette di comprendere il livello di coinvolgimento dell'utente.
- **Numero di carte girate per round:** evento `round1_button_clicks`, `round2_button_clicks`, `round3_button_clicks`, `multiplayer_button_clicks`, che conta il numero di volte in cui una carta viene girata in ogni round. Utile per comprendere la difficoltà percepita dai giocatori e per effettuare regolazioni ai livelli di difficoltà se necessario.
- **Utilizzo della batteria:** evento `battery_usage`, che registra la percentuale della batteria all'inizio e alla fine del gioco. Aiuta a valutare

l'efficienza energetica dell'applicazione e ad evidenziare alcune criticità in specifici dispositivi.

- **Tipo di OS e tipo di dispositivo** (user properties), che raccoglie informazioni sul sistema operativo e sul tipo di dispositivo utilizzato dagli utenti. Utile per identificare specifici dispositivi associandoli con eventuali criticità/altri dati.
- **Utilizzo della RAM:** evento `memory_usage`, che monitora la quantità di RAM totale e disponibile durante il gioco. Importante per identificare e risolvere problemi di performance legati alla memoria.
- **Durata totale del gioco:** evento `total_game_duration`, che misura la durata complessiva di una sessione di gioco. Utile per comprendere quanto tempo gli utenti spendono nell'applicazione.
- **Fine del gioco con esito** (perdita o vittoria): evento `game_end`, che registra se il giocatore ha vinto o perso alla fine di un gioco. Utile per analizzare il tasso di successo dei giocatori e per bilanciare meglio la difficoltà del gioco.

Tra questi abbiamo selezionato come eventi chiave: `app_launch_time`, `click_play_button`, `level_completion_time`, `memory_usage`, `multiplayer_level_completion_time`, `total_game_duration`, `game_end`, `battery_usage`. Abbiamo selezionato questi eventi chiave perché sono fondamentali per ottenere una visione chiara e significativa sull'utilizzo dell'app, il comportamento degli utenti e le prestazioni dell'app stessa.

Oltre alla raccolta di questi dati di telemetria, l'applicazione utilizza Firebase Crashlytics per il tracciamento degli errori. Crashlytics raccoglie informazioni dettagliate su eventuali crash dell'applicazione, compresi i log degli errori e le circostanze che li hanno causati. Questo permette di:

- **Identificare e risolvere rapidamente i problemi:** Le informazioni dettagliate sui crash aiutano a individuare e correggere i bug in modo tempestivo.
- **Migliorare la qualità dell'app:** Monitorando costantemente i crash, è possibile rilevare tendenze e problemi ricorrenti, permettendo di implementare soluzioni più robuste.
- **Ottimizzare l'esperienza utente:** Riducendo il numero di crash e migliorando la stabilità dell'app, si garantisce un'esperienza utente più fluida.

## 6.3 Dashboard di Firebase

Di seguito sono riportati alcune immagini della console di Firebase che mostrano i dati raccolti.

### 6.3.1 Firebase DebugView

La Figura 13 mostra la DebugView di Firebase, che visualizza i dati raccolti in tempo reale durante il debug dell'applicazione. Questa schermata permette di monitorare i tempi di completamento dei livelli, l'utilizzo della batteria, RAM, OS e altri eventi significativi mentre vengono generati.

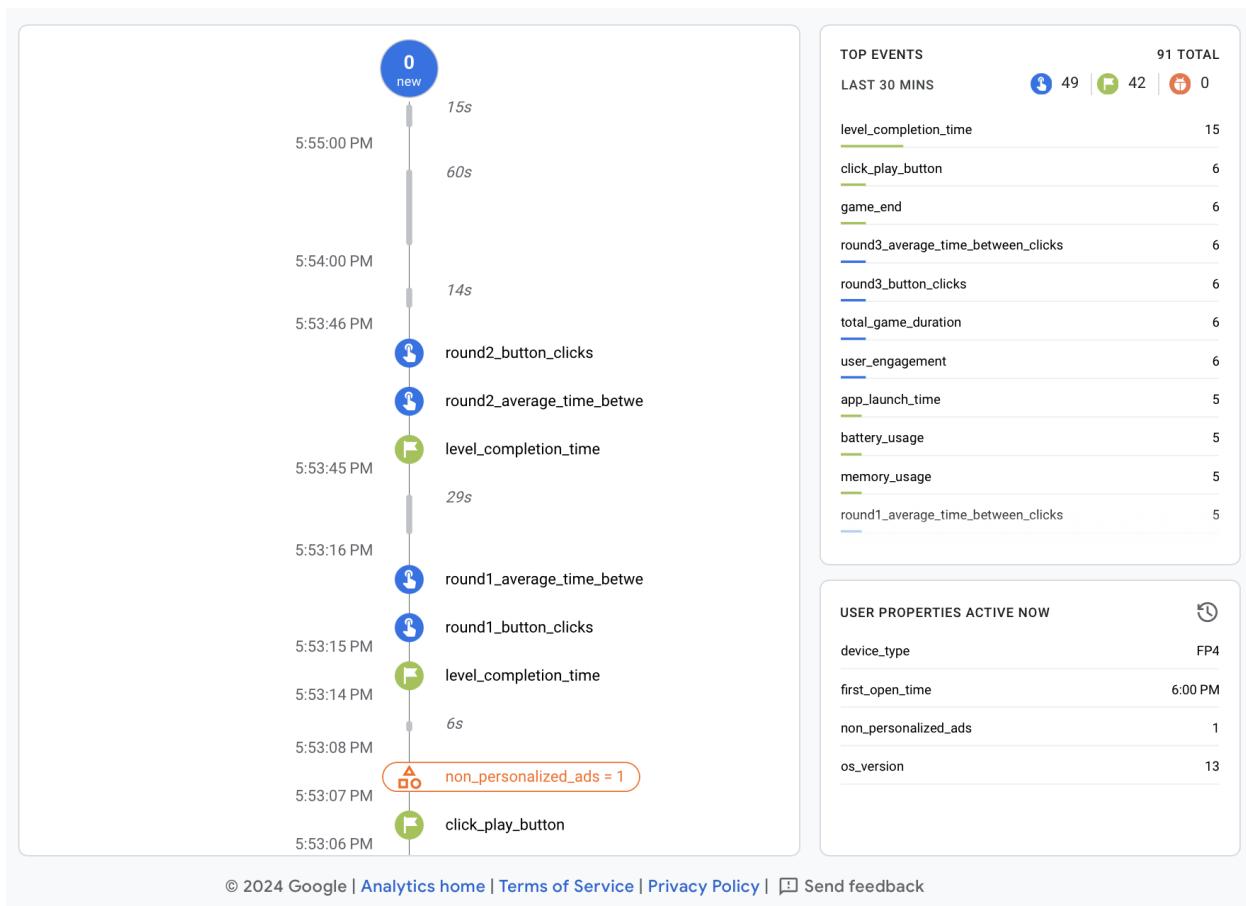


Figura 13: DebugView di Firebase che mostra i dati in tempo reale.

Nella parte sinistra si può vedere una linea temporale degli eventi registrati dall'applicazione. Alcuni degli eventi chiave visualizzati includono: `click_play_button`, `level_completion_time`, `round_button_clicks`,

`round_average_time_between_clicks`, `level_completion_time` e user properties (come device type e OS version).

Questi eventi sono essenziali per comprendere il comportamento dell'utente durante il gioco. Ad esempio:

- **click\_play\_button**: Ci aiuta a determinare quando e con quale frequenza gli utenti iniziano una nuova partita, inoltre si può stimare la scelta tra single player e multiplayer.
- **level\_completion\_time**: Permette di analizzare la durata necessaria per completare un livello, identificando eventuali difficoltà riscontrate dagli utenti.
- **Tempo medio tra i clic**: Un indicatore importante della velocità di reazione e dell'interazione dell'utente con il gioco.

### 6.3.2 Firebase Realtime Overview

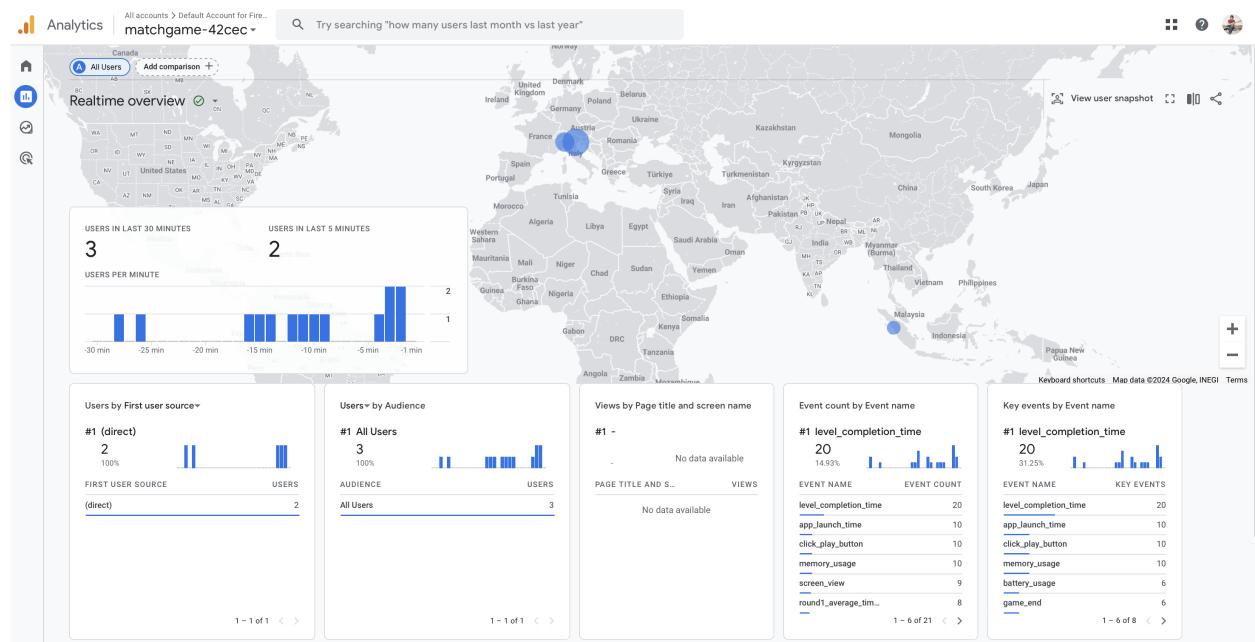


Figura 14: Panoramica in tempo reale su Firebase Analytics.

La Figura 14 mostra la panoramica in tempo reale su Firebase Analytics. Questa schermata fornisce una visualizzazione degli utenti attivi negli ultimi 30 minuti, la loro provenienza geografica, e gli eventi chiave registrati. Mostra anche le fonti di primo accesso degli utenti, la loro distribuzione per audience,

e una suddivisione per eventi, come l'utilizzo della batteria, il tempo di avvio dell'app, l'utilizzo della memoria, e altri. E' utile per monitorare l'attività degli utenti in tempo reale e ottenere una visione immediata del comportamento degli utenti e delle prestazioni dell'app.

Le due sezioni centrali mostrano le origini degli utenti e l'audience:

- **Users by First User Source:** Indica la fonte dalla quale provengono gli utenti. In questo caso, il 100% degli utenti proviene da accessi diretti.
- **Users by Audience:** Mostra che tutti e 3 gli utenti appartengono all'audience generale.

Le due sezioni di destra mostrano informazioni sugli eventi chiave:

- **Event Count by Event Name:** Mostra il conteggio degli eventi per nome. L'evento più frequente è `level_completion_time` con 20 occorrenze.
- **Key Events by Event Name:** Elenca gli eventi chiave prescelti.

Per dimostrare il funzionamento di raccolta dei dati per la telemetria, includiamo alcuni pdf reports di Firebase. Di seguito sono riportati esempi di Events Report e Analytics Dashboard di Firebase.

### 6.3.3 Events Report

La figura 15 mostra un estratto del report sullo storico degli eventi che abbiamo deciso di tracciare nell'app, inclusi il coinvolgimento degli utenti, la durata totale del gioco, l'inizio delle sessioni e i tempo di completamento di un livello.

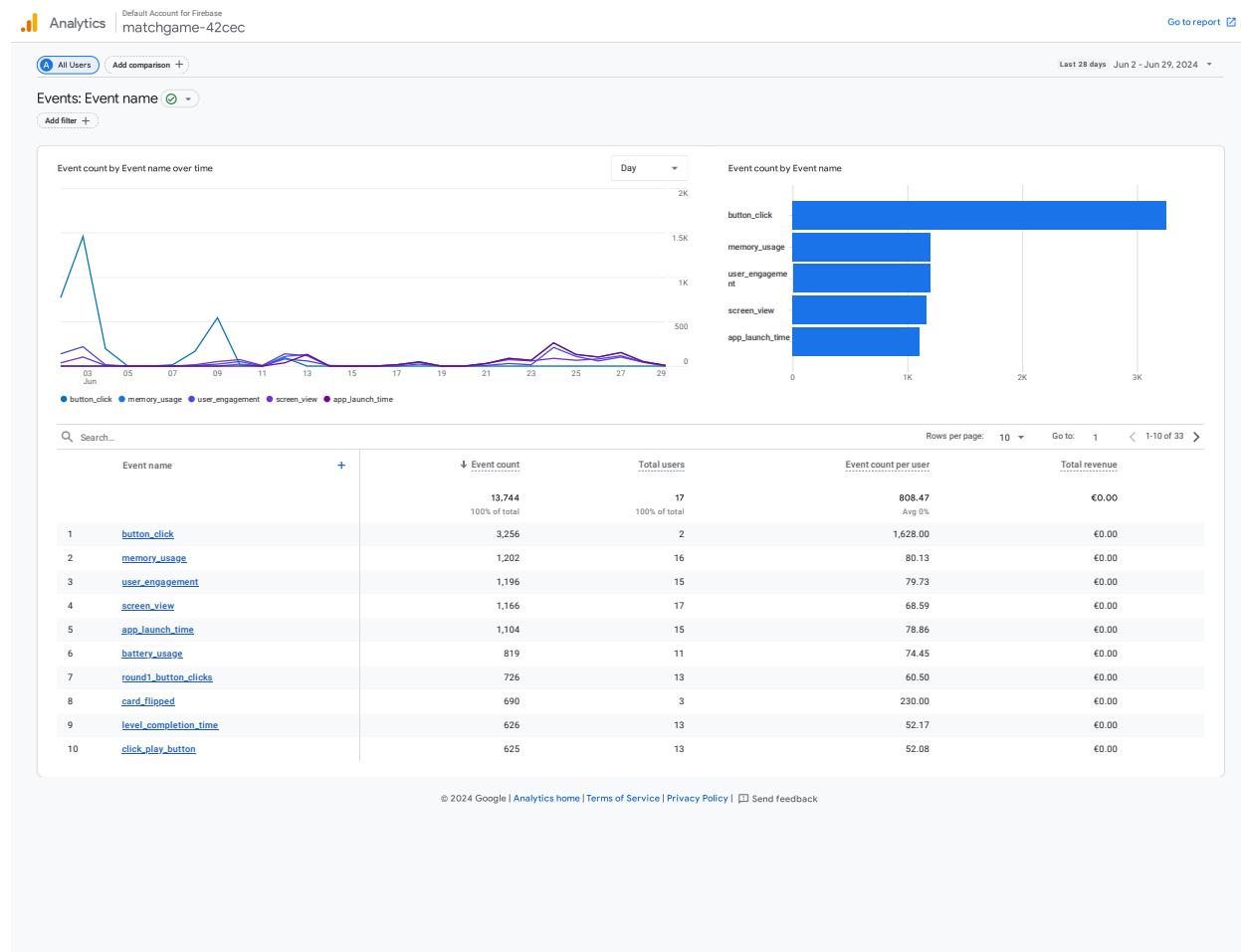


Figura 15: Events Report

Questo report è utile per monitorare l'attività complessiva degli utenti nell'applicazione. Include diverse sezioni chiave che forniscono una visione complessiva dell'interazione degli utenti con l'app. Può essere customizzata, includendo grafici dettagliati su esigenze specifiche, tra i quali eventi chiave, conteggio degli eventi, dati degli utenti in tempo reale, modelli di dispositivo, distribuzione geografica degli utenti, lingue degli utenti, tempo di coinvolgimento medio, ritenzione degli utenti.

### 6.3.4 Firebase Overview Report

La figura 16 mostra il Firebase Overview, che fornisce una panoramica completa delle metriche chiave relative all'utilizzo dell'applicazione, agli eventi e alla demografia degli utenti.

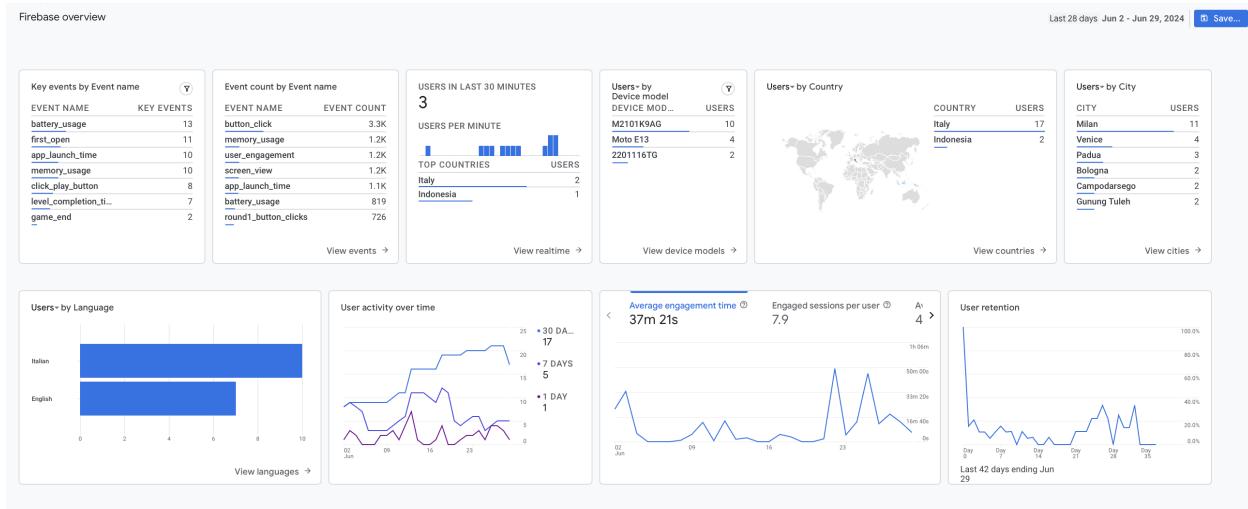


Figura 16: Firebase Overview Report

Questo report (customizzabile) è utile per monitorare l'attività complessiva degli utenti nell'applicazione e include diverse sezioni chiave che forniscono una visione complessiva dell'interazione degli utenti con l'app.

## Riferimenti bibliografici

- [1] Zetter, K. (2014). *Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon*. Crown.
- [2] Cadwalladr, C., & Graham-Harrison, E. (2018). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*. Retrieved from <https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election>
- [3] Barnes, J., Bratus, S., Costello, J., & Shubina, A. (2014). Hacking the Iridium satellite network. In *Proceedings of the 7th USENIX Workshop on Offensive Technologies (WOOT 13)*.
- [4] OpenTelemetry Documentation. (n.d.). Retrieved from <https://opentelemetry.io/docs/>
- [5] Alibaba Cloud. (n.d.). Use OpenTelemetry to report the trace data of Android applications. Retrieved from <https://www.alibabacloud.com/help/en/opentelemetry/user-guide/use-managed-service-for-opentelemetry-to-report-the-trace-data-of-android-applications>
- [6] Firebase Documentation. (n.d.). Firebase Analytics. Retrieved from <https://firebase.google.com/docs/analytics>
- [7] Google Developers. (n.d.). User Messaging Platform (UMP) SDK. Retrieved from <https://developers.google.com/interactive-media-ads/docs/sdk/android/client-side/privacy#kotlin>
- [8] Matomo. (n.d.). Matomo Mobile SDK for Android. Retrieved from <https://github.com/matomo-org/matomo-sdk-android>
- [9] Regolamento (UE) 2016/679 del Parlamento Europeo e del Consiglio, del 27 aprile 2016, relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali e alla libera circolazione di tali dati (Regolamento generale sulla protezione dei dati). Retrieved from <https://eur-lex.europa.eu/legal-content/IT/TXT/?uri=CELEX%3A32016R0679>

- [10] Open Whisper Systems. (n.d.). Signal Android. Retrieved from <https://github.com/signalapp/Signal-Android>
- [11] Firebase Documentation. (n.d.). Firebase Authentication. Retrieved from <https://firebase.google.com/docs/auth>
- [12] Dynatrace. (n.d.). What is Dynatrace? Retrieved from <https://www.dynatrace.com/>
- [13] Datadog. (n.d.). Datadog Documentation. Retrieved from <https://docs.datadoghq.com/>
- [14] AppDynamics. (n.d.). AppDynamics Application Performance Monitoring. Retrieved from <https://www.appdynamics.com/>
- [15] Dotcom-Monitor. (n.d.). Dotcom-Monitor: Website & Performance Monitoring. Retrieved from <https://www.dotcom-monitor.com/>
- [16] Instana. (n.d.). Instana Documentation. Retrieved from <https://www.instana.com/docs/>
- [17] LogicMonitor. (n.d.). LogicMonitor Documentation. Retrieved from <https://www.logicmonitor.com/support>
- [18] Scout APM. (n.d.). Scout Documentation. Retrieved from <https://docs.scoutapm.com/>
- [19] Raygun. (n.d.). Raygun Documentation. Retrieved from <https://raygun.com/documentation>