ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica
Attività progettuale in Sistemi Mobili M

Android Things e Action on Google Smart Home

Applicazione Android per la gestione di una smart home

Autore

Silvia Brescia

Professore

Paolo Bellavista

Anno accademico 2019/2020

Indice generale

Capitolo 1: Introduzione

2.1 Obiettivi

Le tecnologie smart sono in grado di automatizzare una serie di attività e di processi nelle abitazioni, allo scopo di ottimizzare i consumi e garantire maggiore comodità. Fanno riferimento all'IoT e indicano tutti i prodotti che utilizzano la connessione Internet e le app per smartphone per offrire nuove funzioni connesse anche a prodotti tradizionali, ad esempio i termostati ambiente. le tecnologie smart home sono in grado di automatizzare (e ottimizzare) una serie di attività e di processi nelle abitazioni, con lo scopo di ottimizzare i consumi e garantire maggiore comodità per chi vi abita. Fra i vantaggi di avere una casa intelligente e connessa, al primo posto figura il risparmio energetico, che si trasforma in vantaggio economico. Il controllo automatico e personalizzato del funzionamento di luci, elettrodomestici, condizionatori, termosifoni e altro, in base alle proprie abitudini e alle condizioni ambientali esterne, permette di evitare gli sprechi, utilizzando i vari dispositivi solo in caso di reale bisogno. Smart home significa anche monitorare i consumi in maniera puntuale, tenendo sotto controllo la bolletta energetica. Il secondo vantaggio è la comodità. Automatizzare l'accensione e la regolazione di luci, tapparelle, riscaldamento e refrigerazione solleva l'utente dall'incombenza di doversene ricordare. In alternativa all'automazione si può mantenere il controllo umano, ma affidandolo ai comandi vocali, molto più immediati e comodi di quelli manuali. Vi è poi da sottolineare la possibilità del controllo da remoto di molti funzioni domestiche, che aggiunge ulteriore comodità e praticità. Si pensi, ad esempio, all'attivazione dei sistemi di climatizzazione o di alcuni elettrodomestici, eseguibile ad esempio dall'ufficio o dalla propria auto mentre si è in viaggio. Smart home vuol dire anche assistenza; le tecnologie possono essere impiegate per migliorare la vita o assicurare assistenza ad anziani e a persone con disabilità, soprattutto motorie. Infine, ultima ma non ultima, la sicurezza. L'integrazione dei tradizionali sistemi di allarme con un sistema intelligente e connesso ne aumenta sensibilmente il grado di affidabilità, consentendo al proprietario dell'appartamento di tenerlo sotto controllo anche quando si trova molto distante.

Queste tecnologie sono sempre più diffuse oggi e per questo motivo in questa attività progettuale ho voluto approfondire questo argomento studiando in particolare delle soluzione integrate con Android. In particolare, verranno prese in considerazione in questo elaborato due tecnologie: **Android Things**, un sistema operativo rilasciato da Google per IoT e dispositivi integrati, e **Action on Google Smart Home**, piattaforma utile per controllare i dispositivi connessi disponibili in commercio tramite l'app Google Home e Google Assistant.

L'obiettivo dell'attività progettuale è studiare queste due piattaforme e realizzare un'applicazione per Android che si interfacci con la tecnologia Action on Google Smart Home e che permetta all'utente di gestire una casa connessa.

Capitolo 2: Tecnologie usate

2.1 Android Things

2.1.1 Overview

Android Things è un sistema operativo rilasciato da Google per IoT e dispositivi integrati. È basato su Android, che a sua volta utilizza il kernel Linux. Pertanto supporta il multitasking e la memoria virtuale. È progettato per adattarsi a dispositivi con un footprint di memoria limitato, sebbene 512 MB sia il requisito minimo di RAM. Android Things consente di sperimentare la creazione di dispositivi su una piattaforma affidabile, senza una precedente conoscenza della progettazione di sistemi integrati:

- Usa Android SDK e Android Studio
- Accede all'hardware come display e telecamere in modo nativo tramite il framework Android
- Connette le tue app ai servizi di Google
- Integra periferiche aggiuntive tramite le API di I/O periferiche (GPIO, I2C, SPI, UART, PWM)
- Usa ***Android Things Console*** per inviare funzionalità over-the-air e aggiornamenti di sicurezza

Figura 1- Logo Android Things

Android Things consente di creare app su piattaforme hardware come Raspberry Pi 3. Il Board Support Package (BSP) è gestito da Google, quindi non è necessario lo sviluppo di kernel o firmware. Le immagini del software vengono create e distribuite ai dispositivi tramite Android Things Console. Ciò offre una piattaforma affidabile su cui sviluppare aggiornamenti e correzioni standard di Google. Android Things può essere eseguito su piattaforme a 32 e 64 bit, sia su processori ARM che basati su x86. Il design modulare dei SoM consente agli sviluppatori di utilizzarli su una scheda di sviluppo durante la prototipazione e di riutilizzarli su qualsiasi scheda di supporto personalizzata, progettata per i prodotti finali.

Al di sopra del sistema operativo, utilizza ***Weave*** per consentire a diversi dispositivi di comunicare usando schemi condivisi. Android Things include la connettività Weave integrata. Weave può essere utilizzato anche senza Android Things.

2.1.2 SDK

Android Things estende il framework Android di base con API aggiuntive fornite dalla libreria *Things Support Library*, che consente di integrarsi con nuovi tipi di hardware non presenti sui dispositivi mobili.

Lo sviluppo di app per dispositivi integrati è diverso da quello mobile:

- Accesso più flessibile a periferiche e driver hardware rispetto ai dispositivi mobili
- Le app di sistema non sono presenti per ottimizzare i requisiti di avvio e archiviazione
- Le app vengono avviate automaticamente all'avvio per immergere gli utenti nell'esperienza dell'app.
- I dispositivi espongono agli utenti solo una app, anziché più come per i dispositivi mobili.

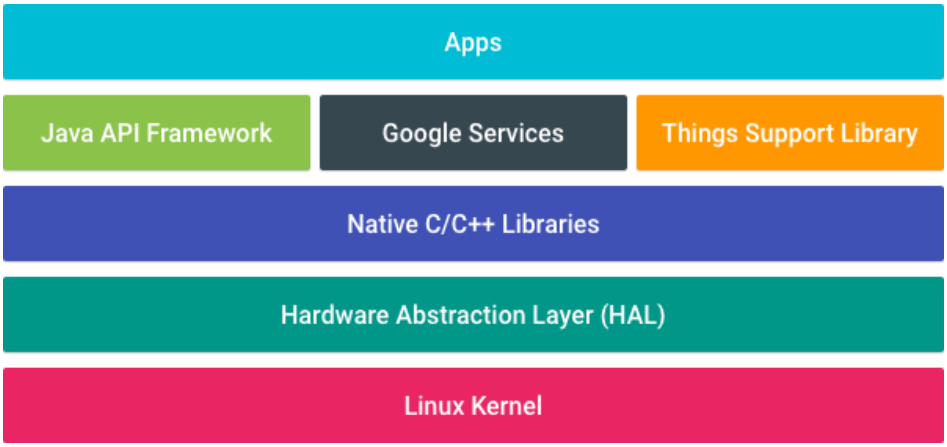


Figura 2- https://developer.android.com/things/images/platform-architecture.png

Java API Framework

- Il display diventa opzionale (non ci sono status bar o navigation bar)
- Supporto alla home activity: avvio automatico al boot.

Google Services

- Supporto a una sottoclasse dei servizi di Google per Android

Things Support Library

- Le API delle periferiche di I/O permettono alle app di interagire con sensori e attuatori, usando protocolli e interfacce industriali standard.
- User Driver API: estende i servizi esistenti di Android e permette alle app di iniettare eventi hardware all'interno del framework a cui altre app possono accedere tramite gli standard Android

2.1.3 Differnze tra Android Things e Android Framework

Android Things si basa sulla piattaforma Android ed è ottimizzato per gli embedded device. Insieme a nuove funzionalità e capacità, Android Things include una varietà di differenze di sistema e API rispetto ad Android.

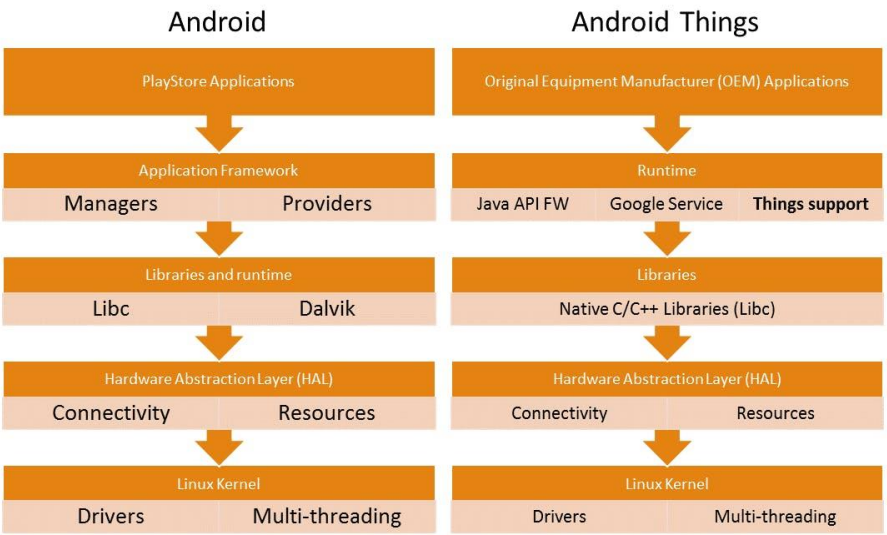


Figura 3 - Comparing the architectures of Android and Android Things. Source: Mahmoudi, 2017.

ANDROID THINGS	ANDROID FRAMEWORK
La raccolta di app installate viene fixata dallo sviluppatore/produttore del dispositivo. Queste vengono modificate tramite gli aggiornamenti OTA, non gestiti dall'utente finale.	Gli aggiornamenti sono gestiti dall'utente finale
Ottimizzato per l'uso con una sola app. Una app viene avviata automaticamente all'avvio del sistema.	Possibilità di utilizzare diverse app.
User interface grafica opzionale (non tutti i dispositivi hanno un display)	User interface grafica persente
Ottimizzato per embedded devices	Ottimizzato per dispositivi come tablet e cellulari
Non include la suite standard di app di sistema. Evita di utilizzare API come CalendearContract, ContactsContract, DocumentsContracts, DownloadManager, MediaStore, Settings, Telephony, UserDictionary, VocemailCotract.	Include la suite standard di app di sistema. Usa API come CalendearContract, ContactsContract, DocumentsContracts, DownloadManager, MediaStore, Settings, Telephony, UserDictionary, VocemailCotract.
Si concedono le autorizzazioni pericolose per tutte le app come parte del processo di creazione del build. Questa attività si può ignorare durante lo sviluppo, ma non sui prodotti reali; gli utenti finali non possono modificare queste autorizzazioni.	Tutte le tutte le autorizzazioni (incluse quelle pericolose) vengono concesse al momento dell'installazione. Questo vale sia per le nuove installazioni di app e sia per gli elementi <uses-permission> aggiornati nelle app esistenti.
È compatibile con Android NDK per includere codice C/C ++ nell'app. Tuttavia, poiché i dispositivi Android Things sono generalmente limitati in memoria, la piattaforma richiede alle app di mantenere le librerie native all'interno dell'APK in fase di runtime utilizzando l'attributo manifest extractNativeLibs.	Usa Android NDK per compilare codice C/C ++ in una libreria nativa e includerla all'interno dell'APK.

La tabella seguente descrive il set di funzionalità Android attualmente non supportate dai dispositivi Android Things e le API del framework interessato:

FEATURE	API
System UI (status bar, navigation buttons, quick settings)	NotificationManager KeyguardManager WallpaperManager
`VoiceInteractionService`	SpeechRecognizer
`android.hardware.fingerprint`	FingerprintManager
`android.hardware.nfc`	NfcManager
`android.hardware.telephony`	SmsManager TelephonyManager
`android.hardware.usb.accessory`	UsbAccessory

android.hardware.wifi.aware	WifiAwareManager
android.software.app_widgets	AppWidgetManager
android.software.autofill	AutofillManager
android.software.backup	BackupManager
android.software.companion_device_setup	CompanionDeviceManager
android.software.picture_in_picture	Activity Picture-in-picture
android.software.print	PrintManager
android.software.sip	SipManager

2.1.4 Android Things e Google

Cloud IoT Core è un servizio completamente gestito su Google Cloud Platform che consente di connettere, gestire e inserire dati in modo semplice e sicuro da milioni di dispositivi dispersi a livello globale. Android Things è progettato per supportare qualsiasi cosa, dalla raccolta di dati di telemetria a potenti applicazioni di visione artificiale, elaborazione audio e apprendimento automatico, utilizzando *Cloud IoT Core*, per inviare i dati a *Google Cloud Platform*, per poter fare ulteriori analisi.

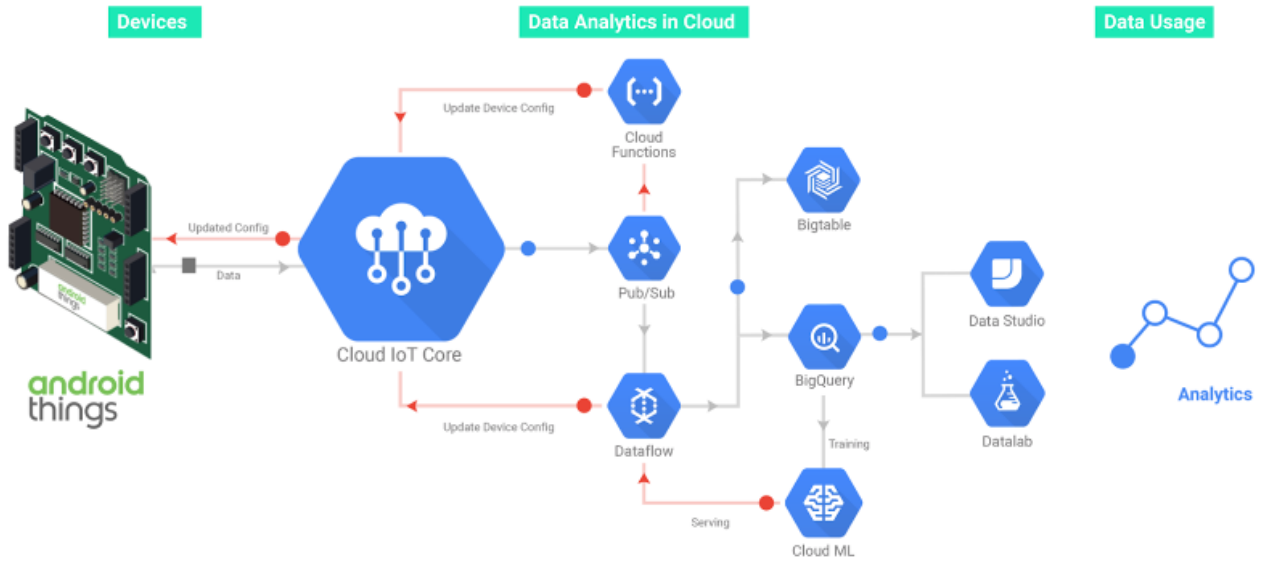


Figura 4 - https://developer.android.com/things/get-started/google-services

La libreria client *Cloud IoT Core* è stata progettata per consentire agli sviluppatori di Android Things di interagire con poche righe di codice. La libreria client gestisce la rete, il threading e la gestione dei messaggi, implementando le best practices per l'autenticazione, la sicurezza, la gestione degli errori e il funzionamento offline.

2.1.5 Android Things 1.0 Feature e API

2.1.5.1 Sistema

Home Activity Support

Android Things prevede che un'app esponga nel proprio file manifest una "home activity" come punto di accesso principale per l'avvio automatico del sistema al boot. Questa activity deve contenere un intent filter che includa sia CATEGORY_DEFAULT che CATEGORY_HOME. Per facilitare lo sviluppo, questa stessa activity dovrebbe includere un intent filter CATEGORY_LAUNCHER in modo che Android Studio possa avviarla come activity predefinita durante la distribuzione o il debug.

```
<application
    android:label="@string/app_name">
    <activity android:name=".HomeActivity">
        <!-- Launch activity as default from Android Studio -->
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
        <!-- Launch activity automatically on boot, and re-launch if the app terminates. -->
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.HOME"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
    </activity>
</application>
```

Device Updates

Android Things 1.0 consente alle app di controllare e monitorare il processo di applicazione degli aggiornamenti software over-the-air (OTA) da Android Things Console. Utilizza un ***UpdateManager*** per impostare la politica di aggiornamento e configurare il comportamento di riavvio del dispositivo a seguito di un aggiornamento OTA. È possibile attivare manualmente il riavvio del dispositivo o il ripristino delle impostazioni di fabbrica con il ***DeviceManager***.

2.1.5.2 Connettività

Bluetooth

Android Things 1.0 arricchisce le principali API Bluetooth Android con funzionalità aggiuntive per la gestione dello stato dei dispositivi. Configura i profili Bluetooth attivi con ***BluetoothProfileManager*** e imposta gli attributi e le funzionalità del dispositivo con ***BluetoothConfigManager***. Utilizza ***BluetoothConnectionManager*** per avviare l'associazione con un dispositivo remoto, gestire le richieste di associazione in arrivo e controllare il processo di connessione.

LoWPAN

Permette di connettere i dispositivi a reti personali senza fili a basso consumo basate su IP (LoWPAN), comprese le reti Thread. Utilizza ***LowpanManager*** per scoprire le interfacce radio supportate sul dispositivo e ascoltare i cambiamenti di stato.

2.2 Actions on Google Smart Home

2.2.1 Overview

La piattaforma Google Smart Home consente agli utenti di controllare i dispositivi connessi disponibili in commercio tramite l'app Google Home e Google Assistant. Le azioni Smart home si basano su Home Graph, un database che memorizza e fornisce dati contestuali sulla casa e sui suoi dispositivi. Il database Home Graph memorizza le informazioni su strutture (ad esempio, casa o ufficio), stanze (ad esempio, camera da letto o soggiorno) e dispositivi (ad esempio altoparlante e lampadina). Queste informazioni sono disponibili per l'Assistente Google per eseguire le richieste degli utenti in base al contesto appropriato. Ogni struttura ha il proprio set di stanze e dispositivi. Una struttura è composta da quanto segue:

- **Gestori** - L'account del proprietario della struttura. Ogni struttura deve avere almeno un responsabile.
- **Camere** - Le camere che fanno parte di una struttura.
- **Dispositivi**: i dispositivi che fanno parte di una struttura. Questi possono essere dispositivi di più produttori.

I dispositivi devono avere queste proprietà:

- **Tipo**: il tipo di dispositivo come una lampada, una fotocamera o un condizionatore d'aria.
- **Tratti**: il tipo di tratti supportati dal dispositivo. Ogni dispositivo può avere una serie di tratti. Ad espio, una luce potrebbe avere tratti come luminosità e settaggio del colore.

I tratti hanno queste proprietà:

- **Attributi**: gli attributi statici per un dispositivo. Un attributo può essere qualcosa come unità di temperatura, modalità, ecc ...
- **Stato**: lo stato o gli stati del dispositivo. Una lampada potrebbe restituire uno stato di luminosità per indicare la luminosità corrente di quella specifica lampada.
- **Etichette**: l'etichetta che identifica il dispositivo

Quando si ha una conversazione con l'Assistente Google come "Hey Google, accendi la luce della camera da letto", Hey Google è l'invocazione e accendi la luce della camera da letto è nota come grammatica. Google determina lo ***smart home intent*** dalla grammatica e lo invia al developer cloud (**fulfillment**). Il developer può quindi eseguire il comando sul dispositivo e restituire una risposta a Google. I tratti del dispositivo definiscono le capacità di un tipo di dispositivo.

2.2.2 Smart Home Intents

Gli smart home intents sono semplici oggetti di messaggistica che descrivono quale azione smart home eseguire, come accendere una luce o trasmettere l'audio a un altoparlante. Tutti gli intents della smart home sono contenuti nello spazio dei nomi **action.devices** e bisogna fornir loro il compito da svolgere. Ogni volta che l'Assistente Google invia un intent per l'adempimento, il token di accesso OAuth2 di terze parti di un utente viene passato nell'intestazione di autorizzazione. Ci sono quattro tipi di intents: SYNC, QUERY, EXECUTE, DISCONNECT.

action.devices.SYNC viene utilizzato per richiedere l'elenco dei dispositivi per la casa intelligente che l'utente ha connesso e che sono disponibili per l'uso.

Quando un utente configura i propri dispositivi con l'app Google Home, viene anche autenticato nella tua infrastruttura cloud. Quindi, l'assistente riceve un token OAuth2. A questo punto, l'Assistente Google invia

un action.devices.SYNC al proprio “fulfillment” per recuperare l'elenco iniziale dei dispositivi e delle funzionalità degli utenti dalla infrastruttura cloud.

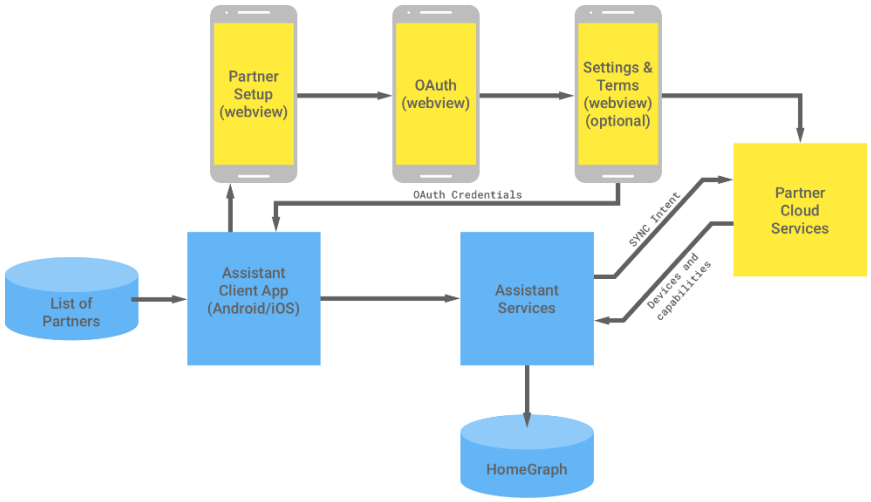


Figura 5 - SYNC

action.devices.QUERY viene utilizzato per interrogare lo stato corrente dei dispositivi domestici intelligenti.

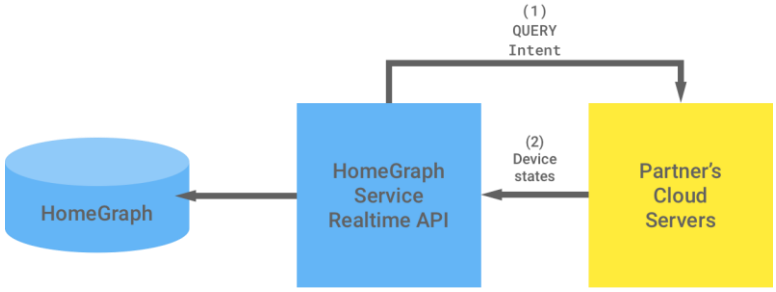


Figura 6 – QUERY

action.devices.EXECUTE viene utilizzato per fornire comandi da eseguire su dispositivi domestici intelligenti.

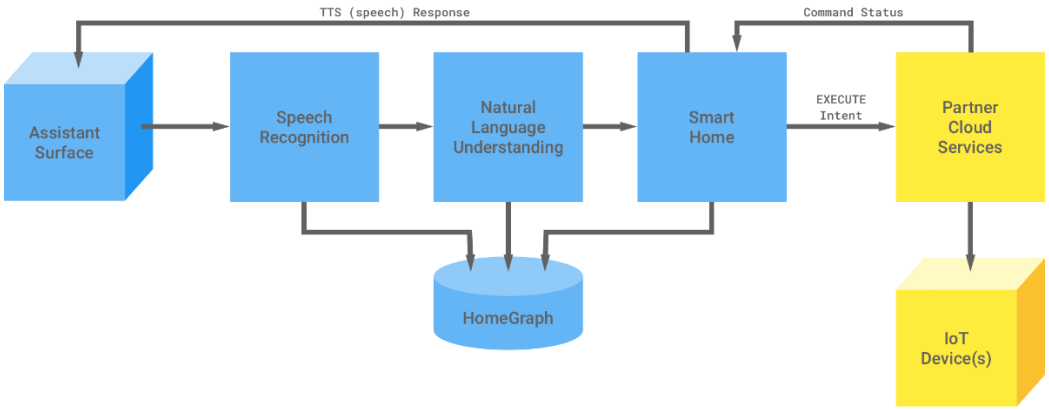


Figura 7 – EXECUTE

action.devices.DISCONNECT viene attivato per informare quando un utente ha scollegato l'account dell'app dall'Assistente Google.

2.2.3 Fulfillment

Il fulfillment è un codice distribuito come webhook che consente di generare risposte dinamiche per ogni tipo di intent domestico intelligente. Durante una conversazione dell'utente con l'Assistente Google, il fulfillment consente di utilizzare le informazioni estratte dall'elaborazione del linguaggio naturale di Google per generare risposte dinamiche o attivare azioni sul back-end come l'accensione di una luce. Il fulfillment riceve richieste dall'Assistente, elabora la richiesta e risponde. Questo processo di richiesta e risposta avanti e indietro porta avanti la conversazione fino a quando non si soddisfa la richiesta iniziale dell'utente.

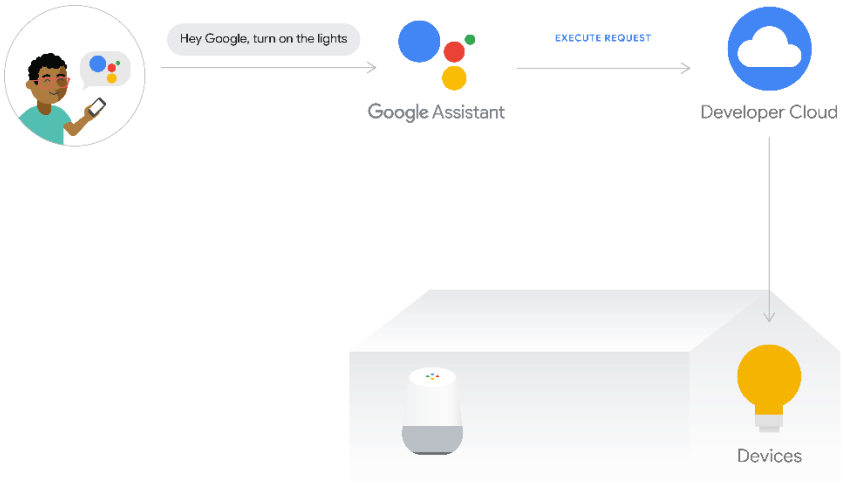


Figura 8 – fulfillment

2.3 Firebase

2.3.1 Overview

Firebase è una piattaforma per la creazione di applicazioni per dispositivi mobili e web sviluppata da Google. È uno strumento utile per aiutare gli sviluppatori nella gestione del server, delle API e dell'archivio dati. In particolare per questo progetto sono stati usati i moduli Firestone, Realtime database, Functions.

2.3.2 Functions

Cloud Functions per Firebase è un framework serverless che consente di eseguire automaticamente il codice di backend in risposta agli eventi attivati dalle funzionalità Firebase e dalle richieste HTTPS. Il codice JavaScript o TypeScript è archiviato nel cloud di Google e viene eseguito in un ambiente gestito. Non è necessario gestire e ridimensionare i propri server. Le funzionalità chiave sono:

- Integra la piattaforma Firebase: Le funzioni scritte possono rispondere agli eventi generati da varie funzionalità di Firebase e Google Cloud, dai trigger di Firebase Authentication ai trigger di Cloud Storage . Cloud Functions riduce al minimo il codice boilerplate, semplificando l'utilizzo di Firebase e Google Cloud all'interno delle proprie funzioni.
- Zero manutenzione: si distribuisce il codice JavaScript o TypeScript sui server di Firebase con un comando dalla riga di comando. Successivamente, Firebase ridimensiona automaticamente le risorse di elaborazione per adattarle ai modelli di utilizzo degli utenti. Non bisogna preoccuparsi delle credenziali, della configurazione del server, del provisioning di nuovi server o della disattivazione di quelli vecchi.
- Mantiene la logica privata e sicura: in molti casi, gli sviluppatori preferiscono controllare la logica dell'applicazione sul server per evitare manomissioni sul lato client. Inoltre, a volte non è desiderabile consentire il reverse engineering di quel codice. Cloud Functions è completamente isolato dal client, quindi si può essere certi che sia privato e che faccia sempre esattamente ciò che si desidera.

Dopo aver scritto e distribuito una funzione, i server di Google iniziano a gestirla immediatamente. Si possono attivare la funzione direttamente con una richiesta HTTP oppure, nel caso di funzioni in background, i server di Google ascolteranno gli eventi ed eseguiranno la funzione quando viene attivata. Man mano che il carico aumenta o diminuisce, Google risponde ridimensionando rapidamente il numero di istanze del server virtuale necessarie per eseguire la funzione. Ogni funzione viene eseguita in isolamento, nel proprio ambiente con la propria configurazione.

2.3.3 Firestore

Cloud Firestore è un database flessibile e scalabile per lo sviluppo di dispositivi mobili, web e server da Firebase e Google Cloud. Mantiene i dati sincronizzati tra le app client tramite listener in tempo reale e offre supporto offline per dispositivi mobili e web in modo da poter creare app reattive che funzionano indipendentemente dalla latenza di rete o dalla connettività Internet. Cloud Firestore offre anche una perfetta integrazione con altri prodotti Firebase e Google Cloud, tra cui Cloud Functions. Le funzionalità chiave sono:

- Flessibilità: il modello di dati di Cloud Firestore supporta strutture di dati flessibili e gerarchiche. Archivia i dati in documenti, organizzati in raccolte. I documenti possono contenere oggetti nidificati complessi oltre alle sottoraccolte.
- Interrogazione espressiva: Cloud Firestore, si possono utilizzare le query per recuperare singoli documenti specifici o per recuperare tutti i documenti in una raccolta che corrispondono ai parametri di query. Le query possono includere più filtri concatenati e combinare filtri e ordinamento. Sono

inoltre indicizzati per impostazione predefinita, quindi le prestazioni delle query sono proporzionali alle dimensioni del set di risultati, non al set di dati.

- Aggiornamenti in tempo reale: utilizza la sincronizzazione dei dati per aggiornare i dati su qualsiasi dispositivo connesso. Tuttavia, è anche progettato per eseguire query di recupero semplici e una tantum in modo efficiente.
- Supporto offline: Cloud Firestore memorizza nella cache i dati che l'app sta utilizzando attivamente, in modo che l'app possa scrivere, leggere, ascoltare ed eseguire query sui dati anche se il dispositivo è offline. Quando il dispositivo torna online, Cloud Firestore sincronizza eventuali modifiche locali di nuovo su Cloud Firestore.
- Progettato per scalare: Cloud Firestore offre il meglio della potente infrastruttura di Google Cloud: replica automatica dei dati in più regioni, forti garanzie di coerenza, operazioni batch atomiche e supporto di transazioni reali.

Cloud Firestore è un database NoSQL ospitato nel cloud a cui le app iOS, Android e Web possono accedere direttamente tramite SDK nativi. Cloud Firestore è disponibile anche in Node.js nativi, Java, Python, Unity, C ++ e Go SDK, oltre alle API REST e RPC.

2.3.4 Realtime Database

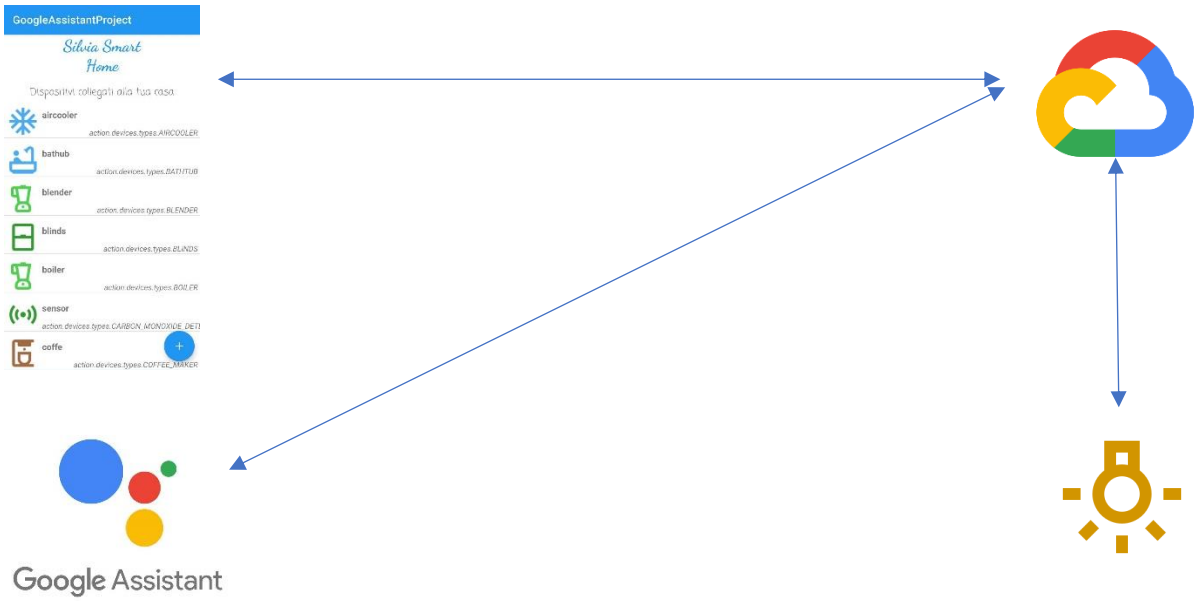
Firebase Realtime Database è un database ospitato nel cloud. I dati vengono archiviati come JSON e sincronizzati in tempo reale con ogni client connesso. Quando si creano app multipiattaforma con SDK iOS, Android e JavaScript, tutti i clienti condividono un'istanza di Realtime Database e ricevono automaticamente gli aggiornamenti con i dati più recenti. Le funzionalità chiave sono:

- Tempo reale: invece delle tipiche richieste HTTP, Firebase Realtime Database utilizza la sincronizzazione dei dati: ogni volta che i dati cambiano, qualsiasi dispositivo connesso riceve l'aggiornamento entro millisecondi.
- Disconnesso: le app Firebase rimangono reattive anche offline perché Firebase Realtime Database SDK conserva i tuoi dati su disco. Una volta ristabilita la connettività, il dispositivo client riceve le modifiche perse, sincronizzandole con lo stato del server corrente.
- Accessibile dai dispositivi client: è possibile accedere al Firebase Realtime Database direttamente da un dispositivo mobile o da un browser web; non è necessario un server delle applicazioni. La sicurezza e la convalida dei dati sono disponibili tramite Firebase Realtime Database Security Rules, regole basate su espressioni che vengono eseguite quando i dati vengono letti o scritti.
- Scala su più database: si può supportare le esigenze di dati dell'app su larga scala suddividendo i dati su più istanze di database nello stesso progetto Firebase.

Capitolo 3: Sviluppo applicazione C/S per la gestione di una smart home

3.1 Obiettivo

L'obiettivo è realizzare una applicazione Android con la quale poter collegare e gestire dispositivi connessi sfruttando la tecnologia "Action on Google Smart Home". Questi dispositivi saranno controllabili non solo con l'app custom ma anche tramite l'Assistente di Google e l'app Google Home.



3.2 Progettazione e sviluppo

3.2.1 Backend e gestione dati

Per poter realizzare questo progetto per prima cose è necessario creare una Smart Home Action nella Action Console (piattaforma per sviluppatori per estendere le capacità dell'Assistente di Google). Per poter gestire l'interazione con la smart home e con l'assistente di Google ho scelto di realizzare un servizio in Node.js che si interfaccia con Cloud Functions for Firebase e Firebase Realtime database. Questo servizio è composto da varie funzioni:

- smarthome: attraverso la quale vengono gestite tutte gli smart home intents (sync, query, execute, disconect)
- fakeauth: per la gestione la richiesta di autorizzazione
- faketoken: per la gestione del token di autorizzazione
- login: per la gestione del login dell'utente associato alla smart home
- requestsync: per la comunicazione dello stato della smart home con Google Home Graph
- addMessage: per la comunicazione dei nuovi dispositivi che provengono dall'app
- reportstate: per riportare a Google Home Graph le modifiche apportate allo stato del dispositivo in seguito all'esecuzione di un comando

Dopo aver sviluppato le funzionalità di base del servizio è stato necessario configurare la Smart Home Action indicando come “fullfillment” i riferimenti alle funzioni di Firebase Functions.

Un intent di SYNC si verifica quando l'assistente desidera sapere quali dispositivi ha connesso l'utente. Viene inviato al servizio quando l'utente collega un account. Quindi la funzione smarthome.onSync risponde con un payload JSON di tutti i dispositivi dell'utente e delle loro capacità. Un intent di QUERY si verifica quando l'assistente desidera conoscere lo stato o lo stato corrente di un dispositivo. Quindi la funzione smarthome.onQuery risponde con un payload JSON con lo stato di ogni dispositivo richiesto. Un intent di EXECUTE si verifica quando l'assistente desidera controllare un dispositivo per conto di un utente. Quindi la funzione smarthome.onExecute risponde con un payload JSON con lo stato di esecuzione di ogni dispositivo richiesto. Un intent di DISCONNECT si verifica quando l'utente scollega il proprio account dall'Assistente, quindi smarthome.onDisconnect interrompe l'invio di eventi per i dispositivi di questo utente, all'Assistente.

Per la gestione dei dati ho scelto di usare due database:

- un realtime database, che è collegato alla funzione smarthome per la gestione dei cambiamenti di stato e dei comandi di attuazione che vengono mandati ai dispositivi connessi. Ho scelto questa tipologia di db perché è comodo per la visualizzazione dei cambiamenti di stato in realtime su più dispositivi.
- un firestore database, per mantenere tutte le informazioni relative ad un dispositivo, e per poter facilmente gestire il collegamento di questi ultimi tramite un’applicazione custom. La function smarthome.onSync per poter inviare tutti i dispositivi connessi all’Assistente di Google va a leggere tutti gli elementi presenti nel Firestore database nella raccolta devices e li prepara per inserirli nella risposta.

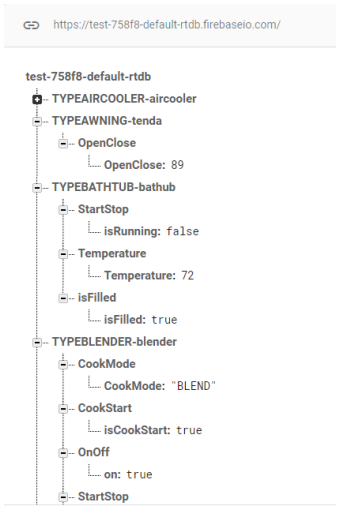


Figura 9 - Realtime Database

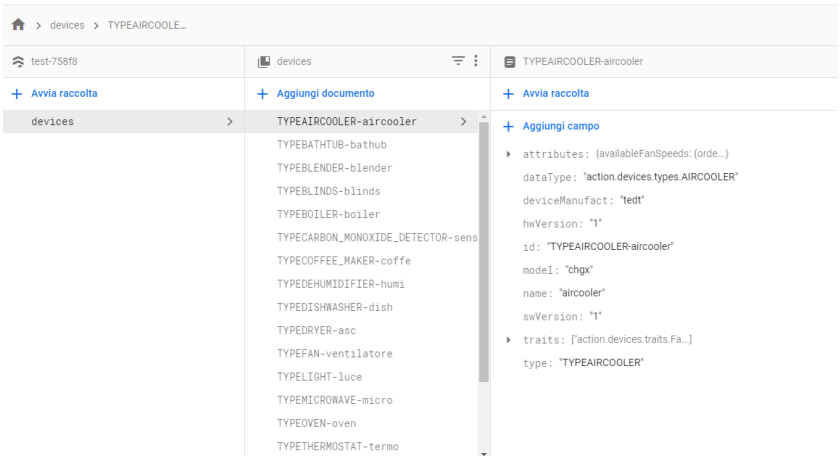


Figura 10 - Firestore DB

3.2.2 Applicazione Android

L'applicazione per la gestione di dispositivi connessi è stata pensata per poter offrire all'utente la possibilità di collegare alla smart home nuovi dispositivi, gestire i dispositivi connessi, scollegare i dispositivi e visualizzare lo stato corrente della propria smart home. L'app si presenta con una schermata principale in cui si può visualizzare la lista dei dispositivi collegati alla smart home. Per ogni dispositivo è presente un'icona, il nome del dispositivo e la tipologia. Con il pulsate "+" in basso sarà possibile accedere alla schermata utile per aggiungere un nuovo dispositivo. Per aggiungere un dispositivo è necessario inserire il nome, il modello, la ditta produttrice e la tipologia.

Figura 11 - Schermata principale

Figura 12 - Aggiunta dispositivo

Nell'applicazione sono supportate varie tipologie di dispositivi con i relativi tratti. Le tipologie disponibili sono: washer, oven, aircooler, awning, bathub,blinds, blender, boiler, carbon monoxide detector, coffemaker, dehumidifier, dishwasher, dryer, fan, freezer, light, microwave, smoke detector, tv, thermostat, window.

I tratti applicabili (a seconda dei quali il dispositivo supporta) sono:

- OnOff: la funzionalità di accensione e spegnimento di base per qualsiasi dispositivo con attivazione e disattivazione binaria, inclusi connettori e interruttori.
- Cook: questa caratteristica appartiene ai dispositivi in grado di cuocere il cibo in base a varie modalità di cottura supportate.
- TemperatureControl: Caratteristica per dispositivi che supportano il controllo della temperatura, all'interno o intorno al dispositivo.
- StartStop: questa caratteristica appartiene ai dispositivi che supportano l'avvio e l'arresto delle operazioni. L'avvio e l'arresto di un dispositivo ha una funzione simile all'accensione e allo spegnimento. I dispositivi che ereditano questa caratteristica funzionano in modo diverso quando vengono accesi e quando vengono avviati. Alcune lavatrici, ad esempio, possono essere accese e modificate le loro impostazioni prima di avviare effettivamente il funzionamento.

- Modes: questa caratteristica copre tutte le modalità disponibili e le impostazioni specifiche della modalità per un dispositivo.
- Toggles: questa caratteristica appartiene a tutti i dispositivi con impostazioni che possono esistere solo in uno dei due stati.
- FanSpeed: Questa caratteristica appartiene ai dispositivi che supportano l'impostazione della velocità di una ventola. Le velocità della ventola (ovvero, soffia aria dal dispositivo a vari livelli, che può essere parte di un'unità di condizionamento o riscaldamento o in un'auto), possono includere impostazioni come bassa, media e alta.
- HumiditySettings: questa caratteristica appartiene ai dispositivi che supportano impostazioni di umidità come umidificatori e deumidificatori.
- TemperatureSettings: questo tratto copre la gestione sia del punto di temperatura che delle modalità.
- OpenClose: questa caratteristica appartiene ai dispositivi che supportano l'apertura e la chiusura.
- Fill: questa caratteristica si applica ai dispositivi che supportano il riempimento come una vasca da bagno.
- Rotation: questa caratteristica appartiene ai dispositivi che supportano la rotazione.
- SensorState: questo tratto copre sia la misurazione quantitativa che lo stato qualitativo.
- ColorSettings: questa caratteristica si applica ai dispositivi, come le luci intelligenti, che possono cambiare il colore o la temperatura del colore.
- Brightness: questa caratteristica spiega come controllare la luminosità di un dispositivo.
- AppSelector: questa caratteristica viene utilizzata per i dispositivi in grado di cambiare input.
- Volume: questa caratteristica appartiene ai dispositivi che sono in grado di modificare il volume (ad esempio, impostando il volume a un certo livello, disattivando o riattivando l'audio).
- LockUnlook: Questa caratteristica appartiene a tutti i dispositivi che supportano il blocco e lo sblocco.

Per gestire i vari dispositivi collegati, basterà cliccare su uno di essi, presente nella lista, e si aprirà la schermata di gestione, in cui si potrà visualizzare lo stato attuale e modificarlo. Inoltre se si preme l'icona con il cestino si potrà anche eliminare il dispositivo.

Figura 13 - Gestione dispositivo

Per poter leggere i dispositivi collegati alla smart-home, nell'app si accede ad un istanza del firestore database e si legge tutta la collezione "devices".

```
db.collection("devices").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        ArrayList<Devices> dev = new ArrayList<>();;
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
               // if(utils.devicesList.co)
               // Devices d = document.toObject(Devices.class);
                String name =(String) document.get("name");
                String type =(String) document.get("type");
                String model = (String) document.get("model");
                String deviceManufact = (String) document.get("deviceManufact");
                DeviceTypes ty = DeviceTypes.valueOf(type);
                Devices toAdd = null;
                try {
                    toAdd = new Devices(name,ty,deviceManufact,model);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
                dev.add(toAdd);
                //utils.SetDevice(d);
            }
            adapter.addAll(dev);

        } else {
            Log.d(TAG, "Error getting documents: ", task.getException());
        }
    }

});
```

Per poter rimuovere un dispositivo collegato alla smart-home si fa una chiamata al Firestore db di tipo delete.

```
public void removeDoc(Devices d){
    db.collection("devices").document(d.id).delete();
}
```

Per poter essere sempre aggiornati sullo stato attuale dei dispositivi, bisogna ottenere un riferimento al realtime database e sottoscrivere un listener.

```
database = FirebaseDatabase.getInstance();
myRef = database.getReferenceFromUrl(url);
// Read from the database
myRef.addValueEventListener(new ValueEventListener() {

    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {

        String dbvalue = dataSnapshot.getValue(String.class);
        if(dbvalue != null) {
            if (listner != null)
                listner.onStringChange(dbvalue); } }
```

Per poter cambiare lo stato relativo ad un dispositivo è necessario riprendere il riferimento al realtime database e andare a settare il valore desiderato. Quando viene cambiato un valore relativo allo stato, viene invocata la function “reportstate” che va ad aggiornare Google Home Graph.

```
public void SetValueOnDb(String value){
    myRef.setValue(value);
}
```

Per poter visualizzare la lista dei dispositivi è stato creato un Adapter:

```
public DeviceArrayAdapter(Context context, ArrayList<Devices> dev) {
    super(context, 0, dev);
}

public View getView(int position, View convertView, ViewGroup parent) {
    Devices dev = getItem(position);
    if (convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_template,
parent, false);
    }
    TextView devName = (TextView) convertView.findViewById(R.id.devName);
    TextView devType = (TextView) convertView.findViewById(R.id.devType);
    ImageView devImg = (ImageView) convertView.findViewById(R.id.devImg);
    devName.setText(dev.getName());
    devType.setText(dev.getTypeToSend());
    devImg.setImageResource(dev.getDevImg());
    return convertView;
}
```

3.2.3 Alcune schermate dell’app

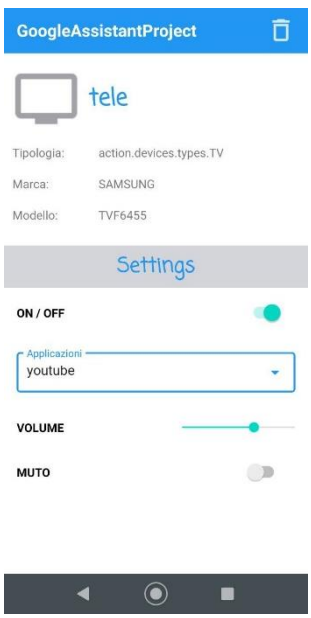


Figura 14 - TV

Figura 15 - Termostato

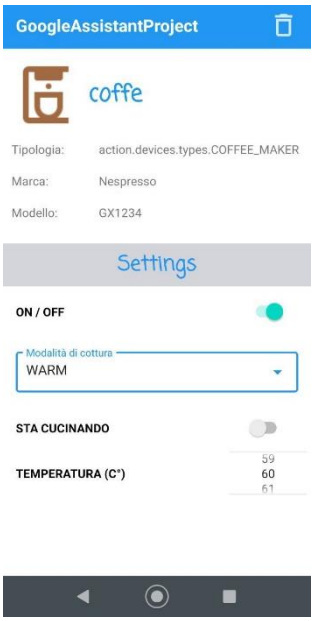


Figura 16 – Macchina caffe

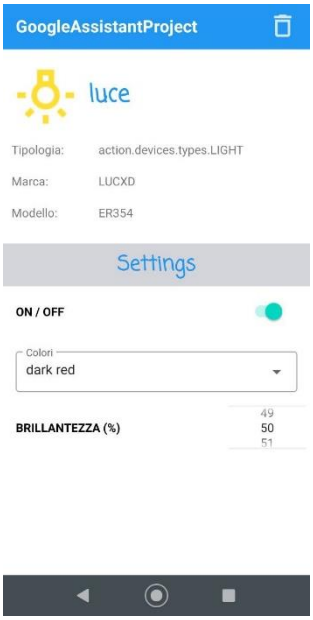


Figura 17 – Luce

Capitolo 4: Analisi delle prestazioni

Per misurare le performance lato app ho usato Android Profiler, uno strumento integrato in Android Studio che fornisce dati in tempo reale sull'utilizzo di CPU, memoria, rete e risorse della batteria. Il dispositivo fisico utilizzato è un Motorola Moto G7 Plus. In figura 18 viene riportato il grafico relativo a circa 7 minuti di utilizzo. Possiamo notare che l'uso di CPU resta relativamente basso durante l'uso dell'app. L'uso della rete presenta dei picchi in corrispondenza della creazione di un nuovo dispositivo (minuto circa 5.40 e 6.30) e in caso di eliminazione di un dispositivo (minuto 1.30 circa), mentre durante le altre funzionalità il carico di rete è basso. In generale il consumo di batteria e di CPU si mantengono nella media.

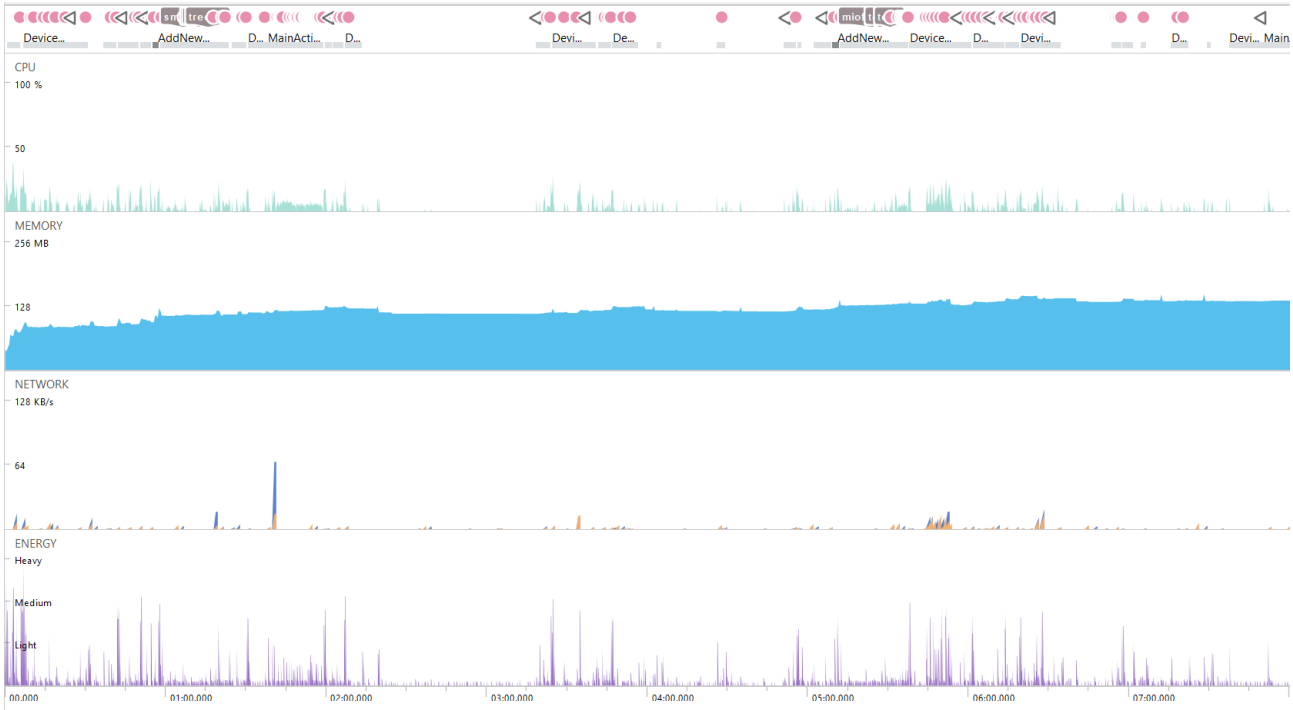


Figura 18 - Performance app

Inoltre ho raccolto altri dati di performance attraverso il modulo Firebase Performance Monitoring, un servizio che aiuta a ottenere informazioni dettagliate sulle caratteristiche delle prestazioni delle app iOS, Android e web. Per le app native viene registrato l'ora di avvio, il rendering dei dati per schermata e l'attività in primo piano o in background e le richieste di rete http.

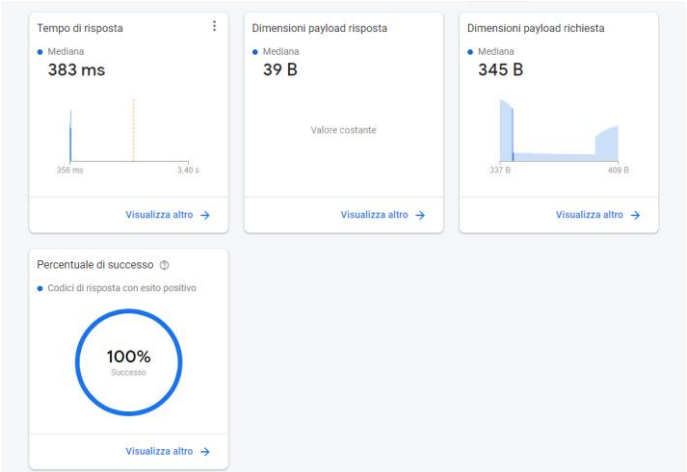


Figura 19 - Performance app – Firebase

Per il lato backend, Firebase offre strumenti per la valutazione delle prestazioni. In seguito verranno riportati i grafici per le functions principali usate per la realizzazione del progetto. Uno dei vantaggi di usare servizi cloud è che le risorse di elaborazione vengono ridimensionate automaticamente, per adattarle ai modelli di utilizzo degli utenti. Non bisogna preoccuparsi della configurazione del server, del provisioning di nuovi server o della disattivazione di quelli vecchi.

- addMessage

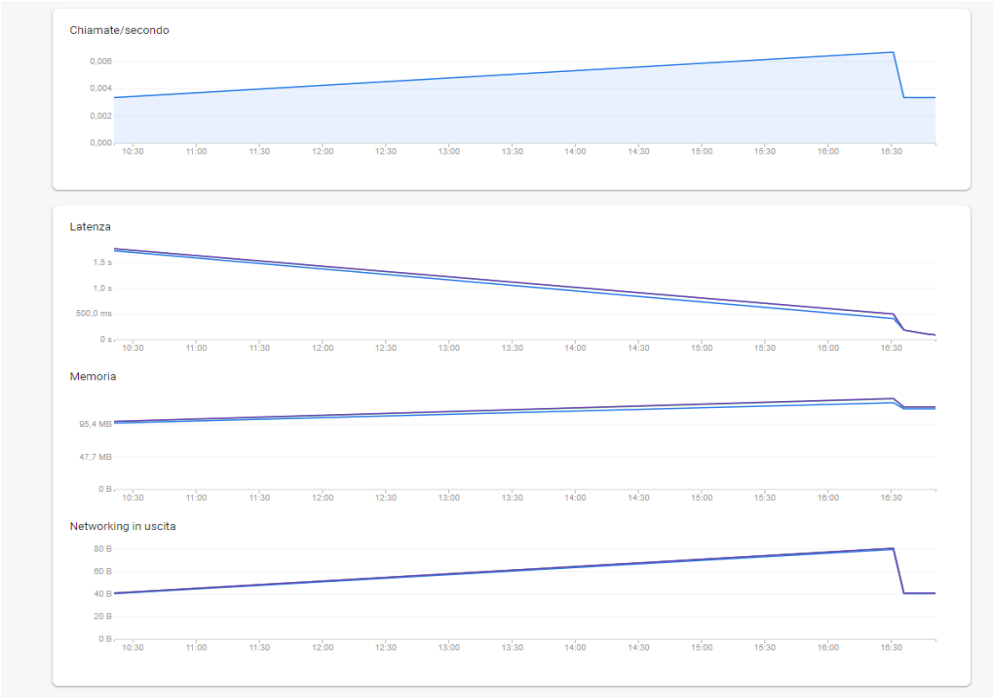


Figura 20 -Analisi performance addMessage

- reportstate

Figura 21 - Analisi performance reportstate

- smarthome

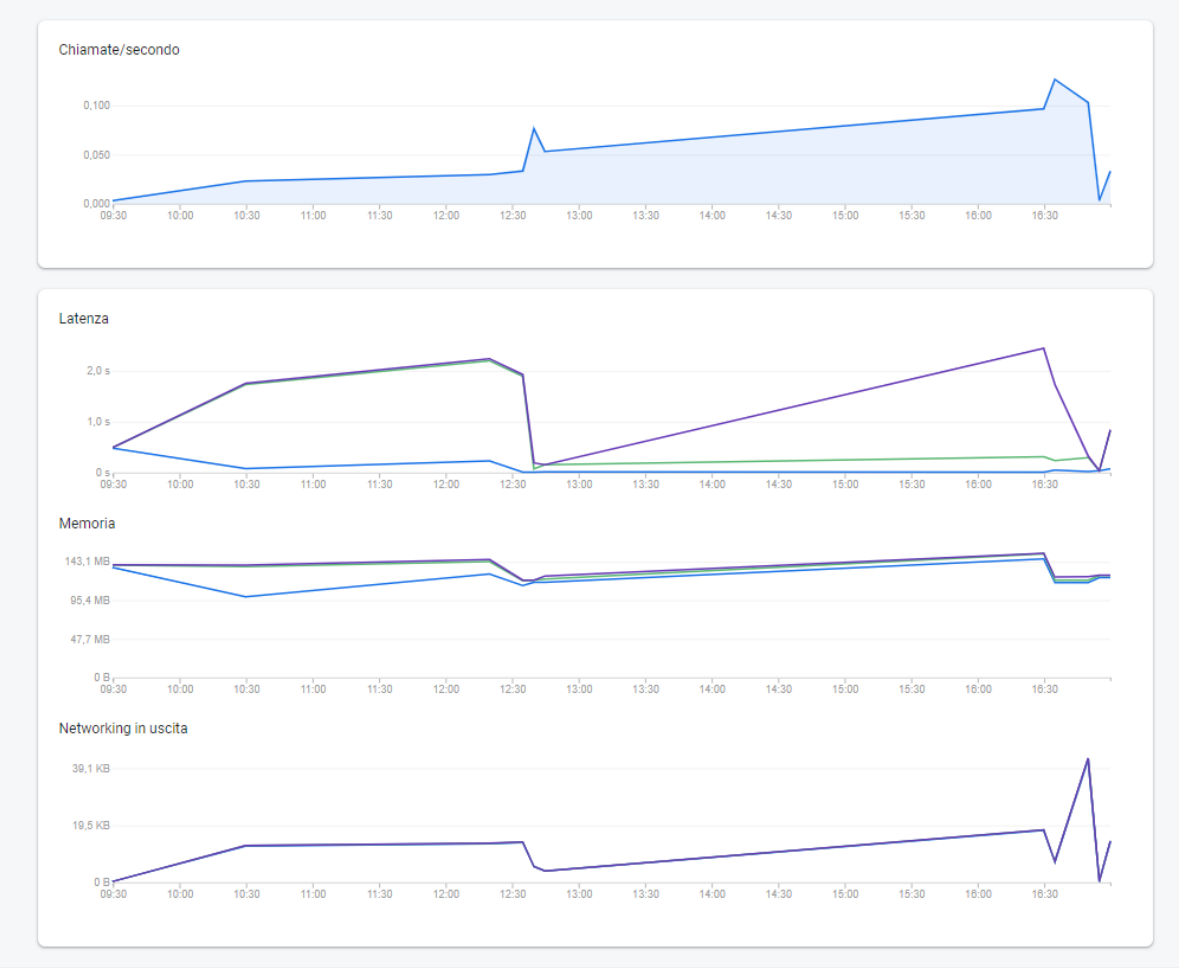


Figura 22 - Analisi performance smarthome