Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

Applicazione per il monitoraggio di parametri vitali attraverso l'uso di Bluetooth Low Energy

Relatore:
Chiar.mo Prof.
Davide Rossi

Presentata da:
Luca Romagnoli

Sessione III
Anno Accademico 2018-2019

Alla mia famiglia ...

Introduzione

Le cure mediche sono costose, spesso si basano su attrezzature all'avanguardia e su personale altamente qualificato. Se l'assistenza sanitaria utilizzasse risorse umane e fisiche in modo più efficiente, allora ci sarebbe un grande potenziale di risparmio sui costi, in particolare riducendo le spese per le attività di routine. [4]

Nel campo dell'assistenza sanitaria l'Internet of Things[1] consente di ottenere enormi risparmi, aumenti in termini di efficienza e miglioramenti nel comfort del paziente. Importanti aree di cura del paziente vengono spostate in casa e quindi fuori dai reparti ospedalieri. Dispositivi di monitoraggio come bilance mediche, cardiofrequenzimetri, sfigmomanometri e termometri digitali possono tenere traccia della salute e avvisare i pazienti, le famiglie e gli operatori sanitari dei cambiamenti nei parametri vitali. L'Internet of Things ha reso necessario l'utilizzo di una tecnologia wireless solida e sicura, che permettesse di collegare i dispositivi medici ad applicazioni diagnostiche, di monitoraggio e assistenziali nelle strutture sanitarie e nell'ambiente domestico. La tecnologia ormai matura e solida che viene impiegata in applicazioni mediche di questo tipo è il Bluetooth.

[1] Internet of Things (letteralmente "Internet delle cose") fa riferimento a un sistema di dispositivi connessi, capaci di acquisire e trasferire dati su rete.

In particolare, con l'ecosistema tecnologico di oggi e gli aspetti di basso consumo energetico, frequency hopping[2], basso costo ecc., sempre più emergenti nello scenario IoT [4], viene utilizzata una particolare versione del Bluetooth: il Bluetooth Low Energy (o BLE). Il Bluetooth Low Energy svolge un ruolo centrale nella rivoluzione dell'IoT, in particolare per le comunicazioni wireless a corto raggio con dispositivi in cui la durata della batteria è fondamentale. Le caratteristiche a bassa potenza, solide e facili da usare del BLE sono ideali per collegare più dispositivi di monitoraggio a un dispositivo locale (come lo smartphone), il quale condivide i dati in modo sicuro con gli operatori sanitari per l'analisi degli stessi. Il numero significativo di dispositivi medici, spesso ad uso domestico, che supportano questa tecnologia, permettono agli ospedali e alle aziende sanitarie di interagire meglio con i loro pazienti, prima, durante e dopo ogni visita. Viene data ai pazienti una maggiore consapevolezza sulla propria condizione fisica, permettendo loro di monitorare i propri livelli di salute e di adottare misure preventive per evitare lo sviluppo di patologie. Un'applicazione medica installata su uno smartphone, alla quale viene collegato via BLE un monitor per misurare la pressione arteriosa (sfigmomanometro[3]), consente di registrare le variazioni cronologiche dei valori di un paziente e informare direttamente un medico sullo stato di salute. Può essere utile, al medico, nel trattamento di alcune malattie e a fare una diagnosi corretta, senza aver bisogno di un contatto visivo col paziente.

In questo elaborato, dopo un'introduzione sulle tecnologie Bluetooth e Bluetooth Low Energy, viene presentata un'applicazione per il monitoraggio di un paziente e in particolare una componente autonoma integrativa finalizzata all'acquisizione di alcuni parametri vitali, quali: peso, pressione arteriosa minima (diastolica), pressione arteriosa massima (sistolica) e pulsazioni. Questi parametri vengono raccolti per mezzo di due dispositivi medici BLE: bilancia e sfigmomanometro.

---

[2]Tecnica di trasmissione radio usata per ridurre le interferenze e contrastare le intercettazioni; consiste nel variare rapidamente il canale di frequenza durante la trasmissione radio.

[3]Termine che indica un misuratore di pressione.

Indice

Introduzione **i**

Elenco delle figure **vii**

Frammenti di codice **ix**

1 Introduzione al Bluetooth e al Bluetooth Low Energy **1**

1.1 Bluetooth 1

1.1.1 Cenni Storici 1

1.1.2 Connessione 2

1.1.3 Profili 4

1.1.4 Usi e Contesti 5

1.1.5 Versioni 6

1.1.6 Portata massima 8

1.1.7 Sicurezza 10

1.1.8 Bluetooth: prima e dopo 11

1.2 Bluetooth Low Energy 12

1.2.1 Componenti fondamentali: GAP 13

1.2.2 Componenti fondamentali: GATT 18

1.2.3 Profili, Servizi e Caratteristiche 21

1.2.4 Universally Unique IDentifier (UUID) 22

1.2.5 Android e BLE 23

1.2.6 Perchè scegliere Bluetooth Low Energy 26

2 Analisi e Progettazione **27**

2.1 Contesto lavorativo 28

2.2 Descrizione generale dell'applicazione 29

2.3 Requisiti imposti dall'azienda 34

2.4 Componente da sviluppare 36

2.4.1 Funzionalità da implementare 37

2.4.2 Ulteriori requisiti 46

2.4.3 Sessione tipica di utilizzo 47

2.4.4 Strumenti utilizzati per lo sviluppo 50

2.4.5 Profili, Servizi e Caratteristiche 52

3 Implementazione **61**

3.1 Panoramica dell'applicazione 61

3.2 Gestione dei dati 64

3.3 Pairing 65

3.4 Sincronizzazione 84

3.5 Visualizzazione 101

Conclusioni **107**

3.6 Sviluppi futuri 108

Bibliografia **111**

Elenco delle figure

1.1 Esempi di piconet master/slave 3

1.2 Profili Bluetooth 5

1.3 Versioni Bluetooth 7

1.4 Classi portata Bluetooth 9

1.5 Logo Bluetooth Smart 12

1.6 Processo di advertising: uno a uno 16

1.7 Processo di advertising: uno a molti 17

1.8 Processo di connessione: uno a molti 19

1.9 Processo di connessione: uno a uno 20

1.10 Schema dell'interfaccia messa a disposizione dal GATT 21

1.11 UUID base dell'UUID completo 22

1.12 UUID completo del Blood Pressure Service 23

1.13 Permessi necessari all'applicazione per poter utilizzare il Bluetooth . 24

1.14 Verifica dell'esistenza di un modulo BLE integrato nel dispositivo Android 24

2.1 Diagramma dei casi d'uso dell'applicazione 37

2.2 Diagramma di sequenza della procedura di pairing 40

2.3 Diagramma di sequenza della procedura di sincronizzazione 43

2.4 Diagramma di sequenza della procedura di visualizzazione 45

2.5 Pulsante hardware Misuratore di pressione 48

2.6 Pairing: fase di advertising 49

2.7 Dispositivo accoppiato correttamente 49

2.8 Sincronizzazione: fase di advertising 50

2.9 Blood Pressure Measurement: Unità di misura e campi opzionali . . 54

2.10 Weight Scale Measurement: Unità di misura e campi opzionali . . . 58

3.1 Schermata Device Setup 62

3.2 Schermata istruzioni Misuratore di pressione 62

3.3 Schermata Dashboard 63

3.4 Menu laterale di navigazione 64

3.5 View relativa al Misuratore di pressione 101

Listings

3.1 Indirizzamento alla schermata di istruzioni 66

3.2 Riferimenti ai dispositivi da accoppiare 67

3.3 Recupero del valore relativo al dispositivo da accoppiare 67

3.4 Impostazione layout della schermata istruzioni 68

3.5 Verifica supporto della tecnologia BLE 68

3.6 Registrazione Broadcast Receiver e relative azioni 69

3.7 Creazione Service 70

3.8 Associazione al Bound Service 70

3.9 Dissociazione dal Bound Service 70

3.10 Creazione interfaccia IBinder 71

3.11 Callback onBind e onUnbind 71

3.12 Metodi chiamati alla creazione/rimozione della connessione 71

3.13 Avvio scansione LE dei dispositivi 72

3.14 Interfaccia per fornire i risultati della scansione 73

3.15 Invio dell'Intent contenente l'azione da gestire 74

3.16 Stop della scansione 74

3.17 Avvio processo di connessione 75

3.18 Connessione al GATT Server 75

3.19 Stato Connessione: GATT Client connesso/disconnesso 77

3.20 Stato Connessione: aggiornamento Servizi 78

3.21 Stato Connessione: scrittura Caratteristica 79

3.22 Stato Connessione: lettura Caratteristica 80

3.23 Scrittura della data 80

3.24 Chiusura della connessione 82

3.25 Impostazione del messaggio di conferma e caricamento della Dashboard 83

3.26 Creazione del Service 84

3.27 Avvio del Service e registrazione del Broadcast Receiver 85

3.28 Verifica attivazione Bluetooth 86

3.29 Creazione della connessione 86

3.30 Avvio scansione su thread differito 87

3.31 Tentativo di connessione con il dispositivo individuato 88

3.32 Verifica pairing del dispositivo 89

3.33 Uno dei 3 metodi intermediari per l'inivio dell'Intent 90

3.34 Recupero tipo di azione da gestire 91

3.35 Tipi di azione da gestire: Gatt Client connesso 92

3.36 Tipi di azione da gestire: Aggiornamento Servizi 92

3.37 Tipi di azione da gestire: Lettura di una Caratteristica 93

3.38 Tipi di azione da gestire: Scrittura di una Caratteristica 95

3.39 Invio notifica al Client per lettura Caratteristica 97

3.40 Recupero dati Caratteristica 97

3.41 Tipi di azione da gestire: Termine lettura dati 99

3.42 Termine procedura di sincronizzazione 99

3.43 Tipi di azione da gestire: Gatt Client disconnesso 100

3.44 Aggiornamento delle View: Dashboard in primo piano 103

3.45 Aggiornamento delle View: Fine sincronizzazione 104

3.46 Aggiornamento delle View: Storico dei valori 105

Capitolo 1

Introduzione al Bluetooth e al Bluetooth Low Energy

In questo capitolo verranno descritti il protocollo Bluetooth e in particolare la versione Bluetooth Low Energy, tecnologia alla base del progetto svolto.

1.1 Bluetooth

Il Bluetooth è uno standard di trasmissione dati per WPAN (acronimo di Wireless Personal Area Network, ovvero Reti Personali Senza Fili). Esso fornisce uno standard economico e sicuro per lo scambio di informazioni tra diversi dispositivi a corto raggio.

1.1.1 Cenni Storici

La tecnologia Bluetooth emerse dal tentativo, intrapreso dalla multinazionale svedese Ericsson Mobile Communications, nel 1994, di trovare alternative all'uso dei

cablaggi per la comunicazione tra telefoni cellulari e altri dispositivi. L'idea era quella di fornire un metodo rapido e semplice. Nel 1998, le società Ericsson, IBM, Nokia e Toshiba formarono il Bluetooth Special Interest Group (SIG[1]), che pubblicò la prima versione Bluetooth nel 1999. Il fatto che le due imprese, Ericsson e Nokia, provengano entrambe dalla Scandinavia è stato sicuramente determinante nella scelta del nome: Harald Blåtand, in italiano "dente blu" e in inglese "blue tooth", fu il re vichingo danese che riuscì, nel X secolo, a unificare le tribù nemiche della Norvegia e della Danimarca in un unico regno. L'iconico simbolo del Bluetooth è composto dalle antiche rune germaniche ᚼ e ᛒ, le iniziali di Harald Blåtand (HB). [7]

1.1.2 Connessione

Un dispositivo digitale diventa "compatibile con il Bluetooth" se dispone di un particolare chip Bluetooth, dotato di un'unità trasmittente e una ricevente, installato nei componenti hardware. Ogni dispositivo dotato di un chip bluetooth è in grado di creare una rete di dimensioni limitate ed estremamente sicura (Figura 1.1). Questa rete Bluetooth, chiamata "Personal Area Network (PAN)" o "piconet", è caratterizzata da un'architettura master/slave (simile a quella server/client). Solitamente, la connessione e lo scambio di dati all'interno di una singola PAN avviene tra due dispositivi: Master e Slave. Tuttavia lo standard prevede che ogni device Bluetooth (master) possa connettersi con un massimo di altri 7 dispositivi (slaves), ognuno dei quali dispone di un indirizzo univoco a 48 bit. Il master, ovvero il dispositivo che crea la rete Bluetooth, può inviare o richiedere dati ai suoi slaves. Gli slaves, ovvero i dispositivi connessi, possono solo trasmet-

[1]Il gruppo SIG è oggi un'organizzazione no-profit composta da circa 33.000 imprese (al 2017), che definiscono in modo congiunto gli standard Bluetooth e portano avanti lo sviluppo di questa tecnologia. Ogni impresa che sviluppa e produce dispositivi dotati di Bluetooth è tenuta a entrare nell'organizzazione. I promotori più importanti del SIG sono al momento Apple, Ericsson, Lenovo, Nokia, Toshiba, Intel e Microsoft.

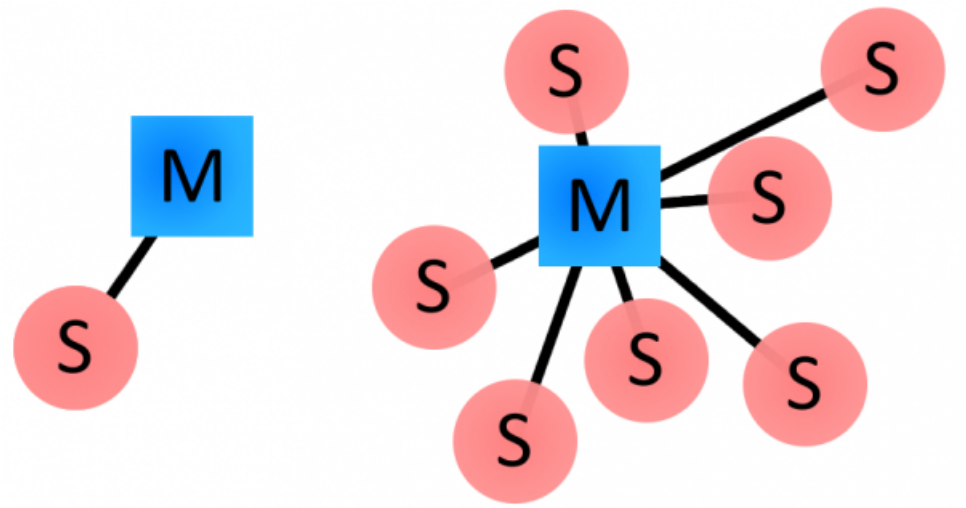

Figura 1.1: Esempi di piconet master/slave

tere o ricevere dati dal proprio master, non possono parlare con altri slaves nel piconet. Ogni slave nella rete può essere collegato con un singolo master, il quale si occupa di coordinare la comunicazione con i vari slaves. La comunicazione si basa sulla trasmissione di pacchetti, ovvero gruppi di bits che contengono non solo l'informazione vera e propria, ma anche dati aggiuntivi (come l'indirizzo del dispositivo) per il riconoscimento. Lo scambio di dati avviene a intervalli di tempo predeterminati e scanditi dal master: ogni intervallo dura 312,5 microsecondi, nel corso dei quali i dispositivi possono inviare i loro pacchetti dati. [8, 10]

L'associazione bluetooth di due o più dispositivi viene chiamata "pairing" ("accoppiamento"). Per consentire il pairing dei dispositivi, la distanza tra di essi deve essere ridotta e la funzione Bluetooth dei dispositivi da sincronizzare deve essere attiva[2]. Successivamente partirà una scansione della piconet, appena creata dal master, nel tentativo di individuare altre periferiche Bluetooth nel suo raggio d'azione. Una volta individuato il dispositivo slave che si vuole connettere,

[2]L'attivazione avviene, a seconda del dispositivo, tramite un software specifico, una casella di controllo o un pulsante, contraddistinti dal simbolo del Bluetooth.

la procedura deve essere autorizzata sullo schermo del dispositivo stesso; questa operazione di accesso ha lo scopo di garantire la sicurezza nei confronti di terzi e deve, in genere, essere eseguita soltanto una volta. Il dispositivo "accoppiato" viene salvato e si connetterà automaticamente non appena entrerà nel raggio della piconet (purché il Bluetooth sia attivato).

Il Bluetooth sfrutta onde radio alla frequenza di 2.4 GHz. La frequenza Bluetooth rientra nella banda ISM: ISM sta per "Industrial, Scientific and Medical Bands" e indica una banda nella quale rientrano le apparecchiature ad alta frequenza dell'industria, della scienza, della medicina e degli ambienti di libero impiego. Una banda di frequenza indica una porzione dello spettro elettromagnetico utilizzato per la comunicazione tecnica. [6]

Rispetto ad altre tipologie di rete wireless, come la WLAN, il bluetooth raggiunge una velocità di trasmissione inferiore, per cui l'invio di pacchetti dati di dimensioni più grandi può richiedere più tempo. Tuttavia, per la trasmissione di file singoli e per usi meno complessi, il bluetooth rappresenta senza dubbio la soluzione ideale.

1.1.3 Profili

Ogni chip Bluetooth viene fornito con un cosiddetto "stack di protocollo"; si tratta di un pacchetto software che contiene i servizi legati all'utilizzo di diversi profili Bluetooth. I profili[3] stabiliscono quali tipi di dati possono essere scambiati tra i dispositivi e quali sono, di conseguenza, i servizi fruibili. Per poter usufruire di determinate funzioni è necessario che tutti i dispositivi partecipanti alla piconet supportino lo stesso profilo. [6]

I profili standard presi in esame e sui quali si basano le tecnologie mediche sono: GATT e HDP (Figura 1.2).

[3]I profili supportati da un dispositivo possono essere verificati nei relativi dati tecnici.

Sigla profilo	Nome profilo	Funzione	Dispositivi (esempi)
GATT	Generic Attribute Profile	Trasferimento a bassa energia di quantità ridotte di dati per Bluetooth 4.0 Low Energy	Computer, notebook, smartphone
HDP	Health Device Profile	Collegamento sicuro a dispositivi medici	Telecomandi, dispositivi medici

Figura 1.2: Profili Bluetooth

1.1.4 Usi e Contesti

Il Bluetooth può essere utilizzato in moltissimi modi e contesti. Alcuni esempi: [6]

- Computer e notebook: un chip integrato nell'hardware consente il collegamento di diverse periferiche come mouse, tastiere, auricolari e stampanti.
- Dispositivi mobili: i chip Bluetooth sono integrati in tutti i telefoni cellulari e i tablet moderni e possono, ad esempio, essere utilizzati per sincronizzare file, foto e video.
- Dispositivi audio: molti altoparlanti e auricolari wireless sono dotati di Bluetooth.
- Attrezzature fitness e tecnologie smart home: alcuni fitness tracker e smartwatch inviano i dati tramite Bluetooth direttamente a un'applicazione dedicata sullo smartphone. Anche gli elettrodomestici da cucina, i sistemi di allarme e altri apparecchi smart home supportano la tecnologia Bluetooth.
- Tecnologia medica.
- Industria: la tecnologia di comunicazione wireless mette in rete macchine e impianti di produzione, consentendo così di automatizzare in larga misura i processi produttivi.

1.1.5 Versioni

La prima versione, Bluetooth 1.0a, uscì a metà del 1999, con una velocità di trasmissione dati di 732,2 kbit/s. Tuttavia, proprio come la versione successiva 1.0b, ha dovuto lottare con alcuni difetti iniziali e problemi di sicurezza. Solo con Bluetooth 1.1 (inizio 2001) è riuscita a crearsi una base solida per prodotti commercializzabili. Da allora, il sistema è stato continuamente sviluppato e migliorato con particolare attenzione a: sicurezza, resistenza alle interferenze e velocità di connessione. [6]

Il risultato è una varietà di versioni Bluetooth (Figura 1.3) che si basano l'una sull'altra e si differenziano principalmente per la massima velocità di trasferimento dati possibile, ma anche per le funzionalità e l'area di applicazione.

Versione Bluetooth	Introduzione	Max.velocità di trasferimento dati	Novità principali
Bluetooth 1.0a	Luglio 1999	732,2 kbit/s	Prima versione ufficiale
Bluetooth 1.0b	Dicembre 1999	732,2 kbit/s	Miglioramenti generali
Bluetooth 1.1	Febbraio 2001	732,2 kbit/s	Risolti i problemi di connessione e sicurezza; prima versione del prodotto commercializzabile; crittografia; fino a sette connessioni simultanee
Bluetooth 1.2	Novembre 2003	1 Mbit/s	Retrocompatibilità con Bluetooth 1.1; meno soggetta a disturbi grazie a AFH (Adaptive Frequency Hopping)
Bluetooth 2.0 + EDR	Novembre 2004	2,1 Mbit/s	Velocità di trasmissione dati triplicata grazie a EDR (Enhanced Data Rate); diverse procedure per il risparmio energetico; utilizzo aggiuntivo di NFC (Near Field Communication) durante il pairing
Bluetooth 2.1 + EDR	Agosto 2007	2,1 Mbit/s	Connessione automatica senza PIN grazie a Secure Simple Pairing
Bluetooth 3.0 + HS	Aprile 2009	24 Mbit/s	Ulteriore canale Highspeed (HS) basato su WLAN e UWB (banda ultralarga)
Bluetooth 4.0 LE (anche Bluetooth smart)	Dicembre 2009	24 Mbit/s	Stack di protocollo Low Energy (LE) per diverse procedure a bassa energia (p. es. profilo GATT) per dispositivi di dimensioni ridotte; risoluzione degli errori migliorata; crittografia a 128 bit
Bluetooth 4.1	Dicembre 2013	25 Mbit/s	I piccoli apparecchi non richiedono più un intermediario; IPv6
Bluetooth 4.2	Dicembre 2014	25 Mbit/s	Miglioramenti generali
Bluetooth 5	Dicembre 2016	50 Mbit/s	Portata e velocità di trasmissione dati nettamente superiore

Figura 1.3: Versioni Bluetooth

La versione 4.0 LE ha dato un nuovo impulso a questa tecnologia. Lo stack di protocollo Low Energy ha permesso un risparmio energetico senza precedenti, rendendo il Bluetooth utilizzabile anche sui dispositivi più piccoli come gli smartwatch e le lampadine intelligenti. La versione 4.1 consente ai dispositivi più piccoli di comunicare con altri dispositivi, anche senza un "intermediario". Questo significa, ad esempio, che un bracciale fitness può comunicare direttamente con un cardiofrequenzimetro[4] senza dover passare tramite uno smartphone. La versione 4.2 è caratterizzata da pacchetti dati più piccoli, maggiore velocità, maggiore durata della batteria e maggiore sicurezza. Bluetooth 5, lanciato dal SIG nel 2016, si specializza nei dispositivi IoT e presenta miglioramenti sotto ogni aspetto, rispetto al suo predecessore. L'obiettivo di quest'ultima versione è quello di aumentare la capacità di trasmissione mantenendo un basso livello di consumo energetico e di raggiungere portate fino a 200 metri (all'esterno) e 40 metri (all'interno). Anche se il numero di dispositivi che supportano la nuova versione è al momento abbastanza ridotto, alcuni esperti affermano che il Bluetooth 5 potrebbe addirittura superare la WLAN[5] (almeno nell'area IoT).

1.1.6 Portata massima

A seconda della classe di appartenenza del dispositivo Bluetooth, è permessa una potenza massima di trasmissione diversa, che consente, di conseguenza, un certo raggio (portata). Finora si possono distinguere tre classi, che dipendono dalle esigenze del dispositivo in questione (Figura 1.4).

[4]Dispositivo elettronico per la registrazione della frequenza dei battiti cardiaci.

[5]Propriamente detta rete Wi-Fi, cioè una rete che deve coprire ambienti eterogenei dove le diverse postazioni da collegare non sono necessariamente visibili, infatti possono essere separate da muri o da intercapedini.

Classe	Max.Potenza di trasmissione	Max.Portata (all'interno)	Max.Portata (all'esterno)	Campi d'impiego (esempi)
Classe 1	100 mW	100 m	200 m	Computer, notebook,
Classe 2	2,5 mW	10 m	50 m	Adattatori Bluetooth, computer, notebook
Classe 3	1 mW	1 m	10 m	Dispositivi mobili

Figura 1.4: Classi portata Bluetooth

L'uso della tecnologia Bluetooth richiede sempre un compromesso tra velocità di trasmissione (o portata) e consumo energetico. La versione 4.0, ad esempio, richiede poca energia nella modalità Low Energy, ma raggiunge solo 1 Mbit/s con una portata massima di 10 metri. Nel funzionamento normale, invece, è possibile raggiungere un massimo di 25 Mbit/s, aumentando proporzionalmente sia la portata che il fabbisogno energetico. Solo l'ultima versione, Bluetooth 5 (utilizzabile anche in modalità LE), è in grado di raggiungere una portata elevata, adottando nello stesso tempo misure di risparmio energetico. [6]

I produttori di dispositivi che supportano tecnologia Bluetooth devono, quindi, decidere come configurare i propri prodotti per renderli adatti all'uso cui sono destinati. Nel complesso, la portata massima di un dispositivo Bluetooth dipende dal luogo di utilizzo, se è all'esterno o all'interno (ad esempio all'interno di un appartamento). Il motivo: ostacoli come pareti, mobili di grandi dimensioni o strutture metalliche possono disturbare il collegamento e diminuire drasticamente il raggio di trasmissione. La struttura delle antenne di trasmissione e ricezione, le condizioni della batteria e la sensibilità del dispositivo ricevente sono ulteriori parametri che possono essere decisivi per la portata di una connessione Bluetooth.

La presenza contemporanea di tecnologie wireless, tutte operanti nella congestionata banda dei 2.4 GHz, potrebbe produrre problemi di interferenza.

Diversi metodi, che possono essere raccolti sotto il termine collettivo "Frequency Hopping", hanno ridotto la suscettibilità del Bluetooth alle interferenze per ogni versione. Con questa tecnologia, la banda di frequenza utilizzata viene suddivisa in singoli canali della stessa dimensione. I dispositivi utilizzano tali canali per scambiare dati a velocità considerevole, cambiando il canale di trasmissione diverse migliaia di volte al secondo, seguendo un ordine pseudo-random (a seconda delle necessità) condiviso tra trasmettitore e ricevitore. La Frequency Hopping garantisce così una trasmissione costante e priva di interferenze.

1.1.7 Sicurezza

Grazie alla cifratura dei messaggi e ad altri meccanismi di sicurezza, il Bluetooth è in genere considerato relativamente sicuro. Tuttavia, alcuni mancati accorgimenti possono rendere vulnerabili anche le versioni più recenti dello standard. Il bersaglio più comune è la procedura di accesso durante il pairing, tramite la quale i criminali informatici tentano di ottenere il PIN utilizzato per l'autorizzazione. Dal momento che questo deve essere assegnato una sola volta per ogni configurazione, solitamente la finestra temporale per tali attacchi è estremamente breve. Tuttavia, con un attacco "Bluesmack", gli aggressori disturbano una connessione Bluetooth già esistente e costringono gli utenti ignari a cambiare nuovamente il PIN, che gli aggressori intercettano per poter accedere al rispettivo dispositivo. Questo permette loro di intercettare e manipolare i flussi di dati ("bluesnarfing") e causare danni economici. Affinché un attacco di questo tipo abbia successo, gli aggressori devono trovarsi nelle vicinanze dei dispositivi da decifrare. [6]

Gli utenti Bluetooth possono, tuttavia, prendere alcune precauzioni:

- Assegnare manualmente un PIN per le connessioni Bluetooth.
- Disattivare l'opzione "Secure Simple Pairing" (collegamento automatico a nuovi dispositivi senza PIN) e tornare a stabilire ogni connessione Bluetooth solo manualmente.

- Se il software lo consente, selezionare un codice PIN lungo di almeno otto caratteri numerici.
- Disattivare la "modalità di rilevamento" per anonimizzare il proprio nome utente Bluetooth.
- Evitare di utilizzare il Bluetooth in luoghi affollati, come i luoghi pubblici.
- Memorizzare i dispositivi affidabili nell'elenco dei dispositivi Bluetooth. In questo modo non sarà più necessario inserire nuovamente il PIN quando si stabilisce una connessione in un secondo momento.
- Se una connessione già esistente richiede una nuova autorizzazione tramite PIN, è un segnale di allarme. In questo caso interrompere momentaneamente il tentativo di connessione e uscire dal raggio d'azione.
- Disattivare il Bluetooth immediatamente dopo l'utilizzo.

1.1.8 Bluetooth: prima e dopo

Inizialmente, con l'uscita delle prime versioni, il Bluetooth fu considerato troppo complicato e lento rispetto alla semplice e veloce trasmissione via cavo. Questa visione venne superata in seguito con l'uscita della versione 4.0 a basso consumo energetico, altrimenti detta Bluetooth Low Energy, diventata la forza trainante dell'Internet of things. Gli oggetti "smart" di uso quotidiano (come i fitness tracker, i dispositivi smart home, certi dispositivi medici, ecc.) appartengono al futuro e proprio per questo si presume che il Bluetooth difenderà la sua reputazione di standard industriale per molto tempo ancora.

1.2 Bluetooth Low Energy

Le novità introdotte dalla versione Bluetooth 4.0 portano sul mercato una nuova tecnologia: Il BLE. Il Bluetooth Low Energy, conosciuto anche come Bluetooth Smart (Figura 1.5), è una tecnologia prodotta appositamente per applicazioni che necessitano di trasmissione wireless con un consumo di energia minore rispetto al Bluetooth classico.

Figura 1.5: Logo Bluetooth Smart

Il Bluetooth Low Energy, rilasciato nel 2010, cambia completamente direzione al processo di evoluzione intrapreso fino a quel momento: il punto focale dell'innovazione infatti non è più la velocità di trasferimento e la distanza di copertura, ma il dispendio energetico, insieme alla sicurezza dei trasferimenti ed alla rapidità di setup. Il BLE unisce la praticità del Bluetooth classico e aggiunge un consumo energetico notevolmente inferiore. Esso si rivolge a dispositivi periferici che non richiedono una velocità di trasferimento dati elevata o una trasmissione costante dei dati (gli smartwatch ne sono un esempio).

Il Bluetooth classico e il Bluetooth Low Energy vengono utilizzati per differenti scopi: il Bluetooth può gestire una grande quantità di dati, a discapito del consumo energetico e del costo (entrambi maggiori); [5]

il BLE viene utilizzato per applicazioni che non hanno bisogno di scambiare grandi quantità di dati, a beneficio del consumo energetico e del costo (entrambi minori). Non per questo l'utilizzo di uno è migliore dell'altro, il tutto dipende da quello che si cerca di realizzare. [5]

Al contrario delle versioni rilasciate sino alla 3.0, tutte retrocompatibili, non tutti i device che supportano Bluetooth 4.0 sono retrocompatibili. Questo perchè il BLE rivoluziona in tutto l'architettura alla base della comunicazione: se prima in una comunicazione tra due device vi erano solo i ruoli di master e slave con assegnazione dinamica, il Low Energy definisce altri ruoli a seconda dell'utilizzo e del compito del device stesso, pur mantenendo i due ruoli delle versioni precedenti. L'ampiezza del gruppo di dispositivi che possono comunicare contemporaneamente non è più limitata dal master (il limite precedente era 7 slave per ogni master), ma è definita durante l'implementazione e variabile a seconda dei casi.

1.2.1 Componenti fondamentali: GAP

Il primo componente fondamentale dello stack protocollare BLE è il GAP, acronimo di "Generic Access Profile". Esso definisce due aspetti principali:

- Rilevazione del dispositivo (Device Discovery): descrive la fase di advertising, durante la quale il device, reso visibile al mondo esterno, informa gli altri dispositivi in ascolto della sua esistenza, tramite l'invio di pacchetti dati e informazioni generiche.
- Connessione del dispositivo (Device Connection): descrive la fase di connessione del dispositivo, come viene stabilita e gestita successivamente.

Nello specifico, gli aspetti che vengono affrontati sono:

- Ruoli dei dispositivi.
- Modi in cui operano.

Il GAP definisce due ruoli chiave per i dispositivi che prendono parte ad una connessione: [1]

- Peripheral: dispositivi visibili agli altri, che inviano notifiche di advertising e accettano connessioni in ingresso. Ricoprono questo ruolo i dispositivi di sensoristica.
- Central: dispositivi in grado di ricercare dispositivi visibili e di iniziare la connessione. Ricoprono questo ruolo gli smartphone, i tablet o altri dispositivi che si connettono ai sensori.

Un dispositivo Peripheral non è in grado di iniziare una connessione e un dispositivo Central non è in grado di accettare connessioni in ingresso. Queste funzioni vengono unicamente svolte dai rispettivi ruoli di competenza.

Il GAP introduce due ulteriori ruoli: [1]

- Broadcaster: ruolo assunto dagli oggetti che hanno l'unica funzione di trasmettere dati, senza supportare connessioni. Il Broadcaster usa notifiche di advertising per trasmettere le informazioni ai device in ascolto.
- Observer: ruolo assunto dagli oggetti che hanno l'unica funzione di ricevere dati, complementare al Broadcaster. La connessione non è supportata ed il device rimane semplicemente in ascolto.

In generale, due dispositivi che ricoprono lo stesso ruolo non sono in grado di connettersi l'un altro.

Un dispositivo Peripheral che si rende visibile vuole informare un dispositivo Central della propria presenza. Questa fase prende il nome di "advertising" e vuole fare in modo che il dispositivo Central possa individuare il dispositivo visibile e connettersi ad esso.

Durante la fase di advertising il dispositivo Peripheral invia al dispositivo Central i seguenti pacchetti di dati: [1]

- Advertising Data payload.
- Scan Response payload.

L'Advertising Data payload è obbligatorio, poiché viene trasmesso costantemente dal dispositivo Peripheral al dispositivo Central per informarlo della sua esistenza. Lo Scan Response payload è opzionale, poiché viene trasmesso dal dispositivo Peripheral al dispositivo Central, se quest'ultimo lo richiede. Il dispositivo Central manda una Scan Request per richiedere informazioni aggiuntive (come il nome del dispositivo) al Peripheral in questione, il quale è tenuto a rispondere con una Scan Response adeguata. Entrambi i payload[6] possono contenere fino a 31 bytes di dati, consentendo di inviare solo piccole quantità di dati per ogni pacchetto. La seguente illustrazione (Figura 1.6) spiega come avviene il processo di advertising.

[6]Con payload si intende la porzione di dati "utili" all'interno del pacchetto, il flusso di dati effettivamente trasmesso.

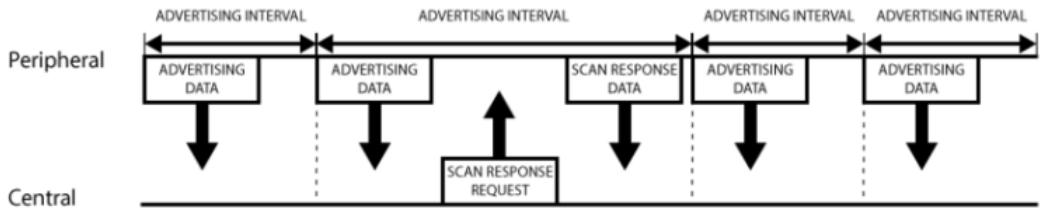

Figura 1.6: Processo di advertising: uno a uno

Un Peripheral imposta uno specifico intervallo di advertising, durante il quale vengono inviati i pacchetti. Ogni volta che si passa da un intervallo al successivo, il dispositivo Peripheral ritrasmette il pacchetto di advertising ("Advertising Data"). Se un dispositivo Central in ascolto è interessato a richiedere alcune informazioni aggiuntive al dispositivo che sta facendo advertising, esso invia una Scan Request ("Scan Response Request"). Il dispositivo Peripheral risponde inviando un pacchetto di dati ("Scan Response Data") contenente le informazioni richieste.

In una prospettiva più ampia, durante il processo di advertising, un dispositivo BLE Peripheral può inviare dati ad ogni dispositivo Central in ascolto all'interno del range di azione, come mostrato nell'illustrazione seguente (Figura 1.7).

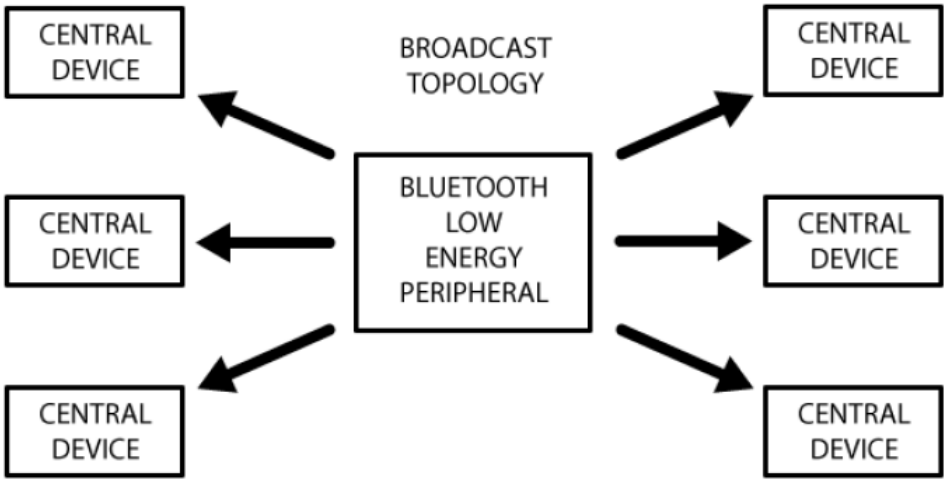

Figura 1.7: Processo di advertising: uno a molti

Questo schema prende il nome di Broadcast Network e mostra l'invio unidirezionale e contemporaneo di dati dal dispositivo Peripheral ai dispositivi Central in ascolto.

Oltre ai ruoli dei dispositivi, il GAP descrive anche i modi differenti in cui essi operano. Un modo è uno stato che un dispositivo può cambiare per raggiungere un certo obiettivo, quindi può essere temporaneo. Tra questi troviamo:

- Broadcast mode: il dispositivo invia dati ma non permette di stabilire una connessione.
- Discoverable mode: il dispositivo è visibile agli altri e accetta connessioni in ingresso.
- Connect mode: il dispositivo periferico si è connesso ad un dispositivo centrale.
- Bonding: il dispositivo è accoppiato ("legato") ad un altro, permettendo una connessione sicura.

Una volta che il device Central ha richiesto informazioni aggiuntive al device Peripheral può stabilire una connessione. Stabilita la connessione, il processo di

advertising si stoppa e il dispositivo periferico non è più in grado di inviare i pacchetti sopra descritti. Da questo momento inizia la fase di connessione, durante la quale la comunicazione e lo scambio di dati tra i dispositivi (Peripheral e Central) avviene in maniera bidirezionale attraverso l'utilizzo di Servizi e Caratteristiche GATT.

1.2.2 Componenti fondamentali: GATT

Un altro componente fondamentale dello stack è il GATT, acronimo di "Generic Attribute Profile". Esso è un'interfaccia software che definisce come i dispositivi possano inviare e ricevere dati, descrivendo i concetti di Servizio e Caratteristica. Il GATT sfrutta a sua volta l'ATT (Attribute Protocol), un protocollo generico che viene utilizzato per memorizzare i dati dei Servizi e delle Caratteristiche che il GATT mette a disposizione all'esterno. [1]

Il GATT entra in gioco una volta che la connessione tra due dispositivi è stata stabilita, ossia dopo aver passato il processo di advertising (governato dal GAP). In questa fase successiva, il device Central diventa il Master della piconet mentre il device Peripheral diventa uno Slave.

Le connessioni GATT sono esclusive. Questo significa che un dispositivo BLE Peripheral può essere connesso con solo un dispositivo Central alla volta. Non appena un dispositivo periferico si connette a un dispositivo centrale, le notifiche di advertising verranno stoppate.

Stabilire una connessione è l'unico modo per permettere una comunicazione bidirezionale tra i dispositivi, dove il dispositivo centrale può inviare dati al dispositivo periferico e viceversa. Il diagramma seguente (Figura 1.8) spiega il modo in cui i dispositivi BLE lavorano in un ambiente connesso.

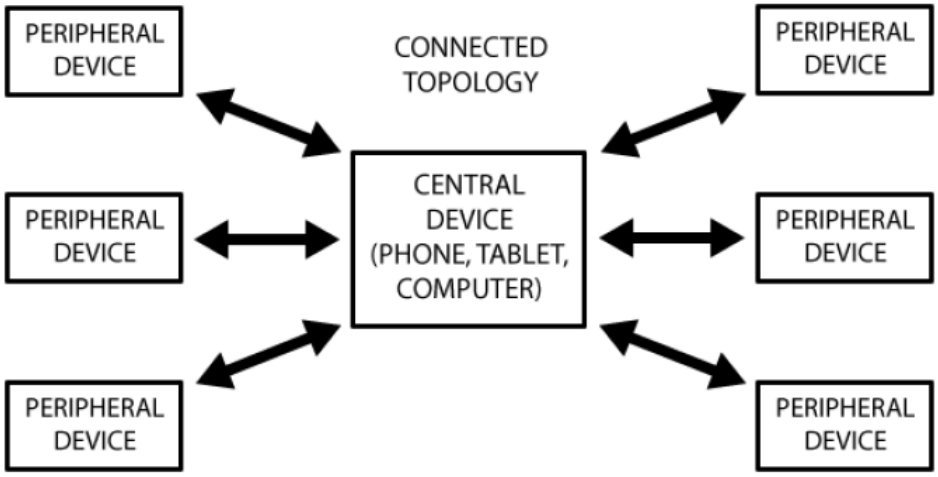

Figura 1.8: Processo di connessione: uno a molti

Questo schema prende il nome di "Connected Network" e mostra lo scambio bidirezionale di dati tra un dispositivo Central e i dispositivi Peripheral connessi. Un dispositivo Peripheral può essere connesso con solo un dispositivo Central alla volta, ma il dispositivo centrale può essere connesso a dispositivi periferici multipli. Se due dispositivi Peripheral vogliono comunicare tra di loro deve essere implementato un sistema mailbox personalizzato in cui tutti i messaggi passano attraverso il dispositivo centrale.

Il GATT definisce i ruoli che i dispositivi interagenti possono assumere:

- Client: il GATT Client invia richieste a un Server e riceve risposte da esso. Il GATT Client non sa nulla in anticipo sulle Caratteristiche del Server, quindi deve prima informarsi sulla presenza e sulla natura di tali Caratteristiche, rilevando il Servizio che le contiene. Dopo aver individuato il Servizio, il Client può iniziare a leggere o scrivere le Caratteristiche trovate nel Server.

- Server: il GATT Server riceve richieste da un Client e invia le risposte. È responsabile della memorizzazione e della messa a disposizione dei dati, organizzati in Caratteristiche. Il GATT Server comunica le Caratteristiche, ossia i dati di cui dispone (es. le misurazioni effettuate da un sensore), raggruppate in contenitori logici chiamati Servizi.

Ogni transazione di dati avviene quindi tra un GATT Server (il dispositivo periferico) e un GATT Client (il dispositivo centrale).

Il diagramma seguente (Figura 1.9) illustra come avviene il processo di scambio dati tra il GATT Server e il GATT Client.

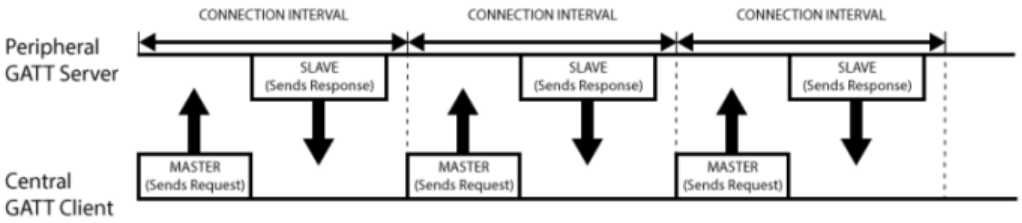

Figura 1.9: Processo di connessione: uno a uno

Le transazioni iniziano solitamente con una richiesta dal dispositivo Master, il GATT Client, che riceve risposta dal dispositivo Slave, il GATT Server. Al momento della connessione, il dispositivo periferico suggerisce un intervallo di connessione al dispositivo centrale, durante il quale avvengono le transazioni (richiesta e risposta) tra i due dispositivi. Il dispositivo centrale proverà a riconnettersi ad ogni intervallo per verificare se sono disponibili nuovi dati. I dispositivi mantengono in piedi la connessione scambiandosi periodicamente dati anche se non c'è nulla da trasmettere. Il dispositivo centrale potrebbe non essere in grado di gestire una transazione perchè impegnato a parlare con un'altra periferica, oppure perchè le risorse di sistema non sono disponibili.

1.2.3 Profili, Servizi e Caratteristiche

Lo scambio di dati tra dispositivi è basato su oggetti di alto livello che prendono il nome di Profili, Servizi e Caratteristiche. Questi oggetti vengono messi a disposizione dal GATT e, come si può notare nell'illustrazione seguente (Figura 1.10), sono organizzati in gerarchia. Si può accedere ad ogni oggetto solo nelle modalità previste dal progettista (sola lettura, sola scrittura, entrambe o nessuna).

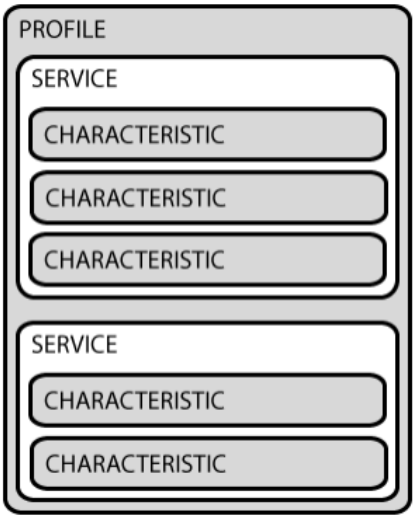

Figura 1.10: Schema dell'interfaccia messa a disposizione dal GATT

Un Profilo specifica il funzionamento di un dispositivo in una particolare applicazione. Un Profilo è una collezione predefinita di Servizi, compilata dal SIG o dal progettista della periferica. Il SIG definisce diversi Profili per dispositivi BLE: Alert Notification Profile, Blood Pressure Profile, Object Transfer Profile ecc.

I Servizi sono sezioni contenute nel Profilo che mettono a disposizione informazioni di vario tipo. Ogni Servizio viene identificato da un UUID e funge da contenitore per una serie di Caratteristiche. Prendendo come esempio il Blood Pressure Profile, esso include due Servizi: Blood Pressure Service e Device Information Service.

Le Caratteristiche sono gli oggetti di livello più basso. Ognuna di esse incapsula un puntatore ai dati (dati singoli di varia lunghezza, come numeri interi o decimali, oppure array di valori). Analogamente ai Servizi, ogni Caratteristica viene identificata da un UUID. Le caratteristiche Blood Pressure Measurement e Blood Pressure Feature sono caratteristiche obbligatorie del Blood Pressure Service.

Una Caratteristica può contenere uno o più Descrittori: attributi che descrivono il valore di una Caratteristica. Ad esempio, un descrittore potrebbe specificare una descrizione, un intervallo accettabile per il valore di una Caratteristica o un'unità di misura del valore di una Caratteristica. [2]

1.2.4 Universally Unique IDentifier (UUID)

I dati relativi a Servizi e Caratteristiche vengono salvati in una tabella di ricerca (lookup table[7]) usando un identificatore lungo 16 byte (128 bit), chiamato UUID. Di questi 16 byte, i primi 4 vengono scelti dal programmatore mentre gli altri 12 byte costituiscono l'UUID base (Figura 1.11), uguale per tutte le entità definite dal SIG. [1]

```
0000-1000-8000-00805F9B34FB
```

Figura 1.11: UUID base dell'UUID completo

Poiché nel BLE è importante limitare al massimo la quantità di dati trasmessi, il SIG ha stabilito un formato abbreviato per ogni UUID. Ad esempio, se il Servizio Blood Pressure viene rappresentato dal seguente (Figura 1.12) UUID completo (128 bit), l'UUID abbreviato sarà 1810 (16 bit).

[7]Struttura dati (array o matrice) concepita per sostituire un calcolo a runtime con una semplice e veloce operazione di consultazione.

Figura 1.12: UUID completo del Blood Pressure Service

In questo modo, per identificare Servizi e Caratteristiche, non è necessario trasmettere ogni volta l'intero UUID, ma è sufficiente comunicare il valore a 16 bit che li rappresenta (alias dell'UUID completo).

I formati abbreviati possono essere utilizzati solo con gli UUID associati alle entità standard, definite dal SIG, ovvero con gli UUID che derivano dalla forma base. Per gli UUID personalizzati, è a carico del programmatore generare un codice univoco a 128 bit per rappresentare l'entità voluta, il quale deve essere comunicato integralmente per fare riferimento all'entità stessa.

1.2.5 Android e BLE

La prima versione di Android a supportare completamente le specifiche del Bluetooth Low Energy è la versione 4.3 Jelly Bean[8].

Le classi principali da utilizzare nella programmazione di un'applicazione Android in grado di interagire con il modulo BLE sono: [17, 18]

- BluetoothGatt: fornisce le API per interagire con i Profili del GATT. Permette, tra le altre cose, di effettuare la connessione, di analizzare i Servizi e le Caratteristiche di un dispositivo connesso e di leggere o scrivere dati.

- BluetoothGattService: rappresenta un Servizio. Contiene i metodi atti a ottenere la collezione di Caratteristiche che si trovano al suo interno.

[8] API level 18, requisito minimo per il funzionamento dell'applicazione.

- BluetoothGattCharacteristic: rappresenta una Caratteristica. Contiene i metodi per leggere e/o scrivere i dati (a seconda dei permessi forniti dalla Caratteristica stessa).
- BluetoothAdapter: fornisce le API per interagire con il modulo Bluetooth del device. Consente, per esempio, di effettuare la ricerca di dispositivi (in modalità classica o in modalità BLE).

Per poter utilizzare le funzionalità del Bluetooth è necessario fornire all'applicazione alcuni permessi, inserendoli nel manifest (Figura 1.13).

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Figura 1.13: Permessi necessari all'applicazione per poter utilizzare il Bluetooth

Il permesso BLUETOOTH è necessario per eseguire qualsiasi comunicazione Bluetooth, come richiedere una connessione, accettare una connessione e trasferire dati. Inoltre, per avviare la rilevazione dei dispositivi o manipolare le impostazioni Bluetooth, è necessario il permesso BLUETOOTH_ADMIN.

È possibile limitare il funzionamento dell'applicazione ai soli dispositivi in grado di supportare il Bluetooth Low Energy settando l'attributo "required" della feature seguente a "true" (Figura 1.14). In questo modo l'app non potrà essere installata su dispositivi senza BLE. [18]

```
<uses-feature android:name="android.hardware.bluetooth_le"
    android:required="true"/>
```

Figura 1.14: Verifica dell'esistenza di un modulo BLE integrato nel dispositivo Android

Per interagire con un dispositivo BLE in background, così da mantenere una connessione esistente e continuare a eseguire le operazioni di accesso anche quando

l'app non è in primo piano, viene aggiunto un Service[9]. Il Service deve essere dichiarato nel manifest come componente dell'applicazione.

Tramite la classe BluetoothManager è possibile ottenere un'istanza del BluetoothAdapter per avviare la ricerca di altri dispositivi nelle vicinanze. In particolare, BluetoothAdapter fa uso di un altro oggetto: BluetoothLeScanner, che si occupa effettivamente di rilevare i dispositivi BLE che stanno facendo advertising. È possibile configurare la scansione eseguita da BluetoothLeScanner e assegnargli delle regole di filtro da applicare in modo che vengano selezionati solo i dispositivi di interesse. [17, 18]

Poiché i dispositivi sono generalmente alimentati a batteria (e comunque date le peculiarità di risparmio energetico del BLE) è consigliabile eseguire la scansione solo per un lasso di tempo prefissato e stopparla una volta che il dispositivo di interesse è stato individuato. Tale operazione è infatti pesante dal punto di vista del consumo energetico.

È importante notare che la ricerca può essere fatta solo per dispositivi BLE oppure per dispositivi Bluetooth classici. Non è possibile eseguire contemporaneamente la ricerca di dispositivi BLE e classici.

In caso di verifica di un evento particolare, tra cui il riconoscimento di un dispositivo o l'interruzione imprevista dell'operazione di scanning, verrà richiamata una callback. Se un dispositivo viene individuato è possibile stabilire una connessione con esso (GATT Server) e ottenere un'istanza BluetoothGatt del dispositivo stesso.

Attraverso l'oggetto BluetoothGatt, l'applicazione Android (ovvero il GATT Client) può condurre operazioni di ricerca sui Servizi e le Caratteristiche che il dispositivo

[9]Un Service è un componente dell'applicazione che può eseguire operazioni di lunga durata in background. Un Service continua a eseguire in background anche se l'utente passa a un'altra applicazione.

Server espone all'esterno. Le istanze BluetoothGattService e BluetoothGattCharacteristic fanno riferimento al Service o alla Caratteristica con UUID specificato. Una volta che l'app si è connessa al GATT Server e ha scoperto i Servizi e le Caratteristiche che offre, essa può leggere o scrivere i dati.

Quando l'app termina di utilizzare un dispositivo BLE, è buona norma chiudere la connessione col dispositivo, in modo che il sistema possa rilasciare le risorse in modo appropriato.

Le operazioni da eseguire in un'applicazione Android che vuole fare uso delle funzionalità BLE si possono così riassumere:

- Dichiarare i permessi nel manifest.
- Aggiungere un Service Android per mantenere attive le operazioni BLE in background.
- Cercare i dispositivi periferici BLE nelle vicinanze.
- Connettersi alla periferica BLE individuata.
- Scoprire i Servizi e le Caratteristiche della periferica BLE.
- Eseguire l'accesso alle Caratteristiche.

1.2.6 Perchè scegliere Bluetooth Low Energy

Per un'applicazione come quella descritta in questo elaborato, una delle caratteristiche fondamentali è il basso consumo di energia. Il sensore dovrà, infatti, essere alimentato a batteria e un limitato consumo energetico permette di aumentarne la durata. L'interfaccia sviluppata deve essere, inoltre, universale, sia nel senso che deve permettere l'utilizzo di diversi tipi di sensore, sia nel senso che deve essere compatibile con il più grande numero di dispositivi possibile. È quindi evidente che la migliore tecnologia è quella del Bluetooth Low Energy.

Capitolo 2

Analisi e Progettazione

In questo capitolo verranno illustrati:

- Il contesto lavorativo.
- La descrizione generale dell'applicazione.
- I requisiti imposti dall'azienda.
- La componente da sviluppare:
  - Funzionalità da implementare.
  - Ulteriori requisiti.
  - Sessione tipica di utilizzo.
  - Strumenti utilizzati per lo sviluppo.
  - Profili, Servizi e Caratteristiche.

2.1 Contesto lavorativo

Ho sviluppato il progetto, alla base di questo elaborato, durante il periodo di tirocinio svolto presso l'azienda Finmatica, a Bologna. Finmatica è un gruppo societario che nasce con l'obiettivo di innovare i processi degli enti privati e delle organizzazione pubbliche, progettando soluzioni software all'avanguardia che semplificano la gestione aziendale. L'azienda opera principalmente in 2 campi: pubblica amministrazione e sanità. È proprio in quest'ultimo campo che sono stato inserito per contribuire ad un progetto sperimentale, già avviato, ma in via di estensione e miglioramento.

Il gruppo di lavoro, con il quale ho partecipato a portare avanti questo progetto, è stato delegato alla realizzazione di un sistema di profilazione e monitoraggio dell'utente, che integri: la raccolta di parametri vitali tramite dispositivi medici (mobili e indossabili), la storicizzazione, l'andamento e la valutazione dei parametri, la segnalazione di strategie di miglioramento. L'idea è quella di sfruttare dei dispositivi di rilevazione (ad esempio un fitness tracker, una bilancia, un monitor della pressione, ecc.) per acquisire, tramite connessione bluetooth, informazioni utili alla profilazione quotidiana dell'utente che utilizza il sistema.

Il lavoro è stato commissionato dall'Azienda Usl di Piacenza, la quale prevede di utilizzare il sistema per il controllo periodico dei propri pazienti, in cura presso il PDTA[1] "Disturbi del comportamento alimentare". I disturbi del comportamento alimentare (abbreviato DCA) riguardano tutte quelle problematiche che concernono il rapporto tra gli individui e il cibo. Sono uno dei problemi di salute più comuni negli adolescenti e nei giovani adulti, fino ad arrivare, specie negli ultimi anni, ai preadolescenti e ai bambini. Vengono seguiti pazienti affetti da anoressia nervosa, bulimia, disturbi da alimentazione incontrollata e disturbi non altrimenti specificati.

[1] Percorso Diagnostico Terapeutico Assistenziale.

Come ci è stato comunicato dall'azienda, il vero problema dei DCA, al giorno d'oggi, non è tanto la cura della patologia in sè, ma il fatto che restano spesso problemi sommersi, ovvero problemi dei quali i pazienti non parlano volentieri o non parlano affatto. Tuttavia i DCA, se non trattati in tempi e con metodi adeguati, possono diventare una condizione permanente e compromettere seriamente la salute dell'individuo. Per questo motivo, nella moderna concezione di pratica medica, il PDTA vuole tenere conto di tutti gli aspetti necessari per la presa in carico globale, non solo della malattia, ma anche della persona e dei suoi bisogni; valorizzando perciò gli aspetti relazionali e comunicativi del percorso di cura.

Con questo progetto l'azienda sanitaria di Piacenza, in accordo con Finmatica, vuole sperimentare una soluzione innovativa per la cura dei propri pazienti, il cui target principale sono i bambini e i preadolescenti. Si vuole garantire una continuità della cura, anche al di fuori degli orari di visita e dell'area ospedaliera, oltre a un sistema di feedback immediato sull'andamento del percorso del paziente.

2.2 Descrizione generale dell'applicazione

Il gruppo di lavoro, incaricato allo svolgimento di questo progetto, ha iniziato a sviluppare un'applicazione mobile che venisse il più possibile incontro alle necessità dell'azienda. L'applicazione è un sistema multiutente che prevede il monitoraggio dei passi giornalieri e di alcuni parametri vitali (frequenza cardiaca e peso), oltre alla possibilità di inserire foto dei pasti e ottenere un riscontro sull'andamento del percorso.

L'accesso è soggetto ad autenticazione con username e password criptati su Server aziendale. I profili utente previsti sono 3:

- Paziente.
- Medico.

- Amministratore.

Il paziente (bambino in età pediatrica) può svolgere le seguenti attività:

- Registrare e sincronizzare con l'applicazione i passi e la frequenza cardiaca, rilevati giornalmente tramite l'utilizzo di un fitness tracker (o fitband) apposito fornito al paziente stesso durante il percorso di cura.
- Inserire manualmente il peso corporeo.
- Visualizzare le misurazioni rilevate e inserite (nel caso del peso), classificate per data e ora.
- Scattare foto dei pasti direttamente dall'applicazione, le quali vengono salvate e organizzate in una raccolta prestabilita.
- Inserire informazioni aggiuntive riguardo la foto scattata: tipo di pasto (colazione, pranzo, merenda, cena) e note aggiuntive (spazio dove l'utente può inserire alcune considerazioni riguardo la foto).
- Visualizzare le foto scattate e le relative informazioni (data e ora, tipo di pasto, note inserite). Il paziente può anche ricevere dal medico dei feedback sulle foto, i quali possono essere visualizzati come commenti sotto le foto stesse.
- Visualizzare il proprio profilo utente con i relativi dati anagrafici.

Per garantire una continuità nel percorso di cura, l'applicazione permette una lettura da remoto del paziente. Questa lettura viene effettuata dal medico di competenza, il quale fa utilizzo dell'applicazione per svolgere le seguenti attività:

- Visualizzare i dati raccolti dal paziente e tenerli costantemente monitorati.
- Inviare feedback al paziente sull'andamento dei parametri forniti (passi, frequenza cardiaca e peso) e sulle foto dei pasti caricate.

Infine, l'amministratore si occupa di gestire gli altri utenti e l'applicazione stessa.

Il sistema di feedback implementato consente un'interazione (al momento unidirezionale) tra paziente e medico: il medico invia un commento al paziente, quest'ultimo lo riceve e lo visualizza. In questo modo, il bambino che viene seguito, sapendo di ricevere un feedback (che può essere semplicemente una faccina o un commento più articolato), può essere incoraggiato a seguire la cura a dovere.

L'integrazione dei parametri vitali del paziente è permessa dall'interazione tra il dispositivo su cui viene eseguita l'applicazione e una tecnologia di misurazione esterna. In particolare:

- L'applicazione verrà eseguita su smartphone o su tablet con sistema operativo Android e versione minima 4.3 (Jelly Bean), che implementa quindi le specifiche del Bluetooth Low Energy.
- La tecnologia esterna utilizzata è un fitness tracker Fitpolo BLE, modello H701.

Ne deriva, quindi, che il metodo di connettività utilizzato per il trasferimento dei dati, dal dispositivo di misurazione all'applicazione, è quello del Bluetooth Low Energy.

L'utente avvia la connessione dei dispositivi (smartphone e Fitpolo) direttamente dall'applicazione, cliccando manualmente sul relativo button. Il processo di connessione prevede i seguenti passi:

1. Verifica che il bluetooth sul fitness tracker sia attivo.
2. Scansione dei dispositivi nelle vicinanze.
3. Individuazione del dispositivo da collegare (Fitpolo con indirizzo MAC specifico).

4. Pairing[2] dei due dispositivi (se non ancora eseguito).

5. Trasmissione e ricezione dei dati sull'applicazione.

6. Disconnessione.

Nella fase di individuazione del dispositivo, il fattore determinante per rintracciare il fitness tracker specifico, utilizzato dal paziente, è l'indirizzo MAC associato al dispositivo stesso. L'indirizzo MAC (abbreviazione di "Media Access Control") è l'indirizzo hardware univoco di una singola scheda di rete inserita nel dispositivo. Questo indirizzo fisico viene assegnato direttamente dal produttore dell'hardware e viene utilizzato per identificare un dispositivo nella rete. L'indirizzo MAC viene impostato sull'applicazione dall'amministratore, in modo che questa sia in grado di rintracciare ogni volta il dispositivo BLE con indirizzo specificato.

Se la connessione è avvenuta con successo il fitness tracker vibrerà per 3 volte e l'applicazione confermerà l'esito positivo della connessione. Ogni volta che il Fitpolo viene connesso con successo, l'applicazione preleva i dati relativi a frequenza cardiaca e passi rilevati. I dati ricevuti vengono memorizzati su Server aziendale e visualizzati direttamente sull'applicazione, raggruppati in ordine storico.

Per permettere all'utente, sia medico che paziente, di effettuare le attività di competenza (elencate in precedenza) in modo rapido e intuitivo, l'applicazione utilizza un sistema di navigazione semplice per passare da una schermata all'altra. Ogni schermata è riferita ad una specifica sezione e presenta tutti gli elementi grafici per consentire all'utente di svolgere le attività.

---

[2]Il pairing (tradotto "accoppiamento") indica quel processo di reciproco riconoscimento che viene effettuato quando due dispositivi Bluetooth vengono collegati. Consiste nello scambio e verifica di un codice d'identificazione al fine di autorizzare lo scambio di dati tra i dispositivi stessi. È necessario eseguire il pairing solo la prima volta. Per effettuare la connessione le volte successive l'operazione di pairing non è più necessaria.

L'applicazione è stata progettata come soluzione sperimentale, su richiesta dell'AUSL di Piacenza, per gestire al meglio il percorso di cura dei propri pazienti. Per questo motivo l'app non verrà pubblicata sul Play Store di Google o su App Store alternativi, ma al contrario verrà fornita all'azienda sanitaria sottoforma di applicazione preinstallata. L'idea è quella di fornire al paziente, oltre al fitness tracker per la rilevazione dei parametri analizzati, un dispositivo mobile (tablet o smartphone) con l'applicazione già installata in modalità Kiosk.

Questa modalità permette agli utenti di interagire solo con l'applicazione, senza avere accesso alle altre funzionalità fornite dal dispositivo mobile. L'applicazione continua a mostrare i propri contenuti in maniera permanente, bloccando tutte le altre funzioni, ad eccezione di quelle relative all'applicazione stessa. Questo perchè avere troppe funzioni disponibili sul dispositivo può distrarre il paziente, che, trattandosi appunto di un bambino, potrebbe distrarsi dal suo reale obiettivo e quindi influire sulla terapia. Al medico curante verrà fornita l'applicazione con la modalità Kiosk disattivata.

La "Kiosk mode" permette, quindi, oltre a una maggiore sicurezza, di avere meno distrazioni possibili, garantendo che i dispositivi vengano utilizzati limitatamente allo scopo per cui sono stati pensati.

Per quanto riguarda l'aggiornamento dell'app, è stato implementato un meccanismo che agisce sia a livello client-side che server-side per poter effettuare aggiornamenti dell'app da remoto. Ogni volta che viene sviluppata una nuova versione e viene generato l'apk relativo, questo viene inserito tramite ssh[3] nel server, sovrascrivendo l'apk della versione precedente. Dopo ogni login avvenuto con successo, il Server manda all'applicazione il nome dell'ultima versione dell'app, la quale viene confrontata con la versione attuale.

[3]SSH (acronimo di Secure SHell) è un protocollo di rete che consente di collegarsi da remoto (tramite interfaccia a riga di comando) ad un altro host. La connessione viene stabilita in tutta sicurezza.

Se le due versioni non coincidono viene richiesto all'utente di aggiornare l'applicazione per proseguire. Premendo il pulsante di aggiornamento viene chiamato un servizio REST per ottenere l'apk aggiornato dal Server. Successivamente l'apk viene scaricato e una volta finito il download viene installato.

2.3 Requisiti imposti dall'azienda

Con l'entrata di nuovi tirocinanti in azienda, Finmatica ha valutato l'idea di estendere le funzionalità fornite dall'applicazione, sia per consentire un'interazione da parte dell'utente sempre più semplice e immediata, sia per permettere un monitoraggio più completo dei parametri vitali del paziente.

Una prima funzionalità che si vuole migliorare è quella dell'acquisizione dei parametri vitali. I valori, che vengono misurati dal paziente stesso, devono essere il più precisi possibili, così da permettere al medico curante di effettuare una diagnosi corretta e proficua. Inoltre, trattandosi di un procedimento metodico che il paziente deve eseguire giornalmente, la sincronizzazione dei dati (misurati dai dispositivi medici) con l'applicazione deve essere diretta, cercando di minimizzare i passaggi intermedi.

Questo obiettivo viene parzialmente raggiunto dal fitness tracker che, tramite il click di un semplice button sull'applicazione, permette un'integrazione immediata dei dati rilevati dal dispositivo stesso, sfruttando la connessione BLE che viene stabilita. Al contrario, per quanto riguarda l'integrazione del peso corporeo, il paziente deve prima pesarsi sulla bilancia, poi scrivere manualmente il valore rilevato nell'apposita sezione dell'applicazione. Questo procedimento può portare ad un errore umano, dovuto a distrazione o dimenticanza, che porta il paziente ad inserire un valore non corretto, o quantomeno non preciso.

Per permettere una sincronizzazione dei dati il più diretta possibile, agevolando l'utente sia nel procedimento (ancora più immediato) che dal commettere errori,

si vuole inserire una funzione di sincronizzazione automatica dei dispositivi con l'applicazione. In questo modo, nel caso del fitness tracker, la sincronizzazione verrà effettuata, ad esempio, quando il paziente effettua il login con l'applicazione o durante l'utilizzo dell'applicazione stessa. Infatti, trattandosi di un dispositivo indossabile, durante l'interazione con l'applicazione si presuppone che il paziente lo indossi al polso e, in tal caso, verrà individuato facilmente durante la scansione dei dispositivi BLE. Nel caso di misurazione del peso, invece, per evitare di scrivere manualmente i dati e, al contrario, acquisirli in maniera automatica, sarà necessario disporre di una bilancia dotata di chip Bluetooth, che permetta di stabilire una connessione con l'applicazione e trasferire i dati. Ne consegue che un altro requisito sia quello di estendere i dispositivi medici Bluetooth supportati dall'applicazione, con l'obiettivo di acquisire e monitorare ulteriori parametri vitali relativi al paziente che fa uso dei dispositivi stessi.

A tal proposito ci è stato richiesto di effettuare una ricerca dei dispositivi che potessero integrarsi con l'applicazione e che fornissero l'sdk per interagire con essi. Il rapporto ideale tra semplicità d'uso e trasferimento automatico dei dati, oltre alla possibilità di usufruire di API specifiche, l'abbiamo individuato nei dispositivi prodotti dalla società A&D.

A&D è un'azienda giapponese specializzata nella produzione di strumenti di misura per uso aziendale, industriale, educativo e sanitario. L'azienda è presente nel mercato internazionale da oltre 40 anni e tutta la strumentazione è sottoposta a protocolli di certificazione della qualità e di validazione clinica secondo gli standard internazionali. Con la linea di prodotti Wellness Connected, A&D si pone all'avanguardia nel mercato consumer dei dispositivi medici che trasmettono dati a Smartphone e Tablet. I prodotti di questa linea vengono distinti, in base al tipo di comunicazione, in: prodotti Bluetooth Smart (o BLE) e prodotti NFC.

I dispositivi scelti devono mantenere un consumo energetico ridotto, in vista di un'alimentazione a batteria, e nello stesso tempo devono avere una portata di trasmissione medio-ampia.

È importante, inoltre, che l'applicazione sia il più universale possibile, in modo che possa essere utilizzata con dispositivi medici diversi senza rendere necessaria una modifica complessa al software.

Dal momento che la struttura dell'applicazione, implementata fino a quel momento, è stata pensata per l'utilizzo di un solo dispositivo, sia per quanto riguarda l'acquisizione dei dati che per la loro visualizzazione, mi è stato richiesto di creare una componente autonoma che permettesse l'integrazione e la visualizzazione dei dati di più dispositivi e che rispettasse i requisiti di universalità e sincronizzazione automatica. La componente, progettata inizialmente come applicazione autonoma, dovrà funzionare su un qualunque dispositivo Android (versione minima 4.3) e verrà integrata nell'applicazione sperimentale con release successive.

2.4 Componente da sviluppare

Il mio obiettivo è quello di creare un'applicazione autonoma che permetta al paziente, tramite l'utilizzo di alcuni dispositivi medici, di integrare automaticamente i dati (rilevati dai dispositivi stessi) al termine della misurazione. L'applicazione deve permettere al paziente di visualizzare tutti i valori misurati e fornire un'interfaccia utente intuitiva e funzionale.

Con questo primo sviluppo, si vuole permettere al paziente di misurare i seguenti parametri: peso, pressione arteriosa minima (diastolica), pressione arteriosa massima (sistolica) e pulsazioni. A tale scopo vengono utilizzati due dispositivi medici A&D: un misuratore di pressione e una bilancia. Entrambi dispongono di tecnologia Bluetooth Low Energy, rispettando i requisiti di basso consumo energetico e portata medio-ampia.

L'interfaccia e la struttura dell'applicazione devono essere universali e scalabili, al fine di rendere possibile l'integrazione con più dispositivi e quindi una raccolta e visualizzazione di dati più ampia.

2.4.1 Funzionalità da implementare

Per ogni dispositivo è necessario implementare le seguenti funzionalità:

- Pairing del dispositivo.
- Sincronizzazione dei dati.
- Visualizzazione dei dati.

L'utente che interagisce con l'applicazione dovrà essere in grado di eseguire ognuna di queste funzionalità (Figura 2.1), le quali a loro volta includeranno una serie di passaggi in back-end.

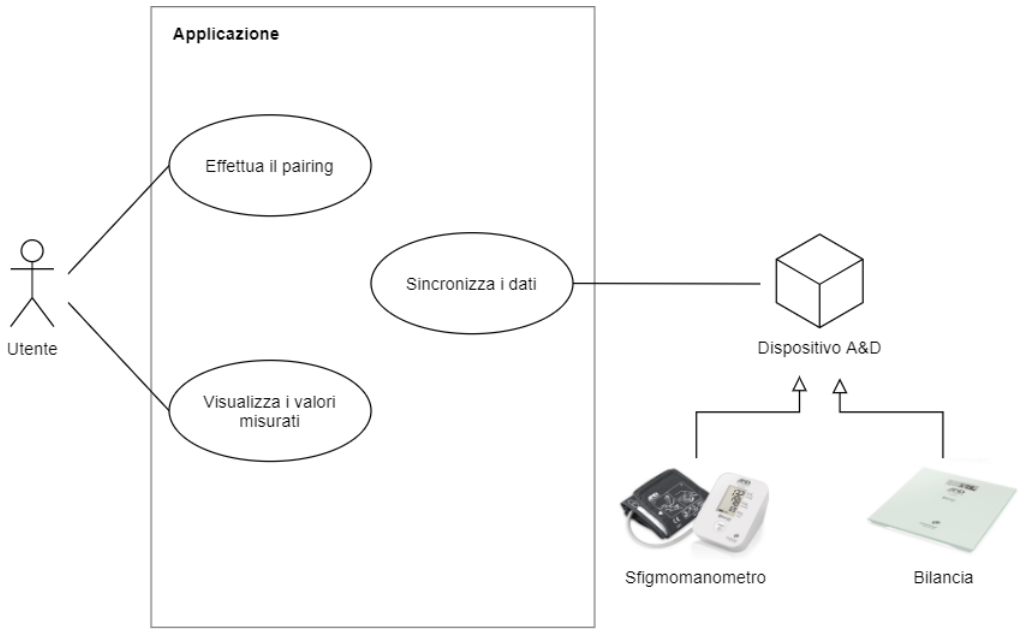

Figura 2.1: Diagramma dei casi d'uso dell'applicazione

Pairing del dispositivo

Il pairing del dispositivo è necessario per effettuare la sincronizzazione dei dati e viene eseguito generalmente una sola volta. Tale operazione viene avviata tramite due comandi impartiti dall'utente:

- Click di un'icona sull'interfaccia grafica.
- Click prolungato sul pulsante hardware del dispositivo.

In particolare, per effettuare il pairing il paziente deve:

1. Cliccare sull'icona del dispositivo A&D da associare.
2. Tenere premuto il pulsante hardware del dispositivo A&D per 3 secondi. In questo modo il dispositivo viene reso visibile all'interno del range di rilevazione.
3. Aspettare che il dispositivo venga rilevato e che venga stabilita la connessione.
4. Accettare la richiesta di autorizzazione al pairing.
5. Aspettare che la procedura di pairing giunga a termine.

Al termine della procedura viene ricevuto un messaggio: "pairing avvenuto con successo". A questo punto l'utente può cliccare sul button di conferma, sottostante al messaggio stesso, per andare alla dashboard dell'applicazione.

La procedura di pairing (Figura 2.2) comprende una serie di fasi, eseguite in back-end dall'applicazione:

a) Scansione dei dispositivi nelle vicinanze.

b) Individuazione del dispositivo da associare.

c) Richiesta di connessione.

d) Richiesta di autorizzazione al pairing.

e) In caso di conferma: Cifratura del canale di comunicazione.

f) Lettura dei Servizi forniti dal dispositivo.

g) Settaggio di data e ora correnti sul dispositivo.

h) Termine della connessione.

i) Se il pairing è avvenuto con successo: Invio conferma di pairing avvenuto.

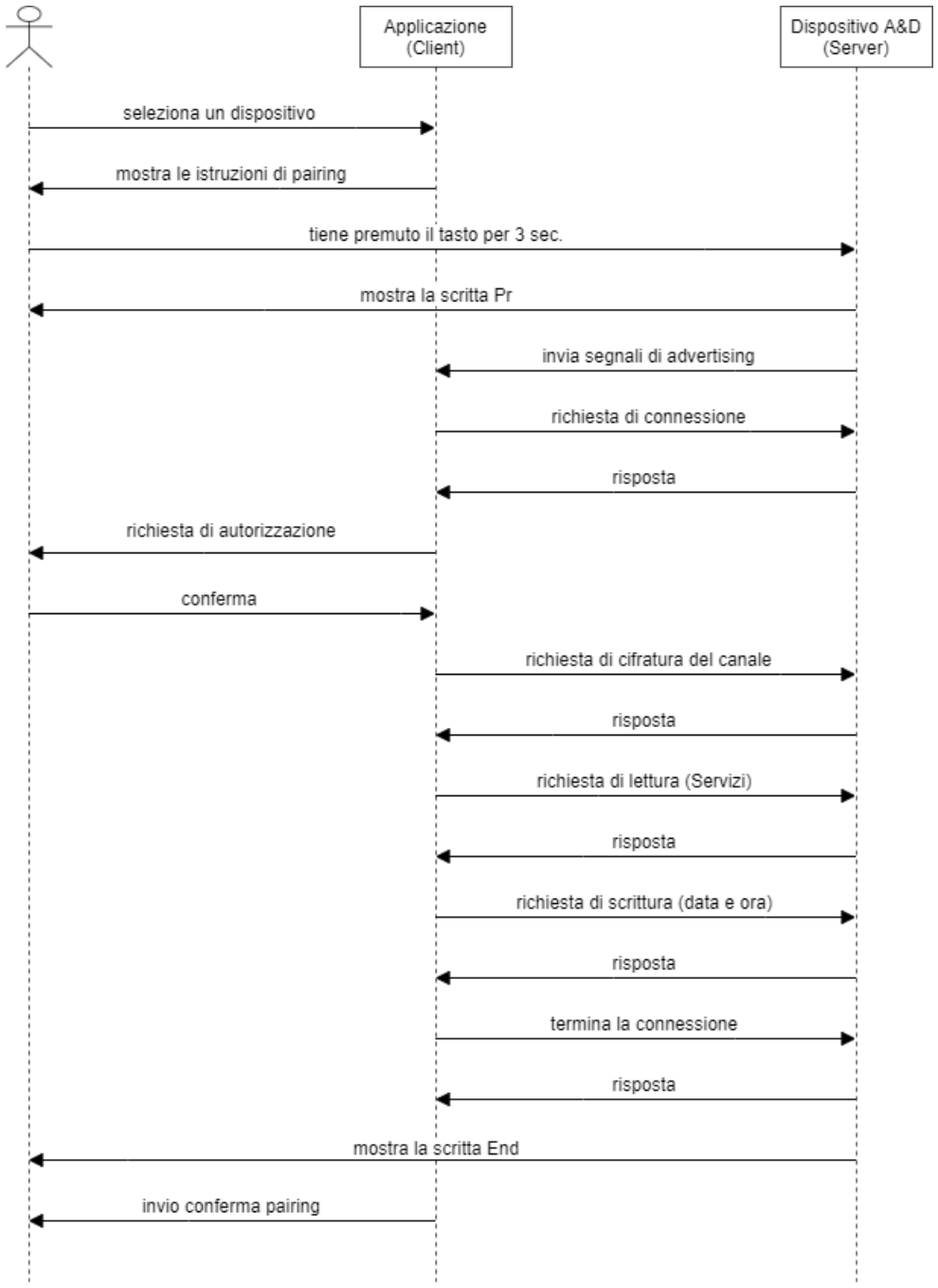

Figura 2.2: Diagramma di sequenza della procedura di pairing

Sincronizzazione dei dati

La sincronizzazione dei dati viene eseguita in maniera automatica al termine della misurazione. I dati che vengono sincronizzati sono, oltre a quelli immediatamente rilevati, tutti quelli contenuti nella memoria interna del dispositivo.

Per permettere la sincronizzazione, il paziente deve:

1. Andare sulla dashboard dell'applicazione (in questo momento inizia la fase di scansione dei dispositivi nelle vicinanze).
2. Effettuare una misurazione, tramite l'utilizzo di uno dei dispositivi A&D forniti (al termine della misurazione il dispositivo entra nella fase di advertising, tentando di essere individuato dal device Android a cui è stato accoppiato).
3. Aspettare che i dati vengano trasferiti dal dispositivo all'applicazione e che la procedura di sincronizzazione giunga a termine.

Al termine della procedura di sincronizzazione, i valori ricevuti verranno visualizzati sulla dashboard dell'applicazione. Se, per qualche problema, o per mancata connessione, i dati non vengono trasferiti all'applicazione, questi vengono salvati nella memoria interna del dispositivo. I dati vengono poi inviati alla prossima connessione riuscita.

La procedura di sincronizzazione (Figura 2.3) comprende le seguenti fasi:

a) Scansione dei dispositivi nelle vicinanze.

b) Individuazione del dispositivo interessato.

c) Richiesta di connessione.

d) Cifratura del canale di comunicazione.

e) Aggiornamento di data e ora correnti sul dispositivo.

f) Trasferimento dei dati dal dispositivo all'applicazione.

g) Memorizzazione dei dati ricevuti.

h) Termine della connessione.

i) Visualizzazione dei dati ricevuti.

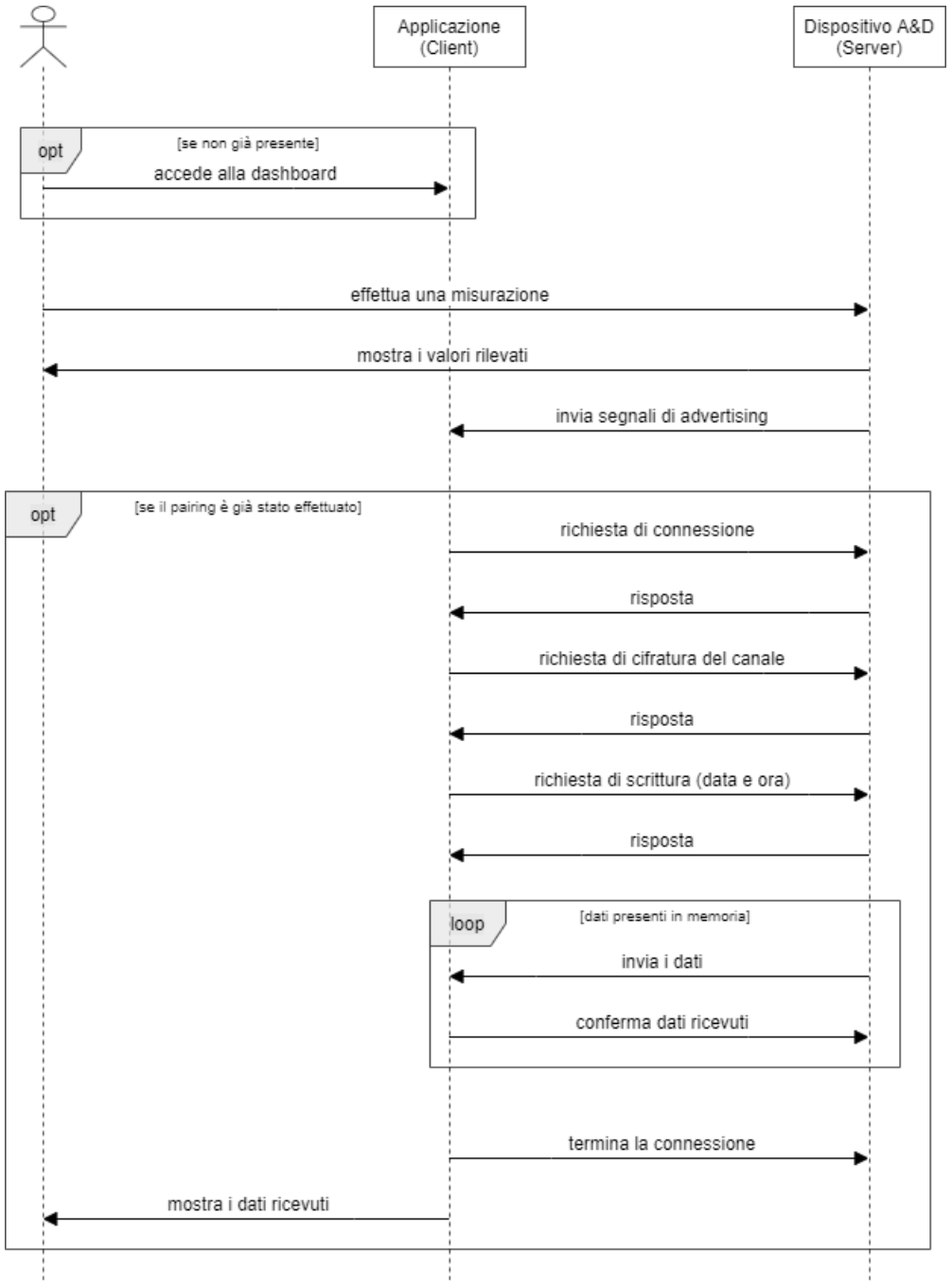

Figura 2.3: Diagramma di sequenza della procedura di sincronizzazione

Visualizzazione dei dati

La visualizzazione dei dati viene eseguita accedendo alla dashboard dell'applicazione. Se l'utente è già sulla dashboard, al termine di una sincronizzazione potrà visualizzare i dati appena ricevuti in maniera tempestiva. Vengono mostrati tutti i valori misurati che siano stati memorizzati su DB. I dati vengono mostrati in forma numerica e suddivisi in apposite View in base al dispositivo da cui provengono.

Il paziente può accedere alla dashboard in due modi:

- Cliccando sulla relativa voce nel menu di navigazione.
- Cliccando, dopo la procedura di pairing, sul button sottostante al messaggio ricevuto.

La procedura di visualizzazione (Figura 2.4) include i seguenti passaggi:

a) Ricerca dei dati memorizzati su db.

b) Estrazione dei dati.

c) Esposizione dei dati a livello di interfaccia utente.

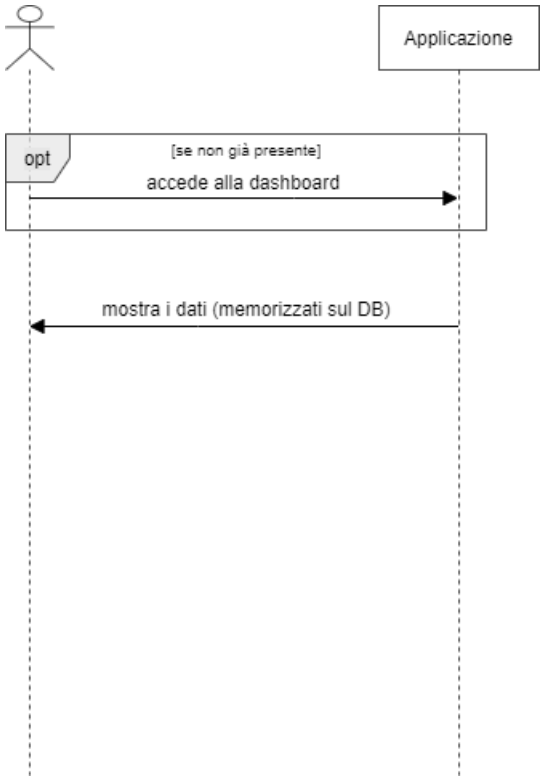

Figura 2.4: Diagramma di sequenza della procedura di visualizzazione

2.4.2 Ulteriori requisiti

- L'applicazione deve essere in grado di effettuare una scansione BLE per verificare la presenza di dispositivi in fase di advertising.
- L'applicazione deve essere in grado di riconoscere un dispositivo tra quelli rilevati durante la fase di scansione. In particolare, deve essere individuato un dispositivo BLE A&D che sia un misuratore di pressione o una bilancia.
- L'applicazione deve essere in grado di stabilire una connessione con il dispositivo. Se non già effettuato, deve permettere di effettuare il pairing con il dispositivo connesso.
- L'applicazione deve essere in grado di mantenere la connessione con il dispositivo e ricevere i dati da esso ad intervalli regolari.
- L'applicazione deve essere in grado di eseguire alcune elaborazioni sui dati ricevuti prima di mostrarli a schermo.
- L'applicazione deve gestire la memorizzazione dei dati ricevuti per poterli consultare in un secondo momento.
- L'utente deve essere in grado di visualizzare, in tempo reale, i dati ricevuti dal dispositivo. L'utente deve anche poter visualizzare i dati memorizzati in precedenza.

Vincoli

Tutti i dispositivi BLE A&D richiedono di sincronizzare data e ora del dispositivo con l'applicazione, ad ogni connessione che viene stabilita. Questo consente al dispositivo di inviare e memorizzare i dati secondo un ordine cronologico. [15, 16]

Per mantenere limitato il consumo energetico, i dispositivi A&D sono stati progettati per interrompere automaticamente la fase di advertising, se entro un certo tempo limite (circa 60 secondi) non è stata stabilita una connessione. [15, 16]

2.4.3 Sessione tipica di utilizzo

Una sessione tipica di utilizzo si svolge in questo modo:

1. L'applicazione Android viene lanciata.
2. Se il pairing del dispositivo in uso non è ancora stato eseguito:
   (a) L'utente seleziona, tramite interfaccia, il dispositivo A&D di cui vuole eseguire il pairing.
   (b) L'applicazione inizia la scansione BLE per verificare la presenza di dispositivi A&D nelle vicinanze.
   (c) L'utente, seguendo le istruzioni visualizzate sull'interfaccia, tiene premuto per alcuni secondi il pulsante hardware presente sul dispositivo (Figura 2.5).
   (d) Quando sul display del dispositivo viene visualizzata la scritta "Pr" l'utente rilascia il pulsante. Il dispositivo inizia la fase di advertising (Figura 2.6).
   (e) La scansione rileva alcuni dispositivi nel range e sceglie quello corrispondente al dispositivo selezionato dall'utente.
   (f) Viene stabilita la connessione tra Client (device Android) e Server (dispositivo A&D).
   (g) L'utente accetta la richiesta di autorizzazione al pairing.
   (h) L'applicazione continua la procedura di pairing finché il dispositivo non è stato accoppiato. Quando il dispositivo è stato accoppiato correttamente viene mostrata sul display la scritta "End" (Figura 2.7).
   (i) L'utente riceve un messaggio di conferma e clicca sul button sottostante.
   (j) L'utente viene rimandato alla dashboard dell'applicazione.
3. Se non già presente: l'utente accede alla dashboard cliccando sulla relativa voce nel menu di navigazione.

4. L'applicazione inizia la scansione BLE per verificare la presenza di dispositivi A&D nelle vicinanze.

5. L'utente fa uso del dispositivo A&D per misurare i propri valori: pressione e pulsazioni nel caso dello sfigmomanometro, peso nel caso della bilancia.

6. Al termine della misurazione il dispositivo mostra i dati rilevati e inizia la fase di advertising (Figura 2.8). La scansione rileva alcuni dispositivi nel range e sceglie il primo dispositivo A&D che viene rilevato tra quelli già accoppiati.

7. Viene stabilita la connessione tra Client (device Android) e Server (dispositivo A&D).

8. Continua la procedura di sincronizzazione. Il dispositivo trasferisce i dati all'applicazione e l'applicazione li memorizzati sul db.

9. Al termine della procedura di sincronizzazione, l'utente visualizza i dati ricevuti sulla dashboard dell'applicazione.

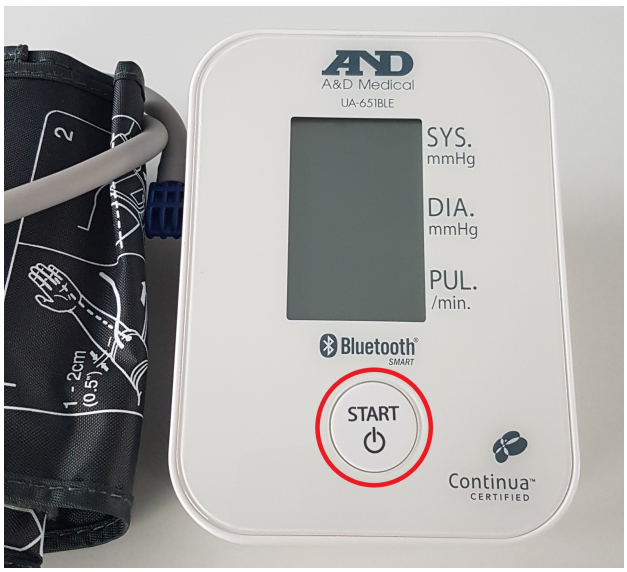

Figura 2.5: Pulsante hardware Misuratore di pressione

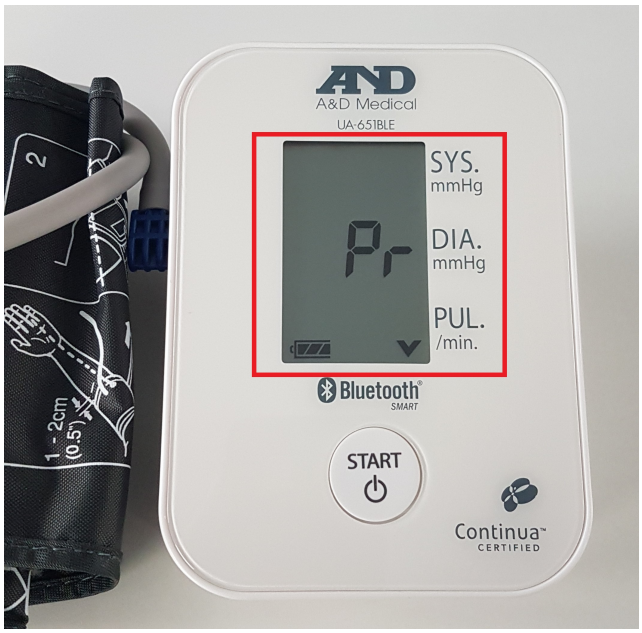

Figura 2.6: Pairing: fase di advertising

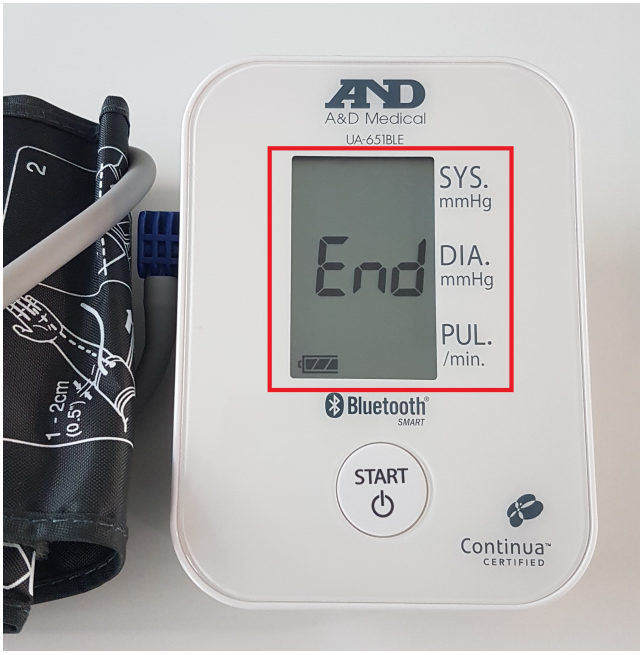

Figura 2.7: Dispositivo accoppiato correttamente

Figura 2.8: Sincronizzazione: fase di advertising

2.4.4 Strumenti utilizzati per lo sviluppo

Ambiente di sviluppo

Come ambiente di sviluppo dell'applicazione ho utilizzato L'IDE ufficiale di Android: Android Studio. Android Studio offre una serie di funzionalità e strumenti utili per la costruzione di un'app Android. L'integrazione di Gradle come sistema di build automation ne è un esempio. Gradle, infatti, permette di automatizzare tutte le attività accessorie allo sviluppo di un progetto (compilazione, packaging..), raggruppando tutti i file di origine (.java e .xml) in un unico file compresso: l'APK finale. Gradle si serve di un file "build.gradle" per reperire le dipendenze (moduli, librerie) e le impostazioni necessarie per effettuare la build dell'applicazione. In questo file viene specificato anche il range di API level compatibili con l'applicazione. In particolare l'applicazione richiede dispositivi Android che supportano un livello API compreso tra 18 (minimo) e 28 (massimo).

Dispositivi BLE A&D

L'applicazione deve interagire con due dispositivi, entrambi forniti dalla società A&D: un misuratore di pressione e una bilancia. I dispositivi sono dotati di funzione wireless Bluetooth Smart e sono in grado di connettersi con dispositivi mobile Android, iOS e Windows che siano compatibili con Bluetooth 4.0 e che abbiano installata un'applicazione per ricevere i dati.

Il misuratore di pressione, modello UA-651BLE, viene utilizzato per il controllo della pressione arteriosa (diastolica e sistolica) e delle pulsazioni. Lo strumento è dotato di memoria interna con una capacità di 30 valori. [15] Come la maggior parte degli sfigmomanometri moderni, esso possiede un display digitale che permette all'utilizzatore di avere un quadro immediato dei propri valori una volta che viene effettuata una misurazione.

La bilancia, modello UC-352BLE, viene utilizzata per il controllo del peso corporeo. Lo strumento memorizza fino a 90 valori [16] e permette la loro consultazione anche sul display digitale. Il peso può essere misurato in Chilogrammi oppure in Libbre.

Per interagire con i Servizi e le Caratteristiche dei dispositivi ho fatto uso delle API fornite dalla società stessa.

Dispositivi per il testing

Come dispositivi per il testing dell'applicazione ho utilizzato uno smartphone Samsung Galaxy S7 edge e un tablet Asus fornito dall'azienda.

2.4.5 Profili, Servizi e Caratteristiche

Nella comunicazione BLE tra i due dispositivi, il device Android ricopre il ruolo di GATT Client mentre il dispositivo A&D ricopre il ruolo di GATT Server. La comunicazione si basa su alcuni Profili Standard rilasciati dal Bluetooth SIG e su un Profilo proprietario A&D rilasciato dalla società stessa. Questi Profili consentono all'applicazione di connettersi e interagire col dispositivo A&D. Ogni Profilo mette a disposizione dei Servizi, i quali a loro volta raggruppano una serie di Caratteristiche. Le Caratteristiche, infine, incapsulano un puntatore ad uno o più dati e ne permettono la lettura/scrittura.

Misuratore di pressione

Il misuratore di pressione implementa il Blood Pressure Profile, che viene utilizzato dall'applicazione per interagire col dispositivo e ottenere le misurazioni (pressione max., pressione min., pulsazioni) e altri dati. Il Blood Pressure Profile (o BLP) fornisce due tipi di Servizi: [11]

- Blood Pressure Service.
- Device Information Service.

Il Blood Pressure Service (o BLS) espone i dati misurati e le funzionalità del dispositivo. Il BLS viene identificato dall'UUID 0x1810.

Il Device Information Service (o DIS) espone le informazioni sul produttore e/o sul fornitore del dispositivo. Il DIS viene identificato dall'UUID 0x180A.

L'applicazione deve interrogare Il BLS per acquisire i dati misurati dal paziente, mentre deve parlare con il DIS se necessita informazioni sul dispositivo.

Ogni Servizio raggruppa al suo interno alcune Caratteristiche, che contengono i dati e le informazioni effettive. Nello specifico, il BLS contiene due caratteristiche distinte: [13]

- Blood Pressure Measurement.
- Blood Pressure Feature.

La Caratteristica Blood Pressure Measurement contiene le misurazioni effettuate dal monitor e viene identificata dall'UUID 0x2A35. Include tutti o alcuni dei seguenti campi: [15]

- Flags: campo obbligatorio contenente l'unità di misura (mmHg o kPa) e usato per indicare la presenza di campi opzionali (Figura 2.9).
- Unit: campo obbligatorio contenente i valori numerici della pressione max. e min.
- Time Stamp: campo opzionale contenente data e ora della misurazione.
- Pulse Rate: campo opzionale contenente il valore delle pulsazioni.
- User ID: campo opzionale contenente l'ID dell'utente.
- Measurement Status: campo opzionale contenente lo stato della misurazione (es. rilevato movimento del corpo durante la misurazione, rilevate pulsazioni irregolari, ecc.).

Bit	Name	Key	Value
0	Blood Pressure Unit Flag	0	mmHg (C1)
		1	kPa (C2)
1	Time Stamp Flag	0	Time Stamp not present
		1	Time Stamp present(C3)
2	Pulse Rate Flag	0	Pulse Rate not present
		1	Pulse Rate present (C4) (fixed)
3	User ID Flag	0	User ID not present (fixed)
		1	User ID present (C5)
4	Measurement Status Flag	0	Measurement Status not present
		1	Measurement Status present(C6)(fixed)

Figura 2.9: Blood Pressure Measurement: Unità di misura e campi opzionali

Il bit "0", relativo all'unità di misura, viene impostato a 0 (key) se l'unità che viene utilizzata per rappresentare il dato è mmHg, oppure, viene impostato a 1 se l'unità utilizzata è kPa. Gli altri bit ("1","2","3","4") vengono impostati a 0 se il campo opzionale non è presente, oppure a 1 se il campo opzionale è presente.

La Caratteristica Blood Pressure Feature descrive le funzionalità supportate dal dispositivo e viene identificata dall'UUID 0x2A49. Esempi di funzionalità sono:

- Rilevamento del movimento del corpo: mostra se è stato rilevato un movimento durante la misurazione.
- Rilevamento di pulsazioni irregolari: mostra se sono state rilevate pulsazioni irregolari durante la misurazione.
- Rilevamento della posizione di misurazione: mostra se è stata rilevata una posizione di misurazione scorretta.

Anche in questo caso è incluso nella Caratteristica un campo Flags che indica quali funzionalità sono supportate dal dispositivo e quali no.

Per sapere quali dati sono presenti nelle Caratteristiche del BLS, l'applicazione deve determinare il contenuto di queste Caratteristiche basandosi sui bit del campo Flags. Ciò consente di determinare quali campi opzionali sono inclusi nella Caratteristica e quali no, quindi di stabilire i valori e le funzionalità che vengono forniti dal dispositivo in questione.

Il DIS contiene invece le seguenti caratteristiche: [15]

- Manufacturer Name String (UUID 0x2A29).
- Model Number String (UUID 0x2A24).
- Serial Number String (UUID 0x2A25).
- Hardware Revision String (UUID 0x2A27).
- Firmware Revision String (UUID 0x2A26).
- Software Revision String (UUID 0x2A28).
- System ID (UUID 0x2A23).
- Registration Certification Data (UUID 0x2A2A).

La Caratteristica Manufacturer Name String rappresenta il nome del produttore del dispositivo. In questo caso A&D Medical.

La Caratteristica Model Number String rappresenta il numero del modello del dispositivo, assegnato dal fornitore. In questo caso UA-651BLE.

La Caratteristica Serial Number String rappresenta il numero seriale di una singola istanza del prodotto.

La Caratteristica Hardware Revision String rappresenta la revisione dell'hardware all'interno del dispositivo.

La Caratteristica Firmware Revision String rappresenta la revisione del firmware all'interno del dispositivo.

La Caratteristica Software Revision String rappresenta la revisione del software all'interno del dispositivo.

La Caratteristica System ID rappresenta un identificatore univoco di una singola istanza del prodotto.

La Caratteristica Registration Certification Data rappresenta le informazioni normative e di certificazione del prodotto.

Un'altra Caratteristica fornita dal dispositivo, distinta da quelle contenute nel BLS o nel DIS, è la Caratteristica Date Time. Questa Caratteristica viene utilizzata per impostare la data e l'ora sul dispositivo A&D e per leggere la data e l'ora impostate. Include i campi per anno, mese, giorno, ore, minuti e secondi. I giorni sono rappresentati utilizzando il calendario Gregoriano, mentre le ore sono rappresentate nel sistema 24h.

Un Servizio comune, che viene fornito dai dispositivi alimentati a batteria, è il Battery Service (UUID 0x180F). Questo Servizio espone lo stato della batteria contenuta in un dispositivo. Contiene la Caratteristica Battery Level (UUID 0x2A19), la quale restituisce il livello corrente della batteria in percentuale (dallo 0% al 100%: 0% rappresenta una batteria completamente scarica, 100% rappresenta una batteria completamente carica). [2, 15]

Oltre al Blood Pressure Profile, il dispositivo implementa un Profilo personalizzato A&D, realizzato dalla società produttrice A&D Medical. Questo Profilo fornisce un Servizio proprietario Custom Service, il quale definisce i comandi per gestire le impostazioni e lo stato del dispositivo. [15]

Bilancia

La bilancia implementa il Weight Scale Profile, che viene utilizzato dall'applicazione per interagire col dispositivo e ottenere le misurazioni (peso) e altri dati. Il Weight Scale Profile (o WSP) fornisce due tipi di Servizi: [12]

- Weight Scale Service.
- Device Information Service.

Il Weight Scale Service (o WSS) espone i dati misurati e le funzionalità del dispositivo. Il WSS viene identificato dall'UUID 0x181D.

Il Device Information Service (o DIS) espone le informazioni sul produttore e/o sul fornitore del dispositivo. Il DIS viene identificato dall'UUID 0x180A.

L'applicazione deve interrogare Il WSS per acquisire i dati misurati dal paziente, mentre deve parlare con il DIS se necessita informazioni sul dispositivo.

Ogni Servizio raggruppa al suo interno alcune Caratteristiche, che contengono i dati e le informazioni effettive. Nello specifico, il WSS contiene due caratteristiche distinte: [14]

- Weight Scale Measurement.
- Weight Scale Feature.

La Caratteristica Weight Scale Measurement contiene le misurazioni effettuate dalla bilancia e viene identificata dall'UUID 0x2A9D. Include tutti o alcuni dei seguenti campi: [16]

- Flags: campo obbligatorio contenente l'unità di misura (Kg o lb) e usato per indicare la presenza di campi opzionali (Figura 2.10).

- Unit: campo obbligatorio contenente il valore numerico del peso.
- Time Stamp: campo opzionale contenente data e ora della misurazione.
- User ID: campo opzionale contenente l'ID dell'utente.
- BMI: campo opzionale contenente il valore dell'indice di massa corporea.
- Height: campo opzionale contenente il valore dell'altezza.

Bit	Name	Key	Value
0	Measurement Units	0	SI(Kg)
		1	Imperial(lb)
1	Time Stamp present	0	Time Stamp not present
		1	Time Stamp present
2	User ID present	0	False(fixed)
		1	True
3	BMI and Height present	0	False(fixed)
		1	True

Figura 2.10: Weight Scale Measurement: Unità di misura e campi opzionali

La Caratteristica Weight Scale Feature descrive le funzionalità supportate dal dispositivo e viene identificata dall'UUID 0x2A9E. Esempi di funzionalità sono:

- Rilevamento dell'indice di massa corporea (BMI): mostra il BMI che viene calcolato dalla misurazione del peso e dall'altezza dell'utente.
- Gestione multi-utente: permette di gestire separatamente le misurazioni effettuate da più utenti che utilizzano il dispositivo.

Anche in questo caso è incluso nella Caratteristica un campo Flags che indica quali funzionalità sono supportate dal dispositivo e quali no.

Per sapere quali dati sono presenti nelle Caratteristiche del WSS, l'applicazione deve determinare il contenuto di queste Caratteristiche basandosi sui bit del campo Flags. Ciò consente di determinare quali campi opzionali sono inclusi nella

Caratteristica e quali no, quindi di stabilire i valori e le funzionalità che vengono forniti dal dispositivo in questione.

Per quanto riguarda il DIS, il Servizio contiene le stesse Caratteristiche presentate nella sezione precedente (Misuratore di pressione).

Come per lo sfigmomanometro, anche la bilancia fornisce la Caratteristica Date Time, il Servizio Battery Service e implementa il Profilo personalizzato A&D. [16]

Capitolo 3

Implementazione

In questo capitolo verranno descritte le parti rilevanti di implementazione della componente di cui mi sono occupato. Verranno anche riportate le scelte implementative fatte per soddisfare i requisiti descritti nel capitolo precedente (Analisi e Progettazione).

3.1 Panoramica dell'applicazione

Nella schermata Device Setup (Figura 3.1) è possibile selezionare un dispositivo di cui si vuole effettuare il pairing. Attualmente vengono gestiti solo due dei dispositivi A&D proposti: misuratore di pressione (blood pressure) e bilancia (weight scale). Cliccando su ognuno di essi, viene aperta la schermata di istruzioni relativa al dispositivo selezionato.

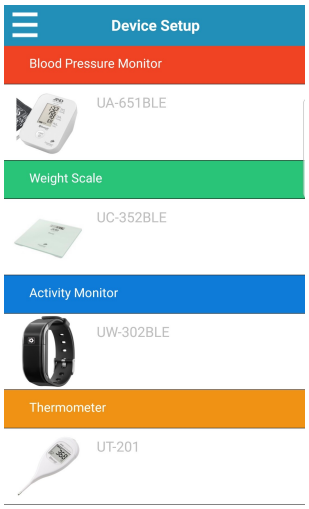

Figura 3.1: Schermata Device Setup

La schermata di istruzioni (Figura 3.2) è il punto di partenza per il pairing. Guida l'utente su come interagire con il dispositivo (fisico) selezionato per renderlo rilevabile dall'applicazione e consentirgli così di effettuare il pairing del dispositivo stesso.

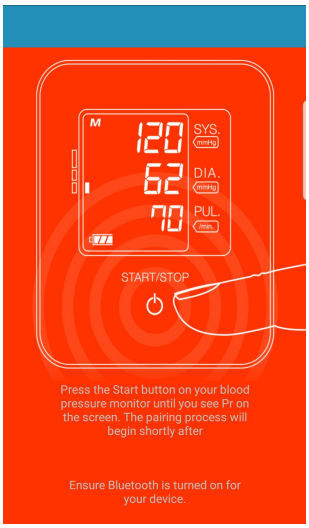

Figura 3.2: Schermata istruzioni Misuratore di pressione

Nella schermata Dashboard (Figura 3.3) è possibile visualizzare i dati delle misurazioni effettuate. Vengono mostrate per prime le misurazioni più recenti. Tramite le due frecce centrali, posizionate sotto la barra dell'app, è possibile scorrere lo storico dei valori misurati. In particolare, cliccando la freccia a sinistra, vengono visualizzati i valori della data precedente a quella attualmente visualizzata, cliccando la freccia a destra, i valori della data successiva. L'accesso e la permanenza nella Dashboard permette all'utente di sincronizzare i dati con quelli rilevati da un dispositivo A&D, visualizzando immediatamente i dati aggiornati.

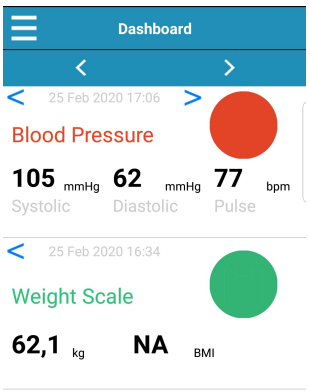

Figura 3.3: Schermata Dashboard

Il menu laterale a scomparsa (Navigation Drawer) permette di navigare nell'applicazione, passando quindi da una schermata all'altra (Figura 3.4). Appare quando l'utente clicca sull'icona "drawer" ≡ nella barra dell'app in alto. Questa barra è presente in ogni schermata, ad eccezione della schermata di istruzioni, nella quale l'utente, per tornare indietro, deve cliccare il tasto back del device Android. Il Navigation Drawer presenta un elenco di voci, sulle quali, cliccando, si viene indirizzati ad una schermata specifica. Attualmente le voci sono 3:

- Device Set-up: porta l'utente alla schermata Device Setup.
- Dashboard: porta l'utente alla schermata Dashboard.
- Sign Out: porta l'utente ad uscire dall'applicazione.

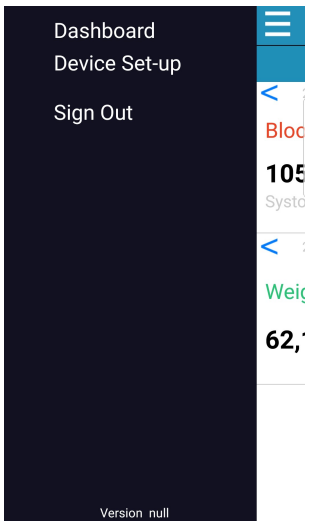

Figura 3.4: Menu laterale di navigazione

3.2 Gestione dei dati

Per salvare i dati localmente sul device Android vengono utilizzate due librerie (fornite da Android):

- SharedPreferences.
- SQLite.

Le SharedPreferences rappresentano un metodo semplice e veloce per memorizzare alcune informazioni in un file privato. Le informazioni che vengono memorizzate sono: unità di misura impostata, nome e indirizzo del dispositivo rilevato, modalità attuale dell'applicazione (pairing, sincronizzazione). Il file contiene coppie chiave-valore ed è accessibile e modificabile attraverso specifici metodi.

SQLite fornisce le funzionalità di un database relazionale in un ambiente con risorse limitate (quale il dispositivo mobile). Vengono memorizzate tutte le informazioni estrapolate dal dispositivo durante la sincronizzazione: valori misurati, data e ora, unità di misura, altri dati in base al dispositivo utilizzato. La classe che rappresenta il DB fornisce i metodi per le operazioni di: inserimento, modifica, cancellazione e lettura dei dati.

3.3 Pairing

Nella schermata di set-up dei dispositivi, nella quale l'utente seleziona un dispositivo di cui vuole effettuare il pairing, le varie scelte vengono predisposte in una lista di View. Una volta che l'utente seleziona una View con l'icona del dispositivo di cui vuole eseguire il pairing, si aprirà una schermata contenente le istruzioni per avviare il dispositivo in modalità advertising. Contemporaneamente, l'applicazione inizia la scansione dei dispositivi BLE nelle vicinanze. Per gestire l'indirizzamento alle diverse schermate di istruzioni, associo un ascoltatore alla lista di View dei dispositivi (list_devicesetup) tramite il metodo setOnItemClickListener(). Questo metodo registra una callback (onItemClick) che viene invocata quando un elemento della lista viene selezionato. Alla generazione dell'evento, viene individuata la posizione dell'elemento nella lista e, in base ad essa, vengono salvati alcuni parametri di configurazione specifici del dispositivo (elemento selezionato). Viene avviata l'Activity contenente le istruzioni di pairing relative al dispositivo stesso.

```
list_devicesetup.setOnItemClickListener(new OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
                            int position, long id) {
        if (position == 1) { //blood pressure
            ADSharedPreferences.putString
                (ADSharedPreferences.KEY_DEVICE_SET_UP_MODE,
                ADSharedPreferences.VALUE_DEVICE_SET_UP_MODE_BP);
            Intent intent = new Intent
                (DeviceSetUpActivityListDesign.this,
                InstructionActivity.class);
            intent.putExtra
                (ADInstructionActivity.DEVICE_SCAN_MODE_KEY,
                ADInstructionActivity.DEVICE_SCAN_MODE_BP);
            startActivity(intent);
        } else if (position == 3) {
            ADSharedPreferences.putString
                (ADSharedPreferences.KEY_DEVICE_SET_UP_MODE,
                ADSharedPreferences.VALUE_DEVICE_SET_UP_MODE_WS);
            Intent intent = new Intent
                (DeviceSetUpActivityListDesign.this,
                InstructionActivity.class);
            intent.putExtra
                (ADInstructionActivity.DEVICE_SCAN_MODE_KEY,
                ADInstructionActivity.DEVICE_SCAN_MODE_WS);
            startActivity(intent);
        }
    }
});
```

Listing 3.1: Indirizzamento alla schermata di istruzioni

Alla creazione dell'Activity viene recuperato, nei contenuti extra dell'Intent, il valore associato alla chiave DEVICE_SCAN_MODE_KEY. Questo valore è semplicemente un numero (0/1/2) che indica quale dispositivo vogliamo accoppiare.

Confronto il valore recuperato con le costanti DEVICE_SCAN_MODE_BP e DEVICE_SCAN_MODE_WS, alle quali è stato assegnato il valore del dispositivo di riferimento (Blood Pressure o Weight Scale).

Una volta individuato il dispositivo, viene impostato il layout dell'Activity, cambiando l'immagine, il colore di sfondo e il messaggio di istruzione. In questo modo viene mantenuto un distacco tra la struttura del layout e i contenuti mostrati. Infatti, il layout che viene utilizzato è lo stesso per entrambi i dispositivi, mentre cambiano i contenuti, che sono quelli relativi al dispositivo stesso. Vengono rispettati, quindi, i requisiti di universalità e scalabilità imposti in precedenza.

```
1 public static final String DEVICE_SCAN_MODE_KEY =
2                             "DeviceScanModeKey";
3 public static final int DEVICE_SCAN_MODE_NONE = 0;
4 public static final int DEVICE_SCAN_MODE_BP = 1;
5 public static final int DEVICE_SCAN_MODE_WS = 2;
```

Listing 3.2: Riferimenti ai dispositivi da accoppiare

```
1 mDeviceScanMode = intent.getIntExtra(DEVICE_SCAN_MODE_KEY,
2                                      DEVICE_SCAN_MODE_NONE);
```

Listing 3.3: Recupero del valore relativo al dispositivo da accoppiare

```
if (mDeviceScanMode == DEVICE_SCAN_MODE_WS) {
  setDialogImageWithResourceID(R.drawable.ws_pairing);
  setDialogBackGroundColorWithResorceID
      (R.color.dashboard_theme_color_weightscale);
  setDialogMessageWithResorceID
      (R.string.paring_message_weight_scale);

} else if (mDeviceScanMode == DEVICE_SCAN_MODE_BP) {
  setDialogImageWithResourceID(R.drawable.bp_pairing);
  setDialogBackGroundColorWithResorceID
      (R.color.dashboard_theme_color_bloodpresure);
  setDialogMessageWithResorceID
      (R.string.paring_message_bloodpressure_monitor);
}
```

Listing 3.4: Impostazione layout della schermata istruzioni

Sempre alla creazione dell'Activity, verifico che il BLE sia supportato dal device Android su cui è eseguita l'applicazione.

```
if (!BleConnectService.isEnableBluetoothFunction(this)) {
  finish();
  return;
}
```

```
public static boolean isEnableBluetoothFunction(Context context) {
    if (Build.VERSION.SDK_INT >= 18) {
        // for API 18
        final BluetoothManager bluetoothManager =
            (BluetoothManager) context.getSystemService
                (Context.BLUETOOTH_SERVICE);
        return (bluetoothManager.getAdapter() != null);
    } else {
        return (BluetoothAdapter.getDefaultAdapter() != null);
    }
}
```

Listing 3.5: Verifica supporto della tecnologia BLE

Viene registrato un Broadcast Receiver mBleConnectionReceiver, il cui compito è quello di porsi in ascolto e gestire i messaggi (eventi) scaturiti durante il processo di pairing del dispositivo. Il BR viene invocato quando il metodo sendBroadcast() invia un messaggio (Intent) contenente l'azione per la quale il BR stesso è stato registrato. Le azioni che il BR deve gestire vengono specificate in un oggetto IntentFilter. Ogni volta che sendBroadcast() lancia un Intent, il sistema richiama la callback onReceive del BR predisposto ad occuparsene. In base all'azione contenuta nell'Intent, la callback eseguirà le istruzioni specifiche per gestire l'azione stessa.

```
registerReceiver(mBleConnectionReceiver,
                 BleConnectService.BleConnectionFilter);
```

```
public static final IntentFilter BleConnectionFilter =
    new IntentFilter() {{
    addAction(ACTION_DEVICE_SCAN);
    addAction(ACTION_DEVICE_SETUP);
    addAction(ACTION_DEVICE_CONNECT);
    addAction(ACTION_DEVICE_DISCONNECT);
    addAction(ACTION_DEVICE_REQUEST_PAIRING);
    addAction(ACTION_READ_CHARACTER);
    addAction(ACTION_WRITE_CHARACTER);
    addAction(ACTION_DISCOVERED_SERVICES);
}};
```

Listing 3.6: Registrazione Broadcast Receiver e relative azioni

Per eseguire le operazioni di connessione BLE col dispositivo A&D viene utilizzato un Bound Service. Esso offre un'interfaccia IBinder che permette ai componenti associati di comunicare con il Service stesso (inviare richieste, ricevere risultati, ecc.). Il Service viene creato dichiarandolo nel Manifest e implementando la relativa classe ("BleConnectService"). Per associare un componente (indicato come Client) al Bound Service utilizzo il metodo bindService(), fornito da Android. La chiamata al metodo bindService() porta il sistema ad invocare la callback onBind, la quale restituisce l'interfaccia IBinder.

Quando il sistema Android crea la connessione tra Client e Service, chiama il metodo onServiceConnected(). Quando un componente ha terminato l'interazione con il Service, chiama il metodo unbindService() per separarsi. La chiamata al metodo unbindService() porta il sistema ad invocare la callback onUnBind. Una volta rimossa la connessione tra Client e Service, il sistema Android chiama il metodo onServiceDisconnected().

Una volta che la connessione tra Client e Service è stata stabilita, quindi al richiamo della callback onServiceConnected, viene fornita un'istanza della classe BleConnectService con cui il Client può interagire e viene invocato un metodo dell'istanza che fa partire la scansione dei dispositivi.

```
<service android:name=
    "jp.co.aandd.bleSimpleApp.gatt.BleConnectService" />
```

Listing 3.7: Creazione Service

```
bindService(new Intent(ADInstructionActivity.this,
         BleConnectService.class), mConnection,
         Context.BIND_AUTO_CREATE);
mIsServiceBind = true; //indica che il componente e' stato
                       //associato al Service
```

Listing 3.8: Associazione al Bound Service

```
unbindService(mConnection);
mIsServiceBind = false; //indica che il componente e' stato
                        //dissociato dal Service
```

Listing 3.9: Dissociazione dal Bound Service

```
public class BleConnectService extends Service {

    public class BleConnectionBinder extends Binder {
        public BleConnectService getService() {
            return BleConnectService.this;
        }
    }

    private final IBinder mBinder = new BleConnectionBinder();
```

Listing 3.10: Creazione interfaccia IBinder

```
@Override
public IBinder onBind(Intent intent) {
    return mBinder;
}

@Override
public boolean onUnbind(Intent intent) {
    return super.onUnbind(intent);
}
```

Listing 3.11: Callback onBind e onUnbind

```
protected BleConnectService mBleService;
private boolean mIsServiceBind = false;

private ServiceConnection mConnection =
    new ServiceConnection() {
    @Override
    public void onServiceConnected
        (ComponentName className, IBinder service) {
        mBleService = ((BleConnectService.BleConnectionBinder)
                        service).getService();
        if(mBleService != null) {
            mBleService.startScanDevices();
        }
    }
```

```
15
16      @Override
17      public void onServiceDisconnected(ComponentName className) {
18        mBleService = null;
19      }
20  };
```

Listing 3.12: Metodi chiamati alla creazione/rimozione della connessione

Il metodo startScanDevices() è il punto di partenza del processo di pairing, in cui viene avviata una scansione Low Energy per individuare il dispositivo A&D nelle vicinanze che sta facendo advertising. Per avviare la scansione viene utilizzata un'istanza BluetoothAdapter. La classe BluetoothAdapter fornisce le API per interagire con il modulo Bluetooth del device Android, tramite il quale è possibile eseguire una serie di funzionalità Bluetooth (tra le quali, appunto, la scansione dei dispositivi). L'istanza viene restituita dal metodo getBluetoothAdapter(), che utilizza il System Service BluetoothManager per ottenere l'Adapter Bluetooth del device. Sull'istanza ottenuta viene chiamato il metodo startLeScan(), il quale avvia effettivamente la scansione. Il metodo prende come parametro una callback interface mLeScanCallback, utilizzata per fornire i risultati della scansione LE.

```
1  public void startScanDevices() {
2      if (isEnableBluetoothFunction(this)) {
3          BluetoothAdapter adapter = getBluetoothAdapter();
4          if (adapter.isEnabled()) {
5              adapter.startLeScan(mLeScanCallback);
6              mIsReserveScan = false;
7          } else {
8              mIsReserveScan = true;
9              adapter.enable();
10         }
11     }
12 }
```

Listing 3.13: Avvio scansione LE dei dispositivi

L'interfaccia contiene una callback onLeScan, la quale viene invocata al rilevamento di un dispositivo BLE durante la scansione. Il dispositivo rilevato viene rappresentato da una classe BluetoothDevice. Questa permette di creare una connessione col dispositivo, o di interrogarlo, per ottenere alcune informazioni su di esso. Le informazioni necessarie, in questo caso, sono:

- Il nome del dispositivo, utilizzato dal metodo ableToScanDevice() per controllare se il dispositivo rilevato è un dispositivo A&D.
- L'indirizzo MAC del dispositivo, inserito in un oggetto Bundle.

Il Bundle viene poi passato al metodo sendMessage().

```
1  private BluetoothAdapter.LeScanCallback mLeScanCallback =
2      new BluetoothAdapter.LeScanCallback() {
3      @Override
4      public void onLeScan(final BluetoothDevice device, int rssi,
5                           byte[] scanRecord) {
6          if (ableToScanDevice(device)) {
7              Bundle bundle = new Bundle();
8              bundle.putString(KEY_DEVICE_ADDRES,
9                               device.getAddress());
10             sendMessage(ACTION_DEVICE_SCAN, bundle);
11         }
12     }
13 };
```

Listing 3.14: Interfaccia per fornire i risultati della scansione

SendMessage() si occupa di creare un Intent contenente l'informazione passata come parametro (in questo caso l'oggetto bundle) e l'azione da eseguire (ACTION_DEVICE_SCAN). Questo intent verrà inviato come messaggio, tramite il metodo sendBroadcat(), al Broadcast Receiver incaricato alla gestione dell'azione (specificata nell'Intent).

```
protected void sendMessage(String action, Bundle bundle) {
    Intent intent = new Intent();
    intent.setAction(action);
    if (bundle != null) {
        intent.putExtras(bundle);
    }
    sendBroadcast(intent);
}
```

Listing 3.15: Invio dell'Intent contenente l'azione da gestire

Invocata la callback onReceive del BR, viene individuato il blocco di istruzioni relativo all'azione ACTION_DEVICE_SCAN. In particolare, tramite il metodo onFindConnectDevice(), viene analizzato il nome del dispositivo (con indirizzo specificato) che è stato rilevato durante la scansione. Dal nome è possibile capire se si tratta di un dispositivo UA-651BLE (Blood Pressure) o un dispositivo UC-352BLE (Weight Scale). Una volta individuato il tipo di dispositivo, viene stoppata la scansione in corso.

Il metodo stopScanDevice() recupera l'istanza BluetoothAdapter, sulla quale vengono richiamate le funzioni stopLeScan() e cancelDiscovery(). StopLeScan() interrompe la scansione in corso, individuata dall'interfaccia mLeScanCallback. CancelDiscovery() annulla il processo di rilevamento del dispositivo.

```
public void stopScanDevice() {
    if (isEnableBluetoothFunction(this)) {
        BluetoothAdapter adapter = getBluetoothAdapter();
        adapter.stopLeScan(mLeScanCallback);
        adapter.cancelDiscovery();
    }
}
```

Listing 3.16: Stop della scansione

A questo punto viene avviato il processo di connessione al dispositivo rilevato, tramite il metodo connectDevice(). Creo un'istanza di SetupDevice, che rappresenta

il dispositivo (con indirizzo specificato) a cui connettersi. Su questa richiamo il metodo connect().

```
public void connectDevice(String address) {
    if (getBluetoothAdapter() == null || mSetupDevice != null) {
        return;
    }
    BluetoothDevice device =
        getBluetoothAdapter().getRemoteDevice(address);
    if (device != null) {
        mSetupDevice = new SetupDevice(device);
        mSetupDevice.connect(this);
    }
}
```

Listing 3.17: Avvio processo di connessione

Il metodo connect fa uso di una variabile gatt, che indica un riferimento ad un oggetto di tipo BluetoothGatt, e di una variabile device, che rappresenta il dispositivo remoto a cui connettersi (BluetoothDevice). La classe BluetoothGatt fornisce le API per interagire con i Profili del GATT. Dopo un breve ritardo (mezzo secondo) viene chiamata la funzione connectGatt(), attraverso la quale viene richiesto di connettersi al GATT Server del dispositivo (su cui è stata richiamata la funzione). La funzione ritorna un'istanza BluetoothGatt, che viene utilizzata dal GATT Client per condurre operazioni sul Server. Il secondo parametro della funzione indica di connettersi automaticamente al dispositivo remoto non appena diventa disponibile. MBleCallback rappresenta la classe astratta BluetoothGattCallback, utilizzata per fornire i risultati della connessione al Client, implementando delle callback. Queste vengono richiamate ad ogni cambio di stato della connessione.

```
public void connect(final Context context) {
    if (gatt != null) {
        disconnect();
    }
    new Handler().postDelayed(new Runnable() {
```

```
6           @Override
7           public void run() {
8               gatt = device.connectGatt(context, true,
9                                                 mBleCallback);
10          }
11      }, 500);
12  }
```

Listing 3.18: Connessione al GATT Server

Le callback implementate sono le seguenti:

- onConnectionStateChange.
- onServicesDiscovered.
- onCharacteristicRead.
- onCharacteristicWrite.

La callback onConnectionStateChange viene invocata quando il GATT Client si è connesso/disconnesso al/dal GATT Server. NewState indica lo stato del dispositivo. Se newState = 0 (valore di BluetoothProfile.STATE_DISCONNECTED), chiamo sendMessage passando l'azione ACTION_DEVICE_DISCONNECT. Se newState = 1 (valore di BluetoothProfile.STATE_CONNECTED), controllo se il dispositivo è già stato accoppiato: se il dispositivo non è stato accoppiato, chiamo sendMessage passando l'azione ACTION_DEVICE_REQUEST_PAIRING; se il dispositivo è già stato accoppiato, chiamo la funzione gatt.discoverServices().

Il Broadcast Receiver gestisce l'azione ACTION_DEVICE_REQUEST_PAIRING, inviando la richiesta di pairing e aggiornando lo stato di accoppiamento del dispositivo. Una volta che il dispositivo risulta accoppiato, viene chiamata la funzione gatt.discoverServices(). La funzione discoverServices() permette di rilevare i Servizi offerti dal dispositivo remoto e ritorna true se il rilevamento è stato avviato. Una volta completato il rilevamento viene richiamata la callback onServicesDiscovered.

```
@Override
protected void onConnectionStateChange(BluetoothGatt gatt,
                                       int status, int newState) {
    super.onConnectionStateChange(gatt, status, newState);
    if (mSetupDevice == null) {
        return;
    }

    if (newState == BluetoothProfile.STATE_DISCONNECTED) {
        sendMessage(ACTION_DEVICE_DISCONNECT, null);
    } else if (newState == BluetoothProfile.STATE_CONNECTED) {
        BluetoothDevice device = gatt.getDevice();
        int bondstate = device.getBondState();
        if (bondstate != BluetoothDevice.BOND_BONDED) {
            sendMessage(ACTION_DEVICE_REQUEST_PAIRING, null);
        } else {
            gatt.discoverServices();
        }
    }
}
```

Listing 3.19: Stato Connessione: GATT Client connesso/disconnesso

La callback onServicesDiscovered viene invocata quando l'elenco dei Servizi individuati nel dispositivo remoto è stato aggiornato, ovvero quando sono stati scoperti nuovi Servizi. In questo caso viene creato un Bundle da passare, insieme all'azione ACTION_DISCOVERED_SERVICES, a sendMessage(). L'azione viene gestita da mBleConnectionReceiver che, chiamando il metodo gatt.getService() e passando come parametro l'UUID del Servizio da recuperare, assegna ad una variabile gattService di tipo BluetoothGattService il Servizio recuperato.

Nello specifico, il Servizio che viene recuperato è il Custom Service di A&D. Attraverso il Servizio si risale poi alla Caratteristica contenuta al suo interno, la quale viene assegnata ad una variabile characteristic di tipo BluetoothGattCharacteristic. Utilizzando la funzione setValue() viene aggiornato il valore memorizzato nella Caratteristica con il valore passato come parametro (array di byte). I byte

inseriti nell'array indicano alcune impostazioni di memoria (in formato esadecimale) che verranno controllate successivamente prima di scrivere la data e l'ora sul dispositivo.

```
@Override
public void onServicesDiscovered(final BluetoothGatt gatt,
                                 final int status) {
    super.onServicesDiscovered(gatt, status);
    delayHandler.postDelayed(new Runnable() {
        @Override
        public void run() {
            Bundle bundle = new Bundle();
            BluetoothDevice device = gatt.getDevice();
            String device_name = device.getName();
            bundle.putString(KEY_DEVICE_NAME, device_name);
            sendMessage(ACTION_DISCOVERED_SERVICES, bundle);
        }
    }, 1000L);
}
```

Listing 3.20: Stato Connessione: aggiornamento Servizi

Il metodo gatt.writeCharacteristic() scrive sul dispositivo la Caratteristica aggiornata, passata come parametro. Una volta completata l'operazione di scrittura viene richiamata la callback onCharacteristicWrite. La callback onCharacteristicWrite crea un Bundle contenente:

- Il risultato dell'operazione di scrittura.
- L'UUID della Caratteristica scritta sul dispositivo remoto.
- Il nome del dispositivo remoto accoppiato.

Queste informazioni vengono inviate tramite sendMessage() al BR mBleConnectionReceiver, che si occuperà di gestire l'azione ACTION_WRITE_CHARACTER.

```
@Override
public void onCharacteristicWrite(BluetoothGatt gatt,
    BluetoothGattCharacteristic characteristic, int status) {
    super.onCharacteristicWrite(gatt, characteristic, status);
    BluetoothDevice device = gatt.getDevice();
    Bundle bundle = new Bundle();
    bundle.putBoolean(KEY_RESULT,
                      (status == BluetoothGatt.GATT_SUCCESS));
    bundle.putString(KEY_UUID_STRING,
                      characteristic.getUuid().toString());
    bundle.putString(KEY_DEVICE_NAME, device.getName());
    sendMessage(ACTION_WRITE_CHARACTER, bundle);
}
```

Listing 3.21: Stato Connessione: scrittura Caratteristica

mBleConnectionReceiver recupera il Custom Service A&D e la sua Caratteristica, dopodiché, tramite il metodo gatt.readCharacteristic(), legge sul dispositivo la Caratteristica passata come parametro. Il risultato dell'operazione di lettura viene riportato dalla callback onCharacteristicRead. La callback onCharacteristicRead crea un Bundle contenente:

- Il risultato dell'operazione di lettura.
- L'UUID della Caratteristica letta dal dispositivo remoto.

Queste informazioni vengono inviate tramite sendMessage() al BR mBleConnectionReceiver, che si occuperà di gestire l'azione ACTION_READ_CHARACTER.

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt,
    BluetoothGattCharacteristic characteristic, int status) {
    super.onCharacteristicRead(gatt, characteristic, status);
    Bundle bundle = new Bundle();
    bundle.putBoolean(KEY_RESULT,
                        (status == BluetoothGatt.GATT_SUCCESS));
    bundle.putString(KEY_UUID_STRING,
                        characteristic.getUuid().toString());
    sendMessage(ACTION_READ_CHARACTER, bundle);
}
```

Listing 3.22: Stato Connessione: lettura Caratteristica

mBleConnectionReceiver recupera il Custom Service A&D e la sua Caratteristica. A questo punto viene assegnato alla variabile pktSize il valore memorizzato nella Caratteristica, interpretato nel formato uint8. Se il valore restituito è uguale a 3, viene chiamato il metodo setupDateTime(). SetupDateTime() si occupa di impostare la data corrente nella Caratteristica Date Time del dispositivo. Viene recuperata innanzitutto la Caratteristica Date Time. Successivamente, viene chiamato il metodo writeCharacteristic() della classe DateTime, il quale si occupa di trasformare la data passata come parametro in un array di byte. Ogni byte rappresenta in successione: anno, mese, giorno, ora, minuti e secondi. L'array di byte viene memorizzato come valore nella Caratteristica attraverso la funzione setValue(). Infine, tornando al metodo setupDateTime, viene scritta la Caratteristica aggiornata sul dispositivo. Completata l'operazione di scrittura si ritorna alla callback onCharacteristicWrite e il BR mBleConnectionReceiver si occuperà di gestire l'azione ACTION_WRITE_CHARACTER.

```
public void setupDateTime() {
   BluetoothGattService gattService = getGattSearvice(gatt);
   if (gattService != null) {
        BluetoothGattCharacteristic characteristic =
            gattService.getCharacteristic(ADGattUUID.DateTime);
        if (characteristic != null) {
```

```
7               Calendar data=Calendar.getInstance();
8               characteristic = DateTime.writeCharacteristic
9                   (characteristic, Calendar.getInstance());
10              gatt.writeCharacteristic(characteristic);
11          }
12      }
13  }
```

Listing 3.23: Scrittura della data

Viene confrontato l'UUID della Caratteristica Date Time con l'UUID della Caratteristica appena scritta sul dispositivo. Se questi combaciano e se la scrittura di data e ora è andata a buon fine, viene richiamato il metodo disconnectDevice(). DisconnectDevice() imposta un Bundle, contenente un booleano indicante il successo della scrittura. Il Bundle viene inviato a mBleConnectionReceiver per la gestione dell'azione ACTION_DEVICE_SETUP. Viene chiamato, inoltre, il metodo disconnect(), che contiene le funzioni per chiudere la connessione col dispositivo.

- gatt.disconnect() termina la connessione stabilita col dispositivo.
- gatt.close() interrompe la possibilità, per il GATT Client del device Android, di comunicare con il GATT Server del dispositivo A&D. Il sistema rilascia le risorse.

```
1  public void disConnectDevice() {
2      if (mSetupDevice != null) {
3          boolean isSuccess = mSetupDevice.isSetupFinish;
4          Bundle bundle = new Bundle();
5          bundle.putBoolean(KEY_RESULT, isSuccess);
6          sendMessage(ACTION_DEVICE_SETUP, bundle);
7
8          mSetupDevice.disconnect();
9          mSetupDevice = null;
10     }
11 }
```

```
public void disconnect() {
    if (gatt != null) {
        gatt.disconnect();
        gatt.close();
        gatt = null;
    }
}
```

Listing 3.24: Chiusura della connessione

mBleConnectionReceiver richiama onDevicePairingResult(), che esegue le operazioni per mostrare il messaggio di conferma del pairing. OnDevicePairingResult() esegue le seguenti operazioni:

1. Nasconde il layout della schermata istruzioni del dispositivo.
2. Crea l'intent che si occuperà di chiamare l'Activity DialogActivity per l'impostazione e la visualizzazione grafica del messaggio di conferma.
3. Lancia l'Intent con il metodo startActivityForResult(), insieme ad un REQUEST_CODE specifico.

DialogActivity cambia il contenuto della schermata con il layout del messaggio. Il layout del messaggio è una finestra di dialogo che informa l'utente che il pairing è avvenuto con successo. L'utente può interagire con la finestra cliccando sul button di conferma sottostante al corpo del messaggio. Quando l'utente clicca sul button di conferma, DialogActivity ritornerà un RESULT_CODE, che verrà gestito dalla callback onActivityResult() per il caricamento della Dashboard.

OnActivityResult() confronta il requestCode ricevuto con quello inviato in precedenza per capire se il RESULT_CODE è stato ottenuto dalla DialogActivity. Successivamente carica la DashboardActivity.

```
@Override
protected void onDevicePairingResult(boolean result) {
  super.onDevicePairingResult(result);
  if(result && !isShowDialog) {
    isShowDialog = true;
    ViewGroup dialogLayout =
        (ViewGroup)findViewById(R.id.dialog_layout);
    dialogLayout.setVisibility(View.INVISIBLE);

    Intent intent = new Intent(InstructionActivity.this,
                               DialogActivity.class);
    startActivityForResult(intent, DialogActivity.REQUEST_CODE);
  }
}
```

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  if(requestCode == DialogActivity.REQUEST_CODE) {
    Intent intent = new Intent(InstructionActivity.this,
        DashboardActivity.class);

    startActivity(intent);
    finish();
  }
}
```

Listing 3.25: Impostazione del messaggio di conferma e caricamento della Dashboard

3.4 Sincronizzazione

Nella Dashboard, oltre a visualizzare i dati delle misurazioni effettuate, è possibile sincronizzare i dati stessi con quelli rilevati da un dispositivo A&D, visualizzando immediatamente i dati aggiornati. Ogni dispositivo medico A&D inizia la fase di advertising in maniera automatica al termine della misurazione dei dati. Tuttavia, l'applicazione potrà connettersi con il dispositivo che sta facendo advertising solo dopo aver svolto alcune procedure preliminari:

1. Risalire ai dispositivi accoppiati e verificare che il dispositivo in fase di advertising rientri tra questi.
2. Creare un Service che si occupi di comunicare col dispositivo A&D.
3. Registrare un Broadcast Receiver che si ponga in ascolto e gestisca gli eventi scaturiti durante il processo di sincronizzazione dei dati.

Viene creato un Bound Service ("BleReceivedService") che permette di:

- Eseguire le classiche operazioni di connessione BLE.
- Ricevere notifiche dal dispositivo A&D.

Il Service viene dichiarato nel Manifest e viene implementata la relativa classe.

```
<service android:name=
    "jp.co.aandd.bleSimpleApp.gatt.BleReceivedService" />
```

```
public class BleReceivedService extends Service {
```

Listing 3.26: Creazione del Service

Per la sincronizzazione si vuole fare in modo che il Service venga eseguito continuamente in background, finché l'utente rimane sulla Dashboard. Questo permette all'applicazione di ricevere i dati in qualunque momento e, siccome il dispositivo A&D tenta di trasferire i dati subito dopo la misurazione, è importante mantenere il Service attivo per rispettare il vincolo di sincronizzazione automatica.

La funzione doStartService() avvia il Service e registra un Broadcast Receiver, il quale si occupa di gestire l'azione specificata nell'IntentFilter.

Il Service viene avviato tramite la funzione startService(), specificando, nell'Intent passato come parametro, il Service stesso da eseguire (BleReceivedService). L'esecuzione del Service verrà poi arrestata, tramite la funzione stopService(), alla chiusura della Dashboard Activity (quindi al richiamo di onDestroy). Il metodo registerReceiver() registra il BR bleServiceReceiver che si occuperà di gestire l'azione ACTION_BLE_SERVICE.

```
1  private void doStartService() {
2      Intent intent1 = new Intent(this, BleReceivedService.class);
3      startService(intent1);
4      if (!mIsBleReceiver) {
5          IntentFilter filter = new IntentFilter
6              (BleReceivedService.ACTION_BLE_SERVICE);
7          registerReceiver(bleServiceReceiver, filter);
8          mIsBleReceiver = true;
9      }
10 }
```

Listing 3.27: Avvio del Service e registrazione del Broadcast Receiver

Una volta che sono stati recuperati e mostrati sull'interfaccia tutti i dati misurati, precedentemente salvati nel DB, viene verificato se il Bluetooth del device Android è attualmente attivo. Se il Bluetooth non è attivo, viene lanciato un Intent per mostrare una finestra di dialogo che consenta all'utente di attivare il Bluetooth.

BluetoothAdapter.ACTION_REQUEST_ENABLE è l'Activity di sistema che permette di visualizzare la finestra di dialogo e completare l'attivazione del Bluetooth sul device. Il risultato dell'interazione viene gestito dalla callback onActivityResult, la quale chiama il metodo doBindBleReceivedService() per associare il Client al Service. Se invece il Bluetooth è già attivo, viene chiamata direttamente la funzione doBindBleReceivedService().

```
if (!bluetoothAdapter.isEnabled()) {
    if (!mIsCheckBleetoothEnabled) {
        mIsCheckBleetoothEnabled = true;
        Intent intent = new Intent
            (BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(intent, REQUEST_ENABLE_BLUETOOTH);
        return;
    }
} else {
    doBindBleReceivedService();
}
```

Listing 3.28: Verifica attivazione Bluetooth

Seguendo il procedimento descritto nel paragrafo di Pairing, viene restituita l'interfaccia IBinder per comunicare col Service e viene creata la connessione tra Client e Service.

```
private void doBindBleReceivedService() {
    if (!mIsBindBleReceivedServivce) {
        bindService(new Intent(DashboardActivity.this,
                    BleReceivedService.class),
                    mBleReceivedServiceConnection,
                    Context.BIND_AUTO_CREATE);
        mIsBindBleReceivedServivce = true;
    }
}
```

Listing 3.29: Creazione della connessione

Una volta che la connessione è stata stabilita, viene fatta partire la scansione dei dispositivi. Se un dispositivo è attualmente connesso, chiudo la connessione in corso. Per evitare scansioni contemporanee, viene settata la variabile isScanning a true. L'avvio effettivo della scansione, effettuato da startLeScan(), viene eseguito, tramite la funzione runOnUIThread(), sul thread dell'interfaccia utente. Questo permette di non pesare l'operazione di scansione sul thread principale, ma di eseguirla su un altro thread, aggiornando poi il risultato sul thread principale.

```
1  private void startScan() {
2      if (shouldStartConnectDevice) {
3          return;
4      }
5      if (BleReceivedService.getInstance() != null) {
6          if (BleReceivedService.getInstance()
7                  .isConnectedDevice()) {
8              BleReceivedService.getInstance().disconnectDevice();
9          }
10         isScanning = true;
11         runOnUiThread(new Runnable() {
12             @Override
13             public void run() {
14                 BleReceivedService.getInstance()
15                     .getBluetoothManager().getAdapter()
16                         .startLeScan(mLeScanCallback);
17             }
18         });
19     }
20 }
```

Listing 3.30: Avvio scansione su thread differito

mLeScanCallback viene utilizzata per fornire i risultati della scansione LE. La callback onLeScan viene invocata al rilevamento di un dispositivo BLE durante la scansione. Il dispositivo rilevato viene rappresentato da una classe BluetoothDevice. La funzione isAbleToConnectDevice() verifica che sia possibile connettersi al dispositivo. Se la verifica ha esito positivo, la funzione ritorna true e si procede a:

1. Interrompere la scansione in corso, attraverso la funzione doStopLeScan().
2. Connettersi al dispositivo individuato, tramite il metodo connectDevice().

doStopLeScan() richiama al suo interno la funzione stopLeScan(), mentre connectDevice() richiama connectGatt(). Le due funzioni sono spiegate nella sezione precedente.

```
private LeScanCallback mLeScanCallback = new LeScanCallback() {
    @Override
    public void onLeScan(final BluetoothDevice device,
                         int rssi, byte[] scanRecord) {
        if (!isScanning) {
            return;
        }
        if (device.getName() != null) {
            if (isAbleToConnectDevice(device, scanRecord)
                    && !shouldStartConnectDevice) {
                shouldStartConnectDevice = true;
                if (device.getName() != null) {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            doStopLeScan();
                            try {
                                Thread.sleep(50);
                            } catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                            BleReceivedService.getInstance()
                                .connectDevice(device);
                        }
                    });
                }
```

Listing 3.31: Tentativo di connessione con il dispositivo individuato

Per verificare che sia possibile connettersi al dispositivo rilevato, vengono eseguite sostanzialmente 2 operazioni:

1. Viene eseguita la funzione getBondedDevices() della classe Bluetooth Adapter, che restituisce un insieme (Set) contenente tutti i dispositivi Bluetooth accoppiati al device Android.

2. Se il dispositivo rilevato è contenuto nell'insieme e il nome del dispositivo contiene la sigla A&D, viene restituito true e la verifica ha esito positivo. In caso contrario, viene restituito false e la verifica ha esito negativo, quindi non è possibile connettersi al dispositivo senza aver prima fatto il pairing.

```
1  private boolean isAbleToConnectDevice(BluetoothDevice device,
2                                        byte[] scanRecord) {
3      if (BleReceivedService.getInstance().isConnectedDevice()) {
4          return false;
5      }
6      BluetoothAdapter bluetoothAdapter = BleReceivedService
7          .getInstance().getBluetoothManager().getAdapter();
8      if (bluetoothAdapter != null) {
9          Set<BluetoothDevice> pairingDevices = bluetoothAdapter
10             .getBondedDevices();
11         if (device.getName() != null) {
12             return pairingDevices.contains(device)
13                 && device.getName().contains("A&D");
14         }
15     }
16     return false;
17 }
```

Listing 3.32: Verifica pairing del dispositivo

Eseguita la connessione col dispositivo, vengono forniti i risultati al Client attraverso la classe bluetoothGattCallback.

Le callback implementate sono le stesse viste nel processo di connessione per il Pairing (onConnectionStateChange, onServicesDiscovered, onCharacteristicRead, onCharacteristicWrite). Cambiano le azioni che vengono inviate, come vengono gestite e il BR (bleServiceReceiver) che si occupa di gestirle.

Rispetto al metodo usato nel Pairing, in questo caso, il BR viene registrato per gestire un'unica azione (ACTION_BLE_SERVICE), la quale viene suddivisa in una serie di tipi ("TYPE") che indicano il tipo di azione da eseguire.

sendBroadcast() è un metodo intermediario al sendBroadcast effettivo, il quale si occuperà di lanciare l'Intent contenente l'azione (e il tipo di azione) da eseguire. Vengono implementati 3 metodi sendBroadcast diversi, distinti in base al numero e al tipo di parametri passati in ingresso (polimorfismo). Ogni metodo crea un Intent con l'azione generale da eseguire (ACTION_BLE_SERVICE) e aggiunge all'Intent stesso un numero specifico di dati extra, in base alle informazioni necessarie da inviare al BR. Tra questi dati rientra anche il tipo di azione, ricevuto come parametro e identificato dal nome EXTRA_TYPE.

```
1 private void sendBroadcast(String type, BluetoothDevice device,
2                             int status) {
3     Intent intent = new Intent(ACTION_BLE_SERVICE);
4     intent.putExtra(EXTRA_TYPE, type);
5     intent.putExtra(EXTRA_ADDRESS, device.getAddress());
6     intent.putExtra(EXTRA_STATUS, status);
7     intent.putExtra(EXTRA_DEVICE_NAME, device.getName());
8     sendBroadcast(intent);
9 }
```

Listing 3.33: Uno dei 3 metodi intermediari per l'inivio dell'Intent

Una volta che sendBroadcast() lancia l'Intent passato come parametro, il sistema richiama la callback onReceive() di bleServiceReceiver, che si occupa di gestire l'azione generale ACTION_BLE_SERVICE. La callback recupera innanzitutto il tipo di azione dall'Intent (informazione relativa al nome EXTRA_TYPE).

```
1 private final BroadcastReceiver bleServiceReceiver =
2     new BroadcastReceiver() {
3     @Override
4     public void onReceive(Context context, Intent intent) {
5         String type = intent.getExtras()
6                         .getString(BleReceivedService.EXTRA_TYPE);
```

Listing 3.34: Recupero tipo di azione da gestire

In base al tipo di azione stessa, la callback eseguirà il blocco di istruzioni specifico per gestire quel tipo di azione. Per individuare il blocco di istruzioni vengono confrontate le costanti, contenute nella classe del Service e riferite ai tipi di azione che si vogliono gestire, con il tipo recuperato dall'Intent. Il blocco di istruzioni è quello relativo alla condizione che viene verificata: costante.equals(tipo).

I tipi di azione che vengono gestiti dal BR sono i seguenti:

- TYPE_GATT_CONNECTED.
- TYPE_GATT_DISCONNECTED.
- TYPE_GATT_SERVICES_DISCOVERED.
- TYPE_CHARACTERISTIC_READ.
- TYPE_CHARACTERISTIC_WRITE.
- TYPE_INDICATION_VALUE.

TYPE_GATT_CONNECTED viene gestito quando il GATT Client si connette al GATT Server.

Viene mostrato l'indicatore di caricamento e viene impostato il ritardo di sincronizzazione della data e dei valori misurati. La funzione discoverServices() rileva i Servizi offerti dal dispositivo remoto e ritorna true se il rilevamento è stato avviato. Una volta completato il rilevamento viene richiamata la callback onServicesDiscovered.

```
if (BleReceivedService.TYPE_GATT_CONNECTED.equals(type)) {
    showIndicator(getResources()
                .getString(R.string.indicator_start_receive));
    BleReceivedService.getGatt().discoverServices();
    setDateTimeDelay = Long.MIN_VALUE;
    indicationDelay = Long.MIN_VALUE;
}
```

Listing 3.35: Tipi di azione da gestire: Gatt Client connesso

TYPE_GATT_SERVICES_DISCOVERED viene gestito quando l'elenco dei Servizi individuati nel dispositivo è stato aggiornato.

Viene utilizzato un Handler (uiThreadHandler) per pianificare l'esecuzione di un oggetto Runnable. Il Runnable, o meglio il metodo run() che implementa, viene eseguito dopo 500L, che rappresenta il ritardo (in millisecondi) di chiamata del metodo. Quindi, dopo mezzo secondo, viene eseguito il metodo requestReadFirmRevision(). Questo metodo si occupa di leggere la Caratteristica Firmware Revision String, contenuta nel Device Information Service. Servizio e Caratteristica vengono recuperati specificando i relativi UUID. Il risultato dell'operazione di lettura viene riportato dalla callback onCharacteristicRead.

```
if (BleReceivedService.TYPE_GATT_SERVICES_DISCOVERED
    .equals(type)) {
    if (shouldStartConnectDevice) {
        if (BleReceivedService.getInstance() != null) {
```

```
            uiThreadHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    BleReceivedService.getInstance()
                        .requestReadFirmRevision();
                }
            }, 500L);
        }
```

Listing 3.36: Tipi di azione da gestire: Aggiornamento Servizi

TYPE_CHARACTERISTIC_READ viene gestito quando l'operazione di lettura è stata completata.

Viene recuperato il valore in byte (memorizzato nella Caratteristica) e viene trasformato in valore stringa. Il valore ottenuto viene confrontato con ogni stringa contenuta in un array prestabilito. L'array contiene un gruppo di revisioni firmware del dispositivo, sulla base delle quali viene stabilito un certo ritardo di sincronizzazione. Se il valore firmware trovato corrisponde ad uno di quelli contenuti nell'array, il ritardo viene impostato a 40L, altrimenti a 100L. Successivamente, viene impostato un Handler per la sincronizzazione della data con ritardo aggiornato.

Trascorso il periodo di tempo specificato, viene eseguita la funzione setupDateTime(), che si occupa di scrivere data e ora correnti nella Caratteristica Date Time del dispositivo (vedi sezione precedente). Finita la scrittura della data, viene richiamata la callback onCharacteristicWrite.

```
} else if (BleReceivedService.TYPE_CHARACTERISTIC_READ
            .equals(type)) {
    if (shouldStartConnectDevice) {
        byte[] firmRevisionBytes = intent.getByteArrayExtra
            (BleReceivedService.EXTRA_VALUE);
        String firmRevision = new String(firmRevisionBytes);
        String[] firmRevisionArray = getResources()
            .getStringArray(R.array.firm_revision_group1);
        boolean isGroup1 = false;
```

```
10          for (String revision : firmRevisionArray) {
11              if (revision.contains(firmRevision)) {
12                  isGroup1 = true;
13                  break;
14              }
15          }
16
17          if (isGroup1) {
18              setDateTimeDelay = 40L;
19              indicationDelay = 40L;
20          } else {
21              setDateTimeDelay = 100L;
22              indicationDelay = 100L;
23          }
24          uiThreadHandler.postDelayed(new Runnable() {
25              @Override
26              public void run() {
27                  BluetoothGatt gatt = BleReceivedService.getGatt();
28                  boolean settingResult;
29                  if (gatt != null) {
30                      String deviceName = gatt.getDevice()
31                          .getName();
32                      settingResult = BleReceivedService
33                          .getInstance()
34                          .setupDateTime(gatt);
35                      if (!settingResult) {
36                          dismissIndicator();
37                      }
38                  } else {
39                      dismissIndicator();
40                  }
41              }
42          }, setDateTimeDelay);
43      }
44  }
```

Listing 3.37: Tipi di azione da gestire: Lettura di una Caratteristica

TYPE_CHARACTERISTIC_WRITE viene gestito quando l'operazione di scrittura è stata completata.

Viene confrontato l'UUID della Caratteristica Date Time con l'UUID della Caratteristica appena scritta sul dispositivo. Se questi combaciano, viene impostato un Handler per la sincronizzazione dei valori misurati con ritardo aggiornato. Trascorso il periodo di tempo specificato, viene eseguita la funzione setIndication(), che si occupa di avviare la procedura di sincronizzazione dei dati contenuti nella Caratteristica Measurement del dispositivo remoto. Viene preferita l'espressione "sincronizzazione", invece di "trasferimento", perchè l'operazione che viene eseguita, a livello di codice, non è un vero e proprio invio dei dati, ma è il Client che, a seguito di una notifica inviata dal Server, legge i dati contenuti nella Caratteristica aggiornata del Server e li memorizza localmente.

```
} else if (BleReceivedService.TYPE_CHARACTERISTIC_WRITE
    .equals(type)) {
    String characteristicUuidString = intent.getExtras()
        .getString(BleReceivedService.EXTRA_CHARACTERISTIC_UUID);
    if (characteristicUuidString
        .equals(ADGattUUID.DateTime.toString())) {
        if (shouldStartConnectDevice) {
            uiThreadHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    BluetoothGatt gatt = BleReceivedService
                        .getGatt();
                    boolean writeResult = BleReceivedService
                        .getInstance().setIndication(gatt, true);
                    if (writeResult == false) {
                        dismissIndicator();
                    }
                }
            }, indicationDelay);
        }
```

Listing 3.38: Tipi di azione da gestire: Scrittura di una Caratteristica

La procedura è la seguente: [9]

1. Il Server invia una notifica al Client per avvertirlo che è disponibile in memoria una nuova misurazione, quindi che la Caratteristica Measurement contenuta nel Server è stata aggiornata. Questa notifica prende il nome di "Indication". Il Server rimane poi in attesa di una risposta dal Client.
2. Il Client riceve la notifica. Legge i dati contenuti nella Caratteristica e li memorizza in un DB locale. Al termine delle operazioni, il Client invia un messaggio di conferma al Server.
3. Quando il Server riceve la conferma del Client, controlla se sono disponibili nuovi dati in memoria e, nel caso, invia un'altra notifica.

La Caratteristica Measurement risulta aggiornata quando:

- Sono disponibili nuovi dati, a causa di una misurazione appena effettuata.
- Sono disponibili nuovi dati, a causa di misurazioni effettuate precedentemente e non ancora sincronizzate (rimaste salvate in memoria).

Ogni volta che la Caratteristica Measurement risulta aggiornata, quindi il Server invia una notifica al Client, viene invocata la callback onCharacteristicChanged. In base al dispositivo che invia la notifica, viene individuata la Caratteristica (Blood Pressure Measurement o Weight Scale Measurement) e viene chiamato il metodo readCharacteristic() della Classe relativa implementata. Questo metodo permette di recuperare i dati memorizzati nella Caratteristica e di inserirli in un oggetto Bundle.

Ogni valore viene recuperato in un determinato campo della Caratteristica (Unit, Time Stamp, Pulse Rate, Flags, ecc.) e viene interpretato secondo un certo formato. Il parametro offset, specificato nei metodi get della Caratteristica, indica la posizione in corrispondenza della quale è possibile trovare il valore.

Il Bundle risultante, contenente tutti i dati recuperati dalla Caratteristica, viene inviato tramite sendBroadcast() al BR bleServiceReceiver, che gestisce il tipo di azione corrispondente (TYPE_INDICATION_VALUE).

```
@Override
public void onCharacteristicChanged(BluetoothGatt gatt,
                    BluetoothGattCharacteristic characteristic) {
    if (ADGattUUID.BloodPressureMeasurement
        .equals(characteristic.getUuid())) {
        Bundle valueBundle = BloodPressureMeasurement
            .readCharacteristic(characteristic);
        sendBroadcast(TYPE_INDICATION_VALUE,
                ADGattUUID.BloodPressureMeasurement.toString(),
                valueBundle);
    } else if (ADGattUUID.WeightScaleMeasurement
                .equals(characteristic.getUuid())) {
        Bundle valueBundle = WeightMeasurement
            .readCharacteristic(characteristic);
        sendBroadcast(TYPE_INDICATION_VALUE,
                ADGattUUID.WeightScaleMeasurement.toString(),
                valueBundle);
    }
}
```

Listing 3.39: Invio notifica al Client per lettura Caratteristica

```
// Unit
offset+=1;
bundle.putFloat(KEY_SYSTOLIC, characteristic.getFloatValue
            (BluetoothGattCharacteristic.FORMAT_SFLOAT, offset));
offset+=2;
bundle.putFloat(KEY_DIASTOLIC, characteristic.getFloatValue
            (BluetoothGattCharacteristic.FORMAT_SFLOAT, offset));

// Time Stamp
bundle.putInt(KEY_YEAR, characteristic.getIntValue
            (BluetoothGattCharacteristic.FORMAT_UINT16, offset));
```

```
12 offset+=2;
13 bundle.putInt(KEY_MONTH, characteristic.getIntValue
14             (BluetoothGattCharacteristic.FORMAT_UINT8, offset));
15 offset+=1;
16 bundle.putInt(KEY_DAY, characteristic.getIntValue
17             (BluetoothGattCharacteristic.FORMAT_UINT8, offset));
18 offset+=1;
19
20 // Pulse Rate
21 bundle.putFloat(KEY_PULSE_RATE, characteristic.getFloatValue
22             (BluetoothGattCharacteristic.FORMAT_SFLOAT, offset));
23 offset+=2;
```

Listing 3.40: Recupero dati Caratteristica

TYPE_INDICATION_VALUE viene gestita quando il Client termina la lettura dei dati contenuti nella Caratteristica Measurement del dispositivo.

Viene chiamato il metodo receivedData() passando come argomento il Bundle ricevuto e l'UUID della Caratteristica Measurement relativa al dispositivo remoto connesso. ReceivedData() recupera i dati contenuti nel Bundle per poi memorizzarli nel DB locale usato dall'applicazione. Inizialmente, per capire a quale Caratteristica sono associati i dati contenuti nel Bundle, viene confrontato l'UUID ricevuto in ingresso con i due UUID delle Caratteristiche disponibili (Blood Pressure Measurement o Weight Scale Measurement). Individuata la Caratteristica corrispondente, vengono recuperati dal Bundle i dati relativi alla Caratteristica stessa e vengono inseriti nella tabella appropriata sul DB. Completato l'inserimento dei dati, vengono aggiornati immediatamente i valori visualizzati nella Dashboard con quelli appena memorizzati.

Rimuovo, se presenti, le callback pianificate per l'esecuzione dell'oggetto Runnable, passato come parametro, che sono ancora in attesa di essere invocate. Se sono disponibili nuovi dati da sincronizzare, il Server invia un'altra "Indication" al Client e riparte la procedura di sincronizzazione dalla callback onCharacteristicChanged.

Se entro 4 secondi il Client non riceve alcuna notifica dal Server, viene eseguito l'oggetto disableIndicationRunnable.

```
} else if (BleReceivedService.TYPE_INDICATION_VALUE
            .equals(type)) {
    setIndicatorMessage(getResources().getString
                        (R.string.indicator_during_receive));
    Bundle bundle = intent.getBundleExtra
        (BleReceivedService.EXTRA_VALUE);
    String uuidString = intent.getExtras().getString
        (BleReceivedService.EXTRA_CHARACTERISTIC_UUID);
    receivedData(uuidString, bundle);
    uiThreadHandler.removeCallbacks(disableIndicationRunnable);
    uiThreadHandler.postDelayed(disableIndicationRunnable, 4000L);
}
```

Listing 3.41: Tipi di azione da gestire: Termine lettura dati

Il metodo run(), eseguito nell'oggetto Runnable, svolge le seguenti operazioni:

1. Richiama la funzione setIndication() per disabilitare le notifiche e stoppare così la procedura di sincronizzazione dei dati.
2. Rimuove l'indicatore di caricamento.
3. Termina la connessione stabilita col dispositivo.

Dopo aver eseguito gatt.disconnect(), viene richiamata la callback onConnectionStateChange, che riporta il nuovo stato della connessione: DISCONNECTED.

```
Runnable disableIndicationRunnable = new Runnable() {
    @Override
    public void run() {
        setIndicatorMessage(getResources().getString
                        (R.string.indicator_complete_receive));
        BluetoothGatt gatt = BleReceivedService.getGatt();
```

```
7           if (BleReceivedService.getInstance() != null) {
8               boolean writeResult = BleReceivedService
9                   .getInstance().setIndication(gatt, false);
10              if (writeResult == false) {
11                  dismissIndicator();
12              }
13              if (gatt != null) {
14                  gatt.disconnect();
15              }
16          }
17      }
18  };
```

Listing 3.42: Termine procedura di sincronizzazione

TYPE_GATT_DISCONNECTED viene gestito quando il GATT Client si disconnette dal GATT Server.

Vengono eseguite le funzioni per interrompere la comunicazione col dispositivo e avviare una nuova scansione.

```
1  } else if (BleReceivedService.TYPE_GATT_DISCONNECTED
2              .equals(type)) {
3      dismissIndicator();
4      if (shouldStartConnectDevice) {
5          BleReceivedService.getInstance().disconnectDevice();
6          uiThreadHandler.postDelayed(new Runnable() {
7              @Override
8              public void run() {
9                  shouldStartConnectDevice = false;
10                 if (!isScanning) {
11                     doStartLeScan();
12                 }
13             }
14         }, 80L);
15     }
```

Listing 3.43: Tipi di azione da gestire: Gatt Client disconnesso

3.5 Visualizzazione

Quando viene eseguito l'accesso alla Dashboard, prima di iniziare la scansione dei dispositivi, vengono inizializzate le View contenenti i dati memorizzati sul DB locale. Le View sono dei layout con struttura già definita (Figura 3.5), in cui vengono semplicemente inseriti i valori numerici delle misurazioni. Ogni View è relativa ad un dispositivo A&D e viene distinta in base al colore, al nome del dispositivo stesso e ai parametri che raccoglie.

È possibile scorrere lo storico dei valori misurati nelle diverse date, grazie a due frecce posizionate sotto la barra dell'app. In particolare: cliccando la freccia a sinistra vengono visualizzati i valori della data precedente, cliccando quella a destra i valori della data successiva.

Per rispettare i vincoli di universalità e scalabilità, evitando variazioni notevoli al software, nel caso di aggiunta di nuovi dispositivi, viene utilizzata per la View una struttura universale, i cui contenuti vengono aggiornati a run-time in maniera semplice e intuitiva.

Figura 3.5: View relativa al Misuratore di pressione

La struttura di ogni View è organizzata in questo modo:

- Due frecce in alto, che vengono utilizzate per indicare, all'utente che scorre lo storico dei valori, la presenza di ulteriori valori per la singola View, quindi per il dispositivo in questione. In particolare, la freccia a sinistra viene visualizzata quando sono disponibili dati precedenti, la freccia a destra quando sono disponibili dati successivi.
- Un campo data, nel mezzo delle frecce, che specifica la data e l'ora in cui è stata effettuata la misurazione mostrata. Viene aggiornato a run-time.
- Il nome del dispositivo a cui è riferita la View.
- Un'icona di un colore simbolico.
- Un numero di campi pari alle misurazioni fornite dal dispositivo. Ogni campo è formato da valore e unità e viene aggiornato a run-time: all'apertura della Dashboard, alla fine della procedura di sincronizzazione o durante l'interazione con le frecce.
- I titoli dei parametri misurati dal dispositivo. Questi vengono disposti ognuno sotto il valore associato al parametro stesso. I parametri vengono impostati all'apertura della Dashboard.

Le View vengono inserite all'interno di blocchi (FrameLayout), predisposti sul layout della Dashboard e organizzati in pila. Questi blocchi consentono di separare la struttura del layout principale (Dashboard) dalla lista di View che vengono mostrate. Grazie al tag <include/>, infatti, le View vengono incorporate all'interno dei FrameLayout e adattate alla larghezza dello schermo. In questo modo, è possibile modificare facilmente il contenuto del blocco stesso in caso di variazioni o cambi di posizione delle View.

Ogni volta che l'Activity della Dashboard viene messa in primo piano, quindi al richiamo della callback OnResume, i campi delle View vengono aggiornati tramite la funzione syncAllMeasuDatas(). Questa funzione si occupa di aggiornare le View in maniera asincrona, tramite l'utilizzo di una classe AsyncTask ("SelectDataFromDatabase"). Essa permette di recuperare i dati in background leggendoli da DB e di mostrarli sull'interfaccia utente. La classe fa uso di 3 metodi:

- onPreExecute: richiamato prima dell'esecuzione in background. Si occupa di inizializzare alcune variabili per le operazioni successive.
- doInBackground: richiamato subito dopo onPreExecute. Gestisce in background il recupero dei dati da DB e l'ordinamento storico degli stessi.
- onPostExecute: richiamato alla fine dell'esecuzione in background. Svolge le operazioni di pubblicazione dei risultati sull'interfaccia utente.

Una volta che viene invocato execute() sull'istanza di SelectDataFromDatabase, i metodi iniziano le operazioni di recupero e pubblicazione dei dati relativi ad un dispositivo specifico. Le operazioni vengono eseguite in un worker thread (thread separato da quello principale), mentre i risultati vengono pubblicati sul thread dell'interfaccia utente.

```
1  public void syncAllMeasuDatas(boolean isSendBroadcast) {
2      if (mDataBase == null) {
3          return;
4      }
5      if (ANDMedicalUtilities.APP_STAND_ALONE_MODE) {
6          setDispMeasuData(MEASU_DATA_TYPE_BP, null);
7          new SelectDataFromDatabase
8              (MEASU_DATA_TYPE_BP, isSendBroadcast, null).execute();
9          setDispMeasuData(MEASU_DATA_TYPE_WS, null);
10         new SelectDataFromDatabase
11             (MEASU_DATA_TYPE_WS, isSendBroadcast, null).execute();
```

Listing 3.44: Aggiornamento delle View: Dashboard in primo piano

Per popolare/modificare il contenuto delle View viene chiamata una funzione setData(), che riceve i dati come parametro e li inserisce nei relativi campi della View in questione.

Un altro momento in cui vengono aggiornati i dati di una View è alla fine della procedura di sincronizzazione, in particolare, dopo l'inserimento su DB dei dati recuperati nella Caratteristica Measurement di un dispositivo. In questo caso, viene chiamata la funzione syncMeasudata(), che permette di visualizzare i dati appena sincronizzati sulla View corrispondente al dispositivo A&D utilizzato.

syncMeasudata crea un'istanza della classe SelectDataFromDatabase, specificando la tipologia di dati (MEASU_DATA_TYPE_BP o MEASU_DATA_TYPE_WS) che si vogliono inserire nella View. Sull'istanza creata viene invocato execute() per avviare l'aggiornamento asincrono in background.

```
1  public void syncMeasudata(int dataType, boolean isSendBroadcast,
2                             MeasureDataSyncListener listener) {
3      if (mDataBase == null) {
4          return;
5      }
6      if (ANDMedicalUtilities.APP_STAND_ALONE_MODE) {
7          new SelectDataFromDatabase(dataType, isSendBroadcast,
8                                     listener).execute();
9      }
10 }
```

Listing 3.45: Aggiornamento delle View: Fine sincronizzazione

L'ultimo caso di aggiornamento delle View avviene quando l'utente interagisce con le frecce per scorrere lo storico dei valori misurati. A questo proposito, è stato associato ad ogni freccia un Click Listener che, una volta premuto sulla freccia corrispondente, aggiorna i dati mostrati nelle View con quelli del giorno precedente o successivo.

Se l'evento Click è stato generato dalla freccia sinistra, viene chiamato il metodo moveDatasToThePast(), che si occupa di recuperare le misurazioni della data precedente a quella attualmente mostrata. Se l'evento Click è stato generato dalla freccia destra, viene chiamato il metodo moveDatasToTheFuture(), che si occupa di recuperare le misurazioni della data successiva a quella attualmente mostrata. In entrambi i casi, viene eseguito, dopo il recupero dei dati, un "refresh" delle View per aggiornare le misurazioni e la data relativa, che vengono mostrate a livello di interfaccia.

```
rightArrow.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        AndMedical_App_Global appGlobal =
            (AndMedical_App_Global) getApplication();
        MeasuDataManager manager = appGlobal
            .getMeasuDataManager();
        if (manager != null) {
            manager.moveDatasToTheFuture();
            refreshDisplay();
        }
    }
});
leftArrow.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        AndMedical_App_Global appGlobal =
            (AndMedical_App_Global) getApplication();
        MeasuDataManager manager = appGlobal
            .getMeasuDataManager();
        if (manager != null) {
            manager.moveDatasToThePast();
            refreshDisplay();
        }
    }
});
```

Listing 3.46: Aggiornamento delle View: Storico dei valori

Conclusioni

L'impiego sempre più frequente di tecnologie smart nell'industria sanitaria, unito alla necessità di trovare soluzioni alternative nei percorsi di cura dei pazienti, modifica la concezione di cura medica: da quella tradizionale, basata unicamente sul rapporto diretto tra medico e paziente, a quella moderna, basata sulla telecomunicazione e sui principi dell'Internet of Things. Nella moderna concezione di pratica medica, si vuole garantire una continuità della cura, che non si limiti ad una valutazione soggettiva del medico curante, in base ad esempio all'esito di alcuni esami, ma che preveda un monitoraggio costante dei parametri vitali di un paziente per valutare l'andamento dei valori e l'adozione di misure preventive. [3]

Il miglior modo per esercitare un monitoraggio continuo, anche dopo una visita e al di fuori del reparto ospedaliero, è quello di utilizzare un'applicazione dedicata che coinvolga sia il paziente, per la raccolta dei dati, sia il medico, per il controllo degli stessi. Tramite questa collaborazione e un po' di buon senso è possibile ottenere informazioni utili per la cura e un miglioramento nel comfort del paziente, il quale può essere monitorato tranquillamente da casa.

L'applicazione discussa in questo elaborato, per quanto basica, mostra un esempio per il monitoraggio continuo di alcuni parametri vitali. Come si è potuto notare, l'applicazione è stata adattata al contesto di utilizzo e fornita di tutti i requisiti necessari imposti dall'azienda. L'applicazione stessa, seppure progettata come componente da integrare in un sistema di monitoraggio più grande, può essere utilizzata come applicazione stand-alone e rappresenta una valida soluzione per

condividere di persona i propri dati clinici con il medico curante, con uno specialista selezionato, oppure con i propri familiari.

Il prodotto finale che si intende realizzare in futuro, comprensivo della componente illustrata e delle altre funzionalità introdotte nella sezione 2.2, è un sistema di monitoraggio completo per la cura dei pazienti affetti da DCA. Il sistema aiuterà i medici di competenza a gestire meglio i propri pazienti, permettendo una valutazione di lungo periodo e una segnalazione tempestiva di strategie di miglioramento.

Le attività di studio, progettazione e implementazione, per realizzare la componente illustrata, mi hanno permesso di capire il funzionamento del Bluetooth e in particolare l'utilizzo dei componenti Bluetooth Low Energy.

Ho appreso le fasi che portano al pairing dei dispositivi e alla sincronizzazione dei dati, sia a livello teorico che pratico. Le principali difficoltà che ho riscontrato sono state nella documentazione fornita, incompleta e poco dettagliata. All'inizio non mi era chiaro come stabilire una connessione con i dispositivi BLE forniti e come comunicare con essi per la lettura e scrittura delle Caratteristiche. È stato necessario un lavoro preliminare di autoapprendimento per capire le classi Android che permettessero di svolgere questi compiti e per capire appieno i concetti di Profilo, Servizio e Caratteristica.

3.6 Sviluppi futuri

Per prima cosa si vuole integrare la componente sviluppata, discussa in questo elaborato, nell'applicazione di monitoraggio generale implementata dagli altri membri del gruppo. Si vuole in questo modo modificare la procedura di acquisizione e visualizzazione dei dati del fitness tracker, adattandola a quella vista per gli altri dispositivi della componente. Relativamente alla componente, si vuole aggiungere, come nuovo dispositivo, un termometro digitale BLE (sempre della società A&D), la cui View è già presente nella schermata di set-up, ma che, per mancanza di

tempo, non è ancora stato gestito. Si vuole anche calcolare il BMI, disponendo dell'altezza del paziente.

Vengono inoltre riportate alcune idee, valutate dal nostro gruppo di sviluppo e dall'azienda stessa, per una futura estensione del progetto:

- Dividere il sistema di monitoraggio, composto finora di una sola applicazione con gestione multi-utente, in 2 parti:
  - Un portale web, utilizzato dal personale medico.
  - Un'applicazione, utilizzata dai pazienti.

  In questo modo i due utenti potranno svolgere le stesse funzionalità di prima, ma con applicativi dedicati e diversi. Questa feature vuole agevolare principalmente i medici, i quali trovano spesso più comodo monitorare i propri pazienti dal pc stesso piuttosto che da un tablet o uno smartphone.
- Fornire al medico la possibilità di inserire documenti (quali referti o risultati di esami di laboratorio), al fine di avere un quadro completo del soggetto da monitorare.
- Permettere ad entrambi gli utenti di visualizzare un grafico riepilogativo dell'andamento dei valori misurati, al fine di avere una panoramica generale sullo stato del percorso di cura e garantire una consultazione semplice e immediata.
- Fornire la possibilità di visualizzare le misurazioni relative ad una data specifica o un intervallo di date, selezionata/e tramite un apposito calendario.
- Aggiungere delle notifiche per ricordare al paziente di effettuare le misurazioni prestabilite e di caricare le foto dei pasti. La frequenza di invio di una notifica viene stabilita dal medico, in base al percorso di cura. A lato portale possono essere abilitate delle notifiche che segnalano al medico il superamento di un certo obiettivo da parte del paziente (numero di passi giornalieri, peso sotto una certa soglia, ecc.).

- Aumentare il numero dei dispositivi di rilevamento, forniti al paziente, per consentire al medico curante di monitorare più parametri vitali e avere una base di dati più ampia, in merito alla quale valutare l'andamento del percorso di cura.
- Estendere il contesto di utilizzo dell'applicazione a tutti gli ospedali per gestire al meglio alcune attività di routine.
- Creare un'architettura decentralizzata per la condivisione di informazioni sanitarie, robusta e interoperabile: Blockchain. La Blockchain permette di gestire con sicurezza i dati sanitari dei pazienti, garantendo protezione contro gli accessi non autorizzati o i tentativi di intrusione.
- Regolamentare l'applicazione con una certificazione medica di tipo A1, per tutelare i pazienti e garantire l'efficacia e la sicurezza dell'applicazione stessa come soluzione medica per la diagnosi e la cura.

Bibliografia

[1] Kevin Townsend, Carles Cufi, Akiba and Robert Davidson. *Getting Started with Bluetooth Low Energy*. O'Reilly Media, First Edition 2014

[2] Naresh Gupta. *Inside Bluetooth Low Energy*. Artech House, Second Edition 2016

[3] Redazione Grey-Panthers. *Il nuovo concetto di salute dell'OMS: cosa cambia nella cura del malato e nell'opinione medica*. 2017

[4] Pelle Svensson. *Bluetooth low energy possibilities in healthcare*. 2017

[5] Glenn Schatz. *IoT In Health Care: What You Should Know*. `https://www.link-labs.com/blog/iot-in-healthcare`

[6] *Bluetooth: tutte le informazioni importanti sul popolare standard radiofonico*. `https://www.ionos.it/digitalguide/server/know-how/bluetooth/`

[7] *Bluetooth: cos'è e come funziona*. `https://www.fastweb.it/smartphone-e-gadget/che-cos-e-il-bluetooth`

[8] *Bluetooth Basics*. `https://learn.sparkfun.com/tutorials/bluetooth-basics/all`

[9] Iliana Berenice Tejeda Hernandez. *Indication and Notification*. `https://community.nxp.com/docs/DOC-328525`, 2015

[10] *How does Bluetooth work?*.
https://www.elprocus.com/how-does-bluetooth-work/

[11] Bluetooth SIG. *Blood Pressure Profile*.
https://www.bluetooth.com/specifications/gatt/

[12] Bluetooth SIG. *Weight Scale Profile*.
https://www.bluetooth.com/specifications/gatt/

[13] Bluetooth SIG. *Blood Pressure Service*.
https://www.bluetooth.com/specifications/gatt/

[14] Bluetooth SIG. *Weight Scale Service*.
https://www.bluetooth.com/specifications/gatt/

[15] A&D Medical. *Application Development Specification for A&D Bluetooth BLE Series: Blood Pressure Monitor UA-651BLE*

[16] A&D Medical. *Application Development Specification for A&D Bluetooth BLE Series: Weight Scale UC-352BLE*

[17] Microchip Developer. *Connecting to BLE Devices*.
https://microchipdeveloper.com/wireless:ble-connect

[18] Android Developers Documentation. *Bluetooth low energy overview*. https://developer.android.com/guide/topics/connectivity/bluetooth-le

Ringraziamenti

In primo luogo vorrei ringraziare il Prof. Davide Rossi per la disponibilità data e per avermi seguito durante la stesura di questo elaborato.

Inoltre, vorrei ringraziare i miei genitori e mia sorella per avermi sempre sostenuto nei momenti di difficoltà e per avermi dato l'opportunità di intraprendere questo corso di studi.

Un ultimo ringraziamento va a tutto il team di sviluppo, incontrato presso l'azienda Finmatica, insieme al quale ho lavorato per la realizzazione del progetto. Grazie anche a Fabrizio per avermi dato l'opportunità di usufruire degli strumenti di sviluppo per alcuni test post-tirocinio, essenziali al completamento della tesi.