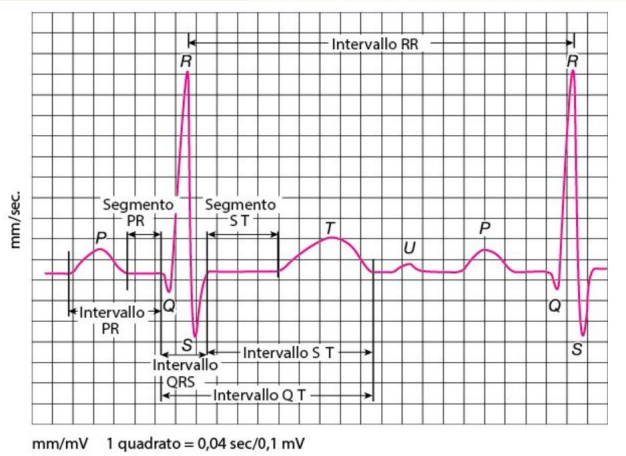
Sviluppo di un'applicazione per dispositivo mobile per la rilevazione della HRV

Pintarelli Debora - VR474808
Relatore: Pravadelli Graziano

HEART RATE VARIABILITY (HRV)

- É una misura che indica la variabilità dell'intervallo battito-battito cardiaco (intervallo RR)
- Riflette l'attività del sistema nervoso autonomo e indica la capacità del corpo di adattarsi a stress fisici e mentali
- HRV elevata è generalmente associata ad un buon stato di salute e maggiore resilienza
- HRV bassa può segnalare stress o affaticamento



PROGETTO

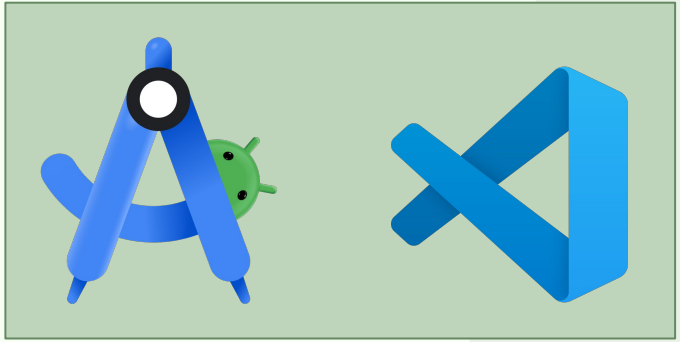
Obiettivo: Implementare un'applicazione mobile per poter rilevare, tramite uno smartwatch, i battiti cardiaci per poi calcolare e monitorare il valore della HRV

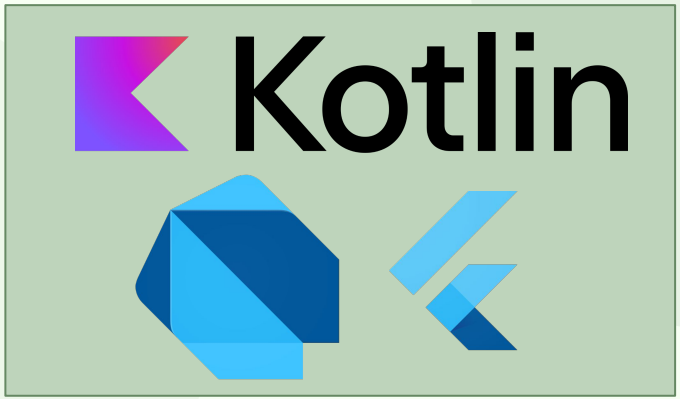
Smartwatch TicWatch E2

- **Sistema operativo:** Wear OS di Google
- **Connettività:** Bluetooth v4.1
- **Sensore utilizzato:** sensore della frequenza cardiaca
- **RAM:** 0,5 GB
- **Memoria interna:** 4 GB

PROGETTO

- **Strumenti di sviluppo**
  - Android Studio
  - Visual Studio Code
- **Linguaggi di programmazione**
  - Kotlin
  - Dart (Flutter)
- SQLite





API Data Layer di Wear OS

Consente la comunicazione tra dispositivi wearable e dispositivi mobile connessi (come gli smartphone).

- **Sincronizzazione dei dati:** permette di sincronizzare e trasferire dati tra dispositivi Wear OS e dispositivi Android associati
- **Modalità di comunicazione:** i dati possono essere trasferiti direttamente tramite Bluetooth o attraverso una rete disponibile (come Wi-Fi)
- **Sicurezza:** la comunicazione è crittografata end-to-end, garantendo che solo le app con lo stesso nome di pacchetto e firma possano accedere ai dati

COMUNICAZIONE TRA IL DISPOSITIVO MOBILE E L'OROLOGIO

1. Il dispositivo mobile invia un messaggio getValori() all'orologio per notificare l'inizio della procedura di trasferimento dei dati
2. L'orologio invia i dati al dispositivo mobile, che li riceve e li salva
3. Dopo aver salvato i dati, il dispositivo mobile invia una richiesta all'orologio per eliminare i dati trasferiti

MESSAGGI

I messaggi che vengono inviati dalle due applicazioni sono in formato JSON.

```json
{
    "canale": "getValori",
    "json": [
        {
            "valore": 98.4,
            "momento": "2024-11-30T16:12:48"
        },
        {
            "valore": 95.4,
            "momento": "2024-11-30T16:12:58"
        }
    ]
}
```

- “**canale**”: viene utilizzato per capire che funzione dovrà lanciare il destinatario (es: getValori)
- “**json**”: insieme di tutti i valori salvati nell’orologio

GESTIONE DEI DATI

- I dati una volta raccolti vengono mantenuti nell'orologio (solitamente ogni 10 min)
- Quando l'app dello smartphone si apre vengono inviati i dati, salvati nell'orologio, all'interno del cellulare e successivamente vengono eliminati dall'orologio

SQLite

Database relazionale leggero e integrato

- **Configurazione:** non richiede un server separato o un processo di installazione complesso, è già tutto gestito all'interno dell'app attraverso la sua libreria.
- **Autonomo:** tutto il database è contenuto in un singolo file sul disco
- **Transazionale:** supporta transazioni ACID (Atomicità, Coerenza, Isolamento, Durabilità)
- **Leggero:** utilizza una quantità molto ridotta di memoria, ideale per dispositivi con risorse limitate

GESTIONE SMARTWATCH

Status permessi: Non ok
Status servizio: Non ok

Attiva servizio

Aggiorna

Status permessi: Ok
Status servizio: Non ok

Attiva servizio

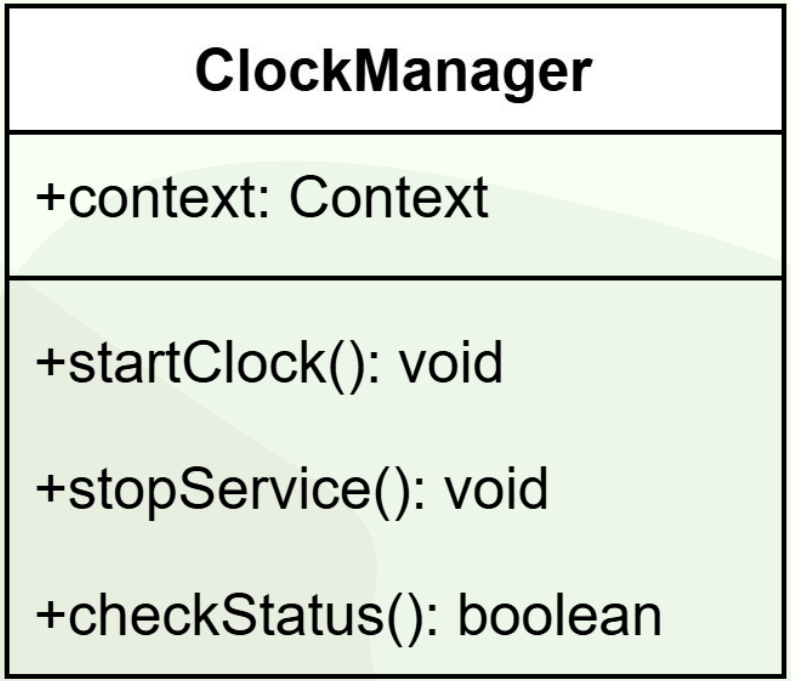
Aggiorna

Status permessi: Ok
Status servizio: Ok

Disattiva servizio

Aggiorna

ClockManager



Questa è la classe principale per gestire l'applicazione sullo smartwatch.

- **startClock():** configura un servizio di sistema che consente allo smartwatch di leggere la HR ogni minuto e avvia il servizio che permette la comunicazione tra l'orologio e il dispositivo mobile
- **stopService():** chiude tutti i servizi attivi nell'applicazione
- **checkStatus():** ritorna true se i servizi sono attivi, altrimenti ritorna false

SERVIZI IMPLEMENTATI

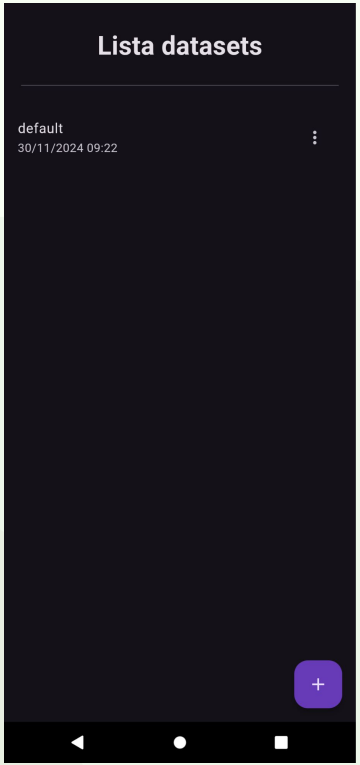
- **MobileService:** gestisce la comunicazione con il dispositivo mobile. Il suo compito è di ricevere le richieste e, in base al canale scelto, avviare la funzione corrispondente e successivamente rispondere al mittente
- **SensorService:** ha il compito di mettersi in ascolto e monitorare la HR. Una volta letto il valore, il servizio si chiude.

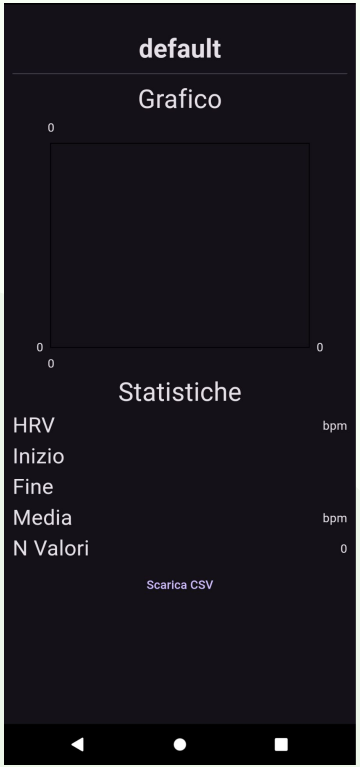
DatiReceiver

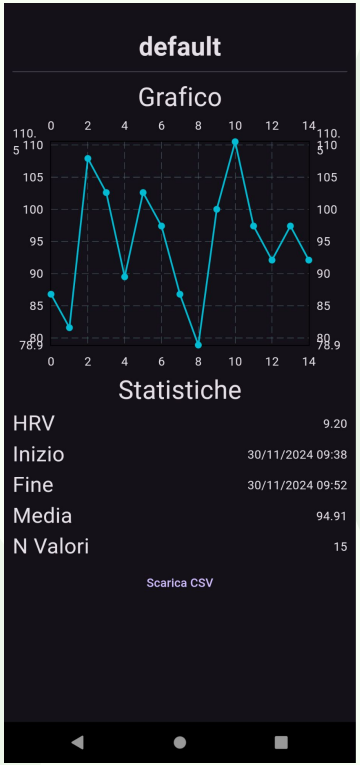
Questo oggetto viene invocato dal sistema operativo ogni minuto e ha due compiti:

- Avviare il servizio per leggere l'HR
- Controllare lo stato del servizio che permette la connessione tra il dispositivo mobile e l'orologio. Se il servizio è stato terminato dal SO, questo oggetto lo riavvia

GESTIONE MOBILE

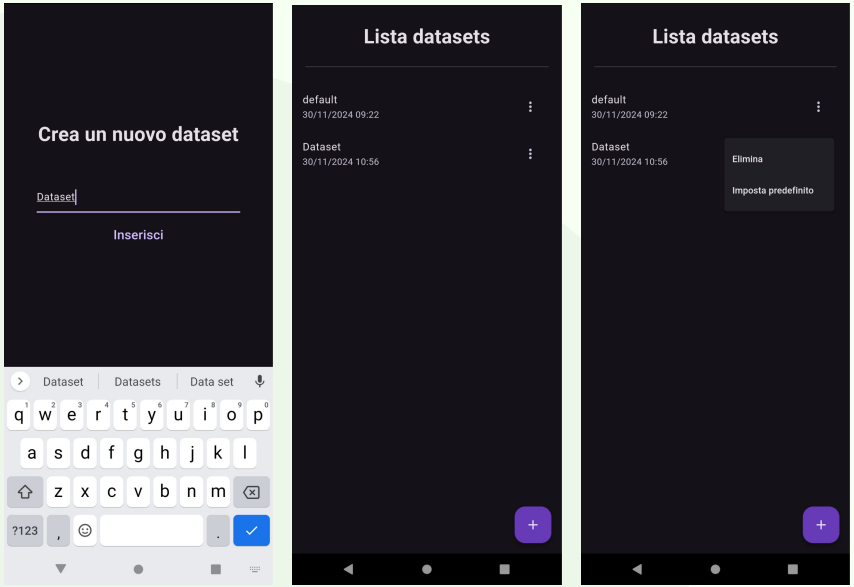






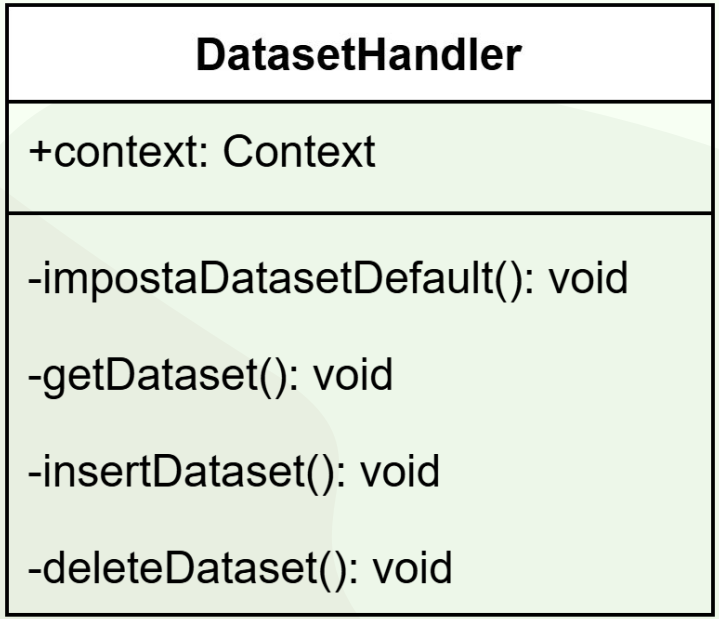
HANDLERS

- Sono stati implementati degli handler per gestire separatamente ogni comando dell'applicazione.
- Quando l'utente preme un pulsante che richiede l'uso di una funzione nel backend, Flutter invia una notifica e poi nel main viene avviato l'handler corretto in base alla notifica ricevuta.



DatasetHandler

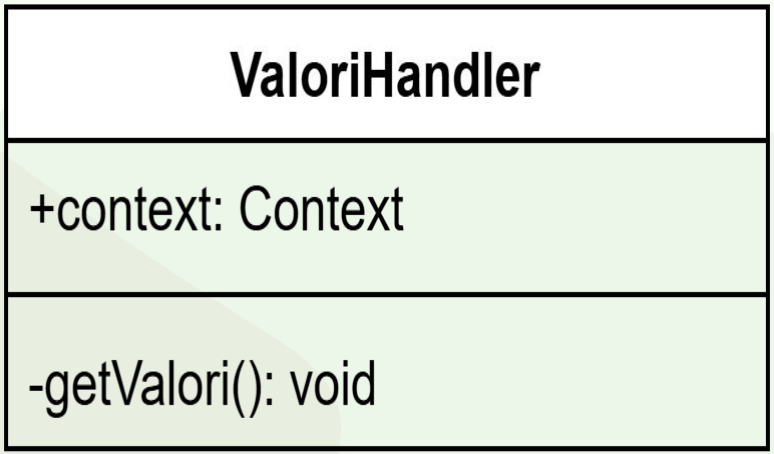
Questo handler permette all'applicazione in Flutter di gestire i dataset all'interno del dispositivo mobile.



- **impostaDatasetDefault():** imposta il dataset dove i dati verranno salvati.
- **getDataset():** esegue una query per ottenere tutti i dataset dal database e li invia all'app in Flutter
- **insertDataset():** crea un nuovo dataset
- **deleteDataset():** elimina un dataset

ValoriHandler

Questo handler permette all'applicazione in Flutter di avviare il trasferimento dei valori dallo smartwatch al dispositivo mobile.



getValori(): fa iniziare la fase di trasferimento dei dati dallo smartwatch al dispositivo mobile

DataLayerRepository

- Questa repository gestisce la comunicazione tra il dispositivo mobile e lo smartwatch tramite l'API Data Layer di Wear OS.
- Permette di inviare e ricevere messaggi dallo/allo smartwatch.

DatabaseRepository

App dello smartwatch

- Repository utilizzata per gestire il database SQLite e quindi per salvare i dati rilevati (HR).

App del dispositivo mobile

- Repository utilizzata per gestire il database SQLite e quindi per salvare i dataset e i dati rilevati dallo smartwatch per ogni dataset.

PLUGIN UTILIZZATI

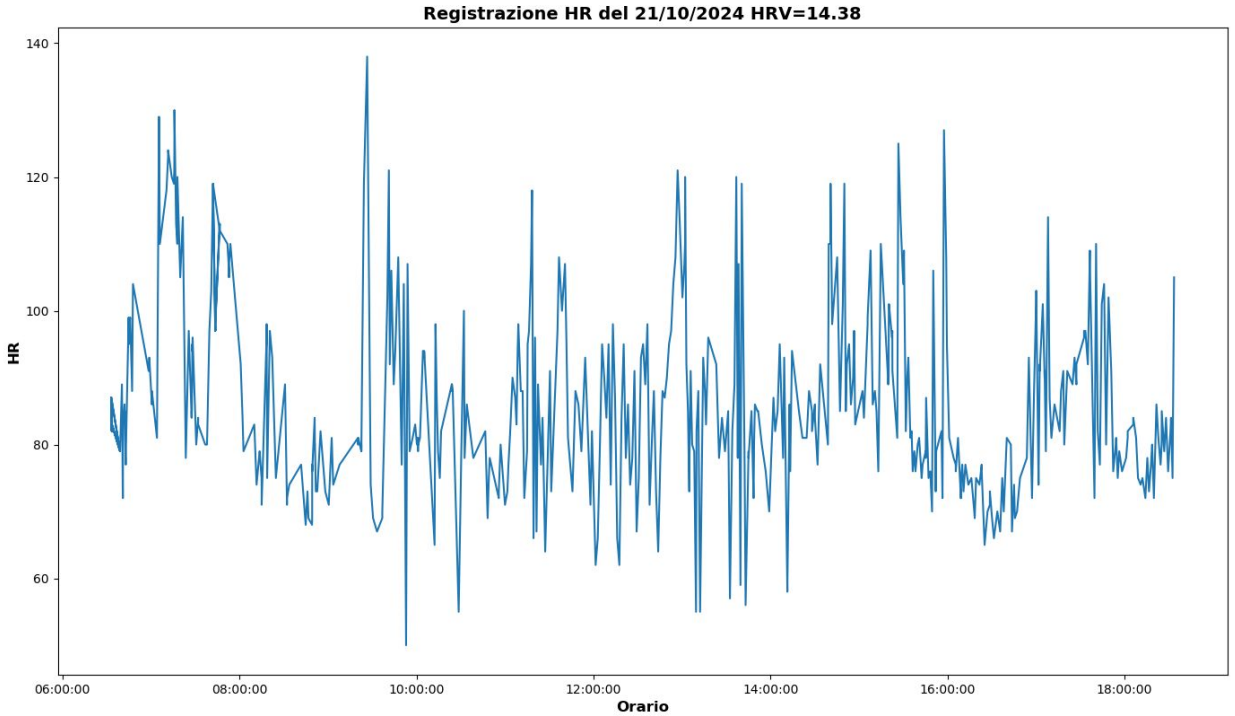
App dello smartwatch

- È stato utilizzato permission_handler_android per gestire in modo efficiente i vari permessi dell'app.

App del dispositivo mobile

- È stato utilizzato file_selector_android per poter salvare i dati su un file.

FASE DI TESTING



Inizio: 8:33
Batteria iniziale: 100%
Fine: 20:30
Batteria finale: 10%
Durata test: 12h