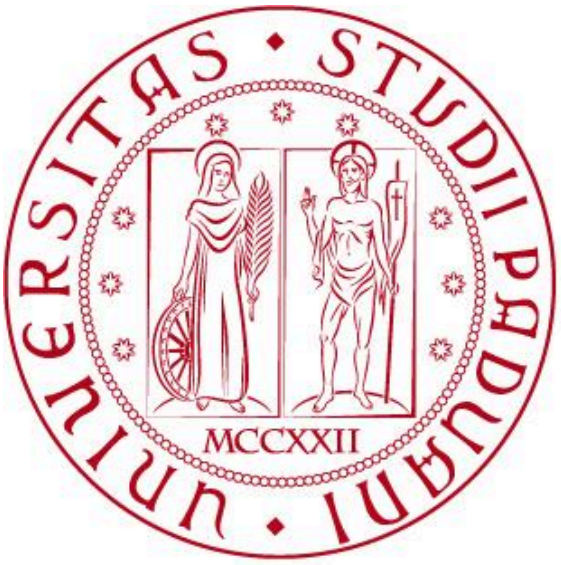


UNIVERSITÀ DEGLI STUDI DI PADOVA

Progetto di Programmazione ad Oggetti

Anno accademico: 2023/2024

Studente: Annalisa Egidi

Matricola: 1216745

Health Sensors

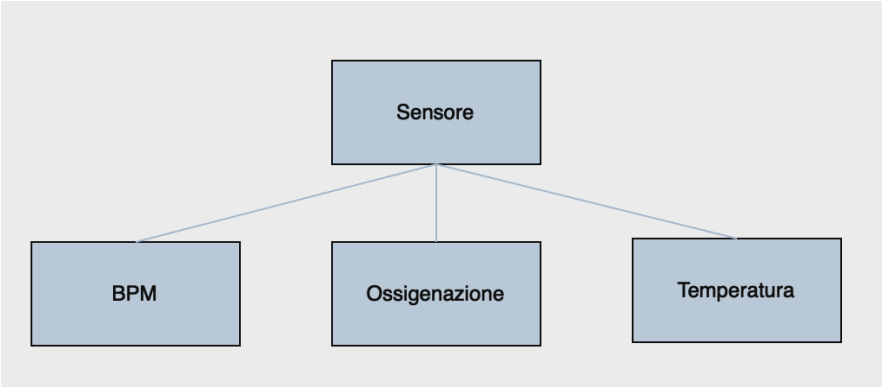
Introduzione

Il progetto sviluppato è un programma relativo alla gestione di sensori inerenti al monitoraggio biometrico destinato all'analisi e gestione della salute.
I sensori disponibili sono relativi alla misurazione della frequenza cardiaca, dell'ossigenazione nel sangue e della temperatura corporea.
Il programma permette inoltre l'aggiunta e rimozione dei vari tipi di sensori, nonché la possibilità di modificare il nome di un dato sensore e la simulazione di un campionamento di dati effettuato nell'arco di 24 ore.

Gerarchie di tipi

E' stato utilizzato il pattern architetturale "**Model-View-Presenter**" (derivazione di "Model-View-Controller") in modo da garantire massima separazione tra vista e modello, delegando al controller l'onere di traduzione dei dati tra la vista ed il modello.

Il progetto prevede la seguente gerarchia relativa alla modellazione del dominio di interesse:



La classe "**Sensore**" è la classe base della gerarchia: essa gestisce le informazioni comuni a tutti i sensori, ovvero il nome, l'identificativo, le misurazioni che sono state effettuate, i valori dell'intervallo considerabile come "accettabile", la frequenza delle misurazioni.

La classe "**Ossigenazione**" è una classe concreta che consente di gestire tutte le operazioni riguardanti un sensore relativo all'ossigenazione sanguigna: a differenza della classe base, contiene un attributo di nome *alertMin* che indica il valore limite (inferiore) sotto al quale la misurazione viene considerata come una misurazione in un soggetto malato.

Il sensore relativo all'ossigenazione nel sangue contiene un metodo per avvisare del numero di misurazioni che hanno registrato dati al di sotto della soglia minima di salubrità.

La classe “**BPM**” modella un sensore relativo alle pulsazioni cardiache (alla loro frequenza); rispetto alla classe base, contiene due attributi in più:

- *sportMin*: indica il valore minimo per l’intervallo di frequenza cardiaca adibito a “intervallo sport”;
- *sportMax*: indica il valore massimo per l’intervallo precedentemente discusso.

La classe implementa inoltre un metodo relativo al calcolo dei battiti cardiaci medi per l’attività sportiva.

La classe “**Temperatura**” è relativa a un sensore di misurazione della temperatura corporea, consente di individuare la presenza di febbre nel soggetto registrato nel caso in cui un certo numero di misurazioni siano superiori a 100 gradi Fahrenheit.

Funzionalità

Le funzionalità offerte dall’applicazione sono le seguenti:

- **Gestione** delle **operazioni** CRUD (create - read - update - delete) sui sensori;
- **Visualizzazione** dei dati immessi per mezzo di un grafico (widget centrale) e due barre laterali
    - Barra dei sensori: widget che si occupa di gestire le operazioni CRUD sui sensori e la ricerca degli stessi;
    - Barra dei dettagli: si occupa di visualizzare i dati (ex. nome, id, massimi e minimi) del sensore, oltre a eventuali campi dati specifici del tipo di sensore.
- **Visualizzazione** delle statistiche calcolate sulle misurazioni campionate in una giornata intera;
- **Simulazione** delle misurazioni avvenute in un intervallo di tempo di 24 ore per un singolo sensore;
- **Simulazione** delle misurazioni avvenute in un intervallo di tempo di 24 ore per tutti i sensori registrati;
- **Filtro/ricerca** di sensori in base al loro nome e identificativo;
- **Salvataggio** dei dati in formato .txt: ogni piano di allenamento può essere salvato tramite la voce “**Salva**” nel menù “File”;
- **Ripristino** dei dati da file: se il file .txt utilizzato è ben formattato, avviene il ripristino dei sensori in esso salvati.

Uso di polimorfismo

Nella gerarchia avente “Sensore” come classe base sono presenti i seguenti metodi virtuali puri:

- `virtual void daySimulazione() const = 0;`
    - Si occupa della simulazione delle misurazioni in un intervallo di 24 ore.

- *virtual* Sensore **clone*() *const* = 0;
    - Metodo di utilità che implementa il costruttore di copia profondo.
- *virtual* ~*Sensore*() = *default*;
    - Distruttore profondo.
- *virtual* int *dayLifeQuality*() *const* = 0;
    - Metodo che calcola la qualità della giornata in base ai dati registrati.

Formati di input/output

Come già riportato nella sezione “Funzionalità”, il programma consente di salvare i dati in file .txt; ogni file riporta una riga per ogni sensore, avente le seguenti caratteristiche ordinate e separate da una virgola:

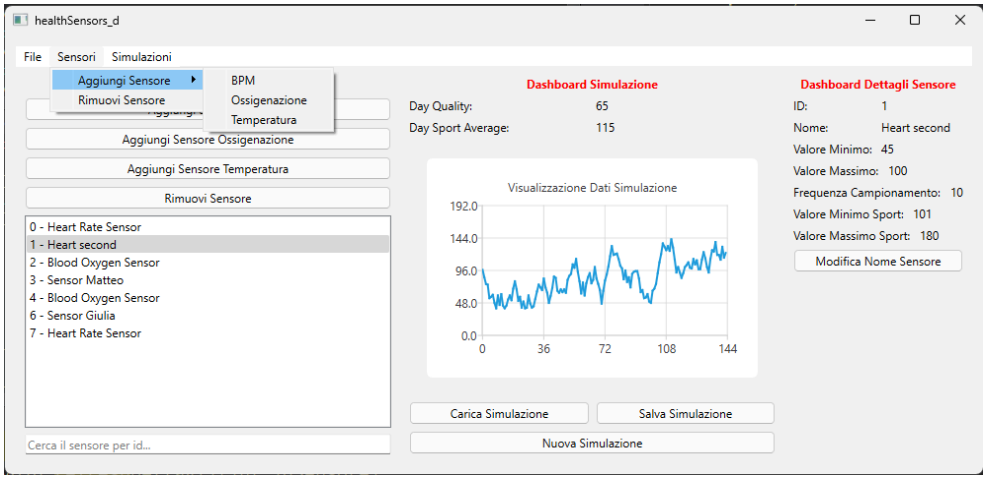
- *Tipo sensore*: può avere come valore “BPM”, “Temperatura” o “Ossigenazione”;
- *ID*: identificativo del sensore;
- *Valid_min*: valore minimo per la salubrità del soggetto in relazione al tipo di misurazione;
- *Valid_max*: valore massimo per la salubrità del soggetto in relazione al tipo di misurazione;
- *Frequenza misurazione*: numero di minuti che intercorrono tra un campionamento e un altro;
- *Nome del sensore*;
- Se il sensore è di tipo BPM*, sport min e max*;
- Se il sensore è di tipo Ossigenazione, *alertMin*;
- Le misurazioni: sono degli interi separati da virgole.

Manuale d’uso GUI

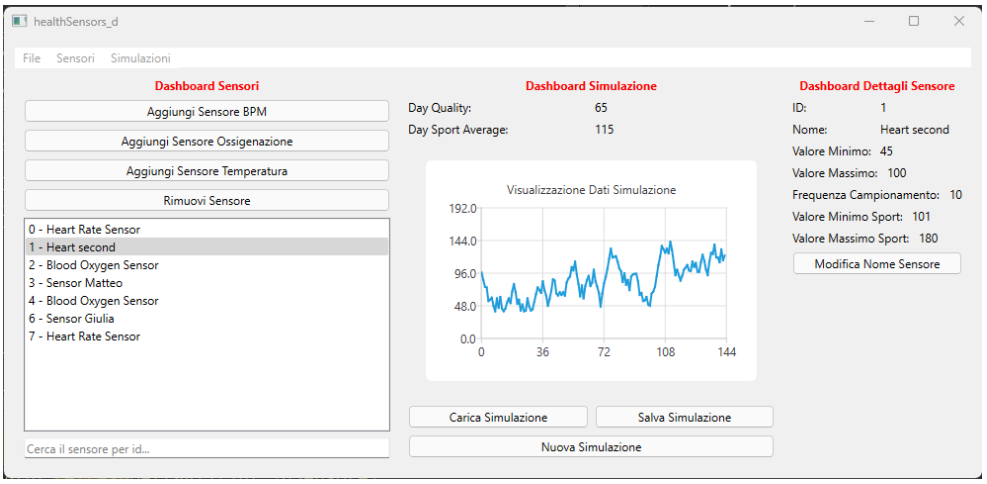
La **schermata principale** è costituita da:

Barra dei menù: sono presenti tre menù:

- **File**: consente di effettuare operazioni su file (salvataggio, ripristino);
- **Sensori**: consente di aggiungere e rimuovere sensori tramite appositi dialog di sistema;
- **Simulazioni**: consente di operare sulle simulazioni per il singolo sensore / tutti i sensori salvati.



- **Dashboard Sensori**:
    - Pulsanti dai quali è possibile aprire dialog per operare sui sensori, oltre che da una barra di ricerca per il filtraggio degli stessi;
    - Lista dei sensori attivi durante l'esecuzione, completi di nome e identificativo;
    - Barra di ricerca dei sensori, esegue un filtraggio in base alla stringa immessa ad ogni carattere digitato.



- **Dashboard Simulazione**: widget principale, contiene dettagli sulla qualità della giornata (simulata), un grafico con l'andamento delle misurazioni e i pulsanti di gestione delle simulazioni (caricamento, salvataggio e creazione simulazione per il sensore attivo);
- **Dashboard Dettagli Sensore**: è costituita da una serie di righe contenenti le informazioni specifiche del singolo sensore (nome, identificativo, valori limite per l'intervallo di salubrità).

Istruzioni compilazione ed esecuzione

Per una corretta configurazione, occorre installare i seguenti pacchetti:

- **qt6-default**
- **libqt6charts6-dev**

Nella cartella di consegna è fornito il file "**healthSensors_d.pro**" sul quale richiamare i comandi:
qmake healthSensors_d.pro → *make healthSensors_d* → *./healthSensors_d*

Ore di lavoro richieste

Analisi preliminare:	1 ora
Progettazione modello:	3 ore
Progettazione GUI:	2 ore
Apprendimento libreria Qt:	10 ore
Codifica modello:	16 ore

Codifica GUI:	19 ore
Debugging:	10 ore
Testing:	2 ore
Totale:	**63 ore**

Il totale delle ore di lavoro richieste supera le 50 ore in quanto sono stati riscontrati problemi relativamente alla codifica dell'interfaccia grafica e il collegamento di determinati segnali (ex. il segnale relativo alla ricerca/filtraggio dei sensori).

Ambiente di sviluppo

Sistema Operativo:	Windows 11
Libreria Qt:	Qt 6.2.4
Compilatore:	g++ 11