Università Politecnica delle Marche

Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Sviluppo di un'applicazione per il monitoraggio di elementi in calcestruzzo tramite sistema multi-sensore

Development of an application to remotely monitor concrete elements through multi-sensor system

Relatore:
Prof. Alessandro Cucchiarelli

Candidato:
Giovanni Giacometti

Correlatore:
Prof. Gian Marco Revel

Anno accademico 2020/2021

Sommario

CAPITOLO 1

Introduzione

Il presente elaborato di tesi illustra il processo di sviluppo di un sistema di monitoraggio dell'impedenza elettrica rilevata su blocchi di calcestruzzo, realizzati con materiali innovativi nell'ambito del progetto europeo EnDurCrete.

Uno dei criteri con cui valutare le prestazioni e l'affidabilità di un materiale cementizio è infatti l'analisi delle sue proprietà elettriche, che risultano dipendenti dalle forze esercitate sul materiale stesso e strettamente legate a fenomeni interni come la corrosione. Lo scopo del lavoro descritto nella tesi è quello di progettare un sistema in grado di monitorare, nel corso del tempo, l'andamento dell'impedenza elettrica sui blocchi di calcestruzzo, attraverso l'esecuzione automatica e continuativa delle misure, il salvataggio dei dati e la realizzazione di un'interfaccia tramite cui visualizzare i risultati.

La prima fase dello sviluppo si è concentrata sulla definizione puntuale dei requisiti, funzionali e non funzionali, che caratterizzano il sistema di monitoraggio; poi, dopo aver elaborato la struttura dell'architettura complessiva e aver individuato gli strumenti tecnologici necessari a sviluppare le varie componenti, si è passati alla realizzazione concreta dei software che

implementa le funzionalità richieste. Quest'ultimo passo è quello che costituisce il fulcro dell'attività presentata in questo elaborato.

L'architettura si compone di tre applicativi, ognuno dei quali deputato ad una particolare funzionalità.

Il primo software è quello che si occupa di interagire con i sensori mediante le schede di acquisizione corrispondenti; consiste in uno script, realizzato con il linguaggio di programmazione Python, che risiede sulle macchine situate presso i siti di monitoraggio e che viene mandato in esecuzione una volta ogni ora, per un totale di ventiquattro volte al giorno. Al suo interno sono contenute le istruzioni necessarie ad effettuare le misure, che avvengono comunicando serialmente con le schede di acquisizione, e ad inviare i valori rilevati, dopo un opportuno processamento, ai database; in particolare, i dati vengono immagazzinati in due DBMS: Sqlite, risiedente sulla macchina stessa, e PostgreSQL, localizzato invece su un server remoto.

L'interfaccia del sistema di monitoraggio è realizzata mediante un'applicazione mobile, sviluppata tramite il framework Flutter e il linguaggio di programmazione Dart; caratterizzata da un design elegante e minimale, l'applicazione offre agli utenti due funzionalità: la prima è quella di visualizzare un grafico relativo alle grandezze oggetto di monitoraggio, la seconda è quella di richiedere l'invio alla propria e-mail di un file contenente i dati estratti dal database, nell'eventualità in cui si necessiti di una loro analisi a parte e senza una rappresentazione grafica interna all'applicazione. In entrambi i casi è possibile personalizzare i parametri che definiscono i dati richiesti, specificando, ad esempio, la scheda di acquisizione o il periodo temporale di cui si vogliono visionare gli andamenti.

I dati necessari ad espletare le funzionalità presenti nell'applicazione vengono ottenuti mediante delle richieste HTTP dirette al server del sistema di monitoraggio, che costituisce il terzo e ultimo software di cui è composta l'architettura. Consiste, analogamente al software che effettua le misure, in uno script Python, il quale implementa un servizio di REST API sfruttando le classi e i metodi offerti della libreria Flask e della sua estensione flask-restful.

Le funzionalità associate agli endpoint esposti dal server consentono l'accesso ai dati immagazzinati nel database remoto; in particolare, i parametri associati ad ogni richiesta, ovvero quelli selezionati dall'utente all'interno dell'applicazione mobile, definiscono la query che va effettuata in modo da estrarre l'insieme di valori desiderato. I dati così ottenuti vengono elaborati, in modo che l'applicazione possa predisporli alla visualizzazione grafica senza bisogno di ulteriore processing, e infine inviati sotto forma di strutture dati JSON.

Lo scopo del presente elaborato di tesi è dunque quello di descrivere il processo di realizzazione del sistema di monitoraggio, analizzando le fasi che hanno preceduto lo sviluppo vero e proprio e dettagliando le soluzioni tecnologiche che caratterizzano i software che lo compongono.

CAPITOLO 2

Contesto Applicativo

EnDurCrete, acronimo di *Environmental friendly and Durable conCrete*, è un progetto europeo che si pone come obiettivo quello di sviluppare nuove tecnologie durature e sostenibili ambientalmente per la realizzazione del calcestruzzo. Allo stato attuale il cemento Portland è l'ingrediente base del calcestruzzo, sia in ambito infrastrutturale che edile, ma la sua produzione comporta un consumo elevato di risorse minerali, come calcare e argilla, e origina emissione massiccia di gas serra, considerati tra le principali cause alla base del problematico fenomeno del riscaldamento globale. Numerosi sono stati i tentativi di rendere l'utilizzo del calcestruzzo più sostenibile, ma tutte le soluzioni proposte nel corso del tempo si sono rivelate poco affidabili, specialmente in contesti di strutture più complesse. EnDurCrete si colloca in questo scenario, con l'intento di ideare un calcestruzzo che sia poco impattante da un punto di vista ambientale ma al contempo in grado di fornire performance elevate [1].

L'Università Politecnica delle Marche, in particolare il DIISM, Dipartimento di Ingegneria Industriale e delle Scienze Matematiche, è un partner del progetto EnDurCrete; uno dei suoi ambiti di ricerca riguarda lo studio e lo sviluppo di elementi in calcestruzzo realizzati con materiali innovativi, come dei riempitivi a base di argille nano-modificate o di micro-carbonio, e dotati

inoltre di rivestimenti multifunzionali che gli conferiscono proprietà autorigeneranti e lo proteggono da agenti esterni aggressivi. Una delle metodologie con cui le proprietà di tali materiali vengono esaminate è lo studio delle loro caratteristiche elettriche, in particolare dell'impedenza.

L'impedenza elettrica è una grandezza fisica definita come la forza che un circuito esprime per opporsi al passaggio di una corrente elettrica variabile. È un numero complesso, esprimibile quindi attraverso l'espressione $x + iy$, dove x e y sono numeri reali ed i è invece l'unità immaginaria, o in forma esponenziale, ovvero come $r * x^{i\Theta}$, dove r e Θ sono detti rispettivamente modulo e fase.

Lo studio delle proprietà elettriche del calcestruzzo, in quanto legate a fattori come la densità, il rapporto acqua/cemento e la composizione dei materiali con cui viene realizzato, consente di valutarne i dettagli microstrutturali e di stimarne il potenziale di corrosione, un fenomeno anch'esso legato a variazioni di natura elettrochimica; in ultima istanza, anche le deformazioni meccaniche sono rintracciabili attraverso l'analisi delle caratteristiche elettriche, in quanto inducono un cambiamento percettibile nella conduttività del materiale.

Il lavoro illustrato in questa tesi, svolto nell'ambito del mio tirocinio presso il DIISM, è consistito nella realizzazione di un sistema di monitoraggio dell'impedenza elettrica di elementi in calcestruzzo, i quali sono stati realizzati dai ricercatori del dipartimento con i materiali e le tecnologie innovative precedentemente descritti. L'obiettivo del progetto è quello di poterne visualizzare l'andamento nel tempo e al variare della temperatura, in modo da ricavare dati e informazioni sulla qualità e sulla durabilità dei particolari materiali utilizzati.

Il mio tirocinio, in particolare, si è concentrato sullo sviluppo dei software per poter ottenere, visualizzare e usufruire dei dati rilevati dai sensori di monitoraggio.

L'architettura informatica deve ovviamente interagire con gli strumenti hardware, che costituiscono il primo livello del sistema di monitoraggio. Nella prossima sezione verrà quindi analizzato lo strato fisico, quello composto dai sensori e dalle schede di acquisizione, cercando di evidenziarne le peculiarità da cui è partito lo sviluppo dei vari software che compongono il sistema.

2.1 Sistema di monitoraggio

Il sistema di monitoraggio si articola in due installazioni, situate in luoghi diversi ma funzionalmente identiche fra loro.

La prima è collocata in Spagna, nella città di Leon, presso la *Fundación Santa Bárbara,* un istituto di ricerca spagnolo anch'esso partner del progetto EnDurCrete. I blocchi di calcestruzzo

monitorati in questo sito si trovano all'interno di un tunnel, isolato termicamente rispetto all'esterno. Anche la seconda installazione si trova in Spagna, ma è situata a Gijon, precisamente in un molo del suo porto; in una zona di esposizione di questo sito i blocchi di calcestruzzo si trovano a contatto con l'acqua e ciò ne permetterà quindi lo studio nell'ambito di applicazioni marine.

2.1.1 Sensori

In figura 1 è riportato lo schema dei sensori installati sui blocchi di calcestruzzo da monitorare.

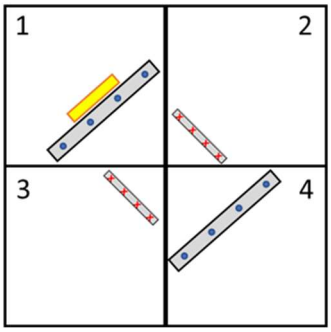

Figura 1 – Schema dei sensori

Ogni elemento, di dimensioni 35x35x20 cm, presenta 4 array, ognuno contenente 4 elettrodi, di cui due, ovvero quelli situati nei quadrati 1 e 4, sono caratterizzati da una spaziatura inter-elettrodo di 4 cm e si trovano rivolti verso l'alto, mentre gli altri due hanno una spaziatura di 1 cm e sono rivolti verso il basso. Tutti sono collocati ad una profondità di 5 cm dalla base del blocco di calcestruzzo. I 4 elettrodi posti in ogni schiera hanno le seguenti funzioni: quelli esterni sono collegati direttamente al generatore di corrente alternata e sono dunque utilizzati per l'eccitazione, quelli interni rilevano invece il potenziale elettrico generatosi e inviano tale informazione alle schede di acquisizione. Ogni array è collegato attraverso un cavo Ethernet ad un connettore RJ45 femmina, il quale, al momento della gettata, è stato fissato sulla faccia del blocco di calcestruzzo; in questo modo è stato possibile collegare il sistema di acquisizione (attraverso un RJ45 maschio) direttamente presso il sito di monitoraggio, evitando di trasportare i campioni di calcestruzzo con cavi esterni facilmente danneggiabili. La procedura del trasporto dei blocchi è schematizzata in figura 2.

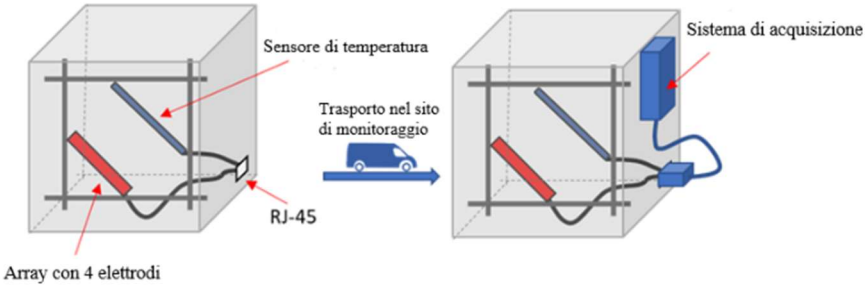

Figura 2 - Trasporto dei campioni di calcestruzzo

I connettori sono di tipo IP68; il codice IP (*International Protection*) è un codice internazionale che classifica il grado di protezione di un involucro meccanico ed elettrico. Il valore 68, stando alla definizione ufficiale, identifica una protezione "da immersione permanente" per quanto riguarda l'accesso di liquidi e "a tenuta di polvere" relativamente alle particelle solide [2]; garantisce quindi una copertura elevata dell'apparato sensoristico installato all'interno dei blocchi, una caratteristica fondamentale per evitare infiltrazioni che possano danneggiare le componenti, ma particolarmente rilevante nell'installazione costantemente esposta al contatto con l'acqua del mare nel sito di Gijon.

La misurazione dell'impedenza elettrica avviene mediante il metodo Wenner, che si basa sulla configurazione dei quattro elettrodi precedentemente presentata; viene inoltre sfruttata la tecnica di misura EIS (Electrochemical Impedance Spectroscopy), la quale consente di ricavare l'impedenza elettrica di un materiale in funzione della frequenza della corrente alternata erogata sugli elettrodi situati all'interno del materiale stesso.

Ogni blocco, inoltre, è dotato di un ulteriore sensore, in grado di misurare la temperatura rilevata internamente allo stesso.

2.1.2 Schede di acquisizione

Le schede di acquisizione costituiscono l'ultimo strato hardware del sistema di monitoraggio, ovvero l'interfaccia con cui l'architettura software deve interagire per effettuare le misure. Sono dei dispositivi elettronici programmabili, in grado quindi di trasformare un input digitale in un output analogico da inviare ai sensori e, successivamente, di trasferire indietro il valore della misura effettuata.

Il sistema si serve di due schede di acquisizione: un'evaluation board, denominata AD5940BIOZ, prodotta dall'azienda Analog Devices e utilizzata per effettuare le misure di impedenza elettrica, e un Arduino Nano, che si occupa invece di leggere il valore di temperatura rilevato dal sensore apposito. Ogni array di elettrodi ha una propria scheda con cui

interagisce e comunica; di conseguenza, ad ogni blocco di calcestruzzo corrispondono quattro board AD5940BIOZ e un Arduino, che risulta infatti specifico per ogni sensore di temperatura. Come visibile in figura 3, le schede di acquisizione sono state poste all'interno di piccole scatole elettriche, le quali sono poi state riempite di un gel protettivo, anch'esso IP68, con l'obiettivo di tutelarne il funzionamento anche nelle zone più sfavorevoli, come l'installazione marina.

Figura 3 - Scatole elettriche con gel IP68

2.1.2.1 AD5940BIOZ

La scheda AD5940BIOZ è realizzata dall'azienda statunitense Analog Devices, uno dei leader mondiali nell'ambito della produzione di semiconduttori; è una evaluation board, un termine indicato per descrivere una scheda elettronica sulla quale viene installato un determinato dispositivo, in questo caso il rilevatore di impedenza, con annessi un certo numero di componenti accessori e dotato inoltre di ingressi e uscite facilmente accessibili.

Come suggerito dal nome, il modello è stato progettato per effettuare misure di impedenza elettrica in ambito medico; esempi di campi di impiego sono la BIA, acronimo di Body Impedence Assessment, una tecnica di valutazione della composizione corporea basata sulla misurazione dell'impedenza, e l'analisi dell'EDA, ovvero l'electrodermal activity, una proprietà del corpo umano per cui le caratteristiche elettriche della pelle variano secondo alcune condizioni. Il codice sorgente del firmware della scheda è tuttavia disponibile in modo open source sul repository GitHub dell'azienda ed i ricercatori del dipartimento hanno apportato delle modifiche per adattarlo anche alla misurazione di impedenza di materiali cementizi come quelli coinvolti nel monitoraggio.

Le schede sono collegate serialmente alle macchine localizzate presso i siti. Per controllarle, ovvero per attivare le misure e visualizzarne poi i risultati, è necessario installare SensorPal,

un software prodotto dalla stessa Analog Devices, il quale fornisce un'interfaccia grafica per effettuare le misurazioni e analizzarne i valori attraverso dei grafici; le funzionalità offerte da questo software verranno approfondite successivamente nella sezione 2.2.
SensorPal fornisce inoltre delle API, implementate nel file assembly SensorPal.API.dll, presente nella cartella in cui è contenuto l'eseguibile al momento dell'installazione; le API consentono di integrare l'esecuzione delle funzioni presenti nell'interfaccia grafica del software all'interno di altri programmi, in modo da garantire flessibilità agli sviluppatori che vogliono usufruire delle schede in prodotti specifici.
Le API sono realizzate in C#, per cui sono necessarie delle librerie che permettono l'integrazione del proprio software con tale linguaggio di programmazione. Inoltre, essendo un file con estensione dll, sono utilizzabili solamente in un ambiente Microsoft Windows.

2.1.2.2 Arduino Nano

Le schede Arduino sono delle schede programmabili basate su risorse hardware e software open source. Sono di facile utilizzo e risultano infatti ideali per utenti con poca esperienza nell'ambito dei microcontrollori, ma costituiscono un'ottima soluzione anche per un'utenza esperta, soprattutto come strumenti per la prototipazione rapida dei propri progetti. Arduino fornisce inoltre un IDE, ovvero un ambiente di sviluppo, multipiattaforma e gratuito, tramite il quale è possibile scrivere i programmi da eseguire nel microcontrollore della scheda attraverso il linguaggio di programmazione proprietario Wiring, derivato dal C e C++ [3]. Esistono tuttavia numerose integrazioni per controllare le schede Arduino attraverso altri linguaggi di programmazione.
Arduino Nano è una delle 16 versioni hardware di Arduino; ha le stesse caratteristiche delle altre schede, ma si distingue per avere dimensioni e peso ridotti, rispettivamente 18x45 mm e 7 grammi. È basato sul microcontrollore ATmega328, della famiglia AVR, ed è quindi caratterizzato dalla presenza di una memoria flash interna, che nel caso dell'ATmega328 è di 32 kB, di cui 2 utilizzati dal boot loader; presenta inoltre numerose funzionalità per gestire la comunicazione con un computer, un altro Arduino o altri microcontrollori. In particolare, sulla scheda è situato un dispositivo, chiamato Ft232rl, che svolge la funzione di convertire la comunicazione seriale messa a disposizione dal microcontrollore in USB, fornendo quindi al sistema operativo del computer cui è collegato una porta COM virtuale tramite cui interagire con il microcontrollore stesso.
Nell'ambito del sistema di monitoraggio, la scheda viene utilizzata per leggere il valore di temperatura prodotto in output dal sensore relativo. Il programma che si occupa di gestire le

misure deve quindi interagire con la scheda Arduino sfruttando la porta COM virtuale che questo mette a disposizione.

2.1.3 Schema di installazione

Lo schema di installazione completo del sistema di monitoraggio è rappresentato in figura 4. In particolare, la figura mostra lo schema relativo all'installazione di Leon, quella situata all'interno del tunnel (da qui la sigla TU presente in tutti gli elementi), ma è valida anche per l'altro sito di monitoraggio, rispetto al quale variano solamente i nomi delle componenti. Inoltre, vengono riportati esclusivamente i collegamenti relativi alle schede AD5490BIOZ e non quelli con le schede Arduino Nano, i quali determinerebbero la rappresentazione di un collegamento in più fra ciascun blocco e l'hub centrale.

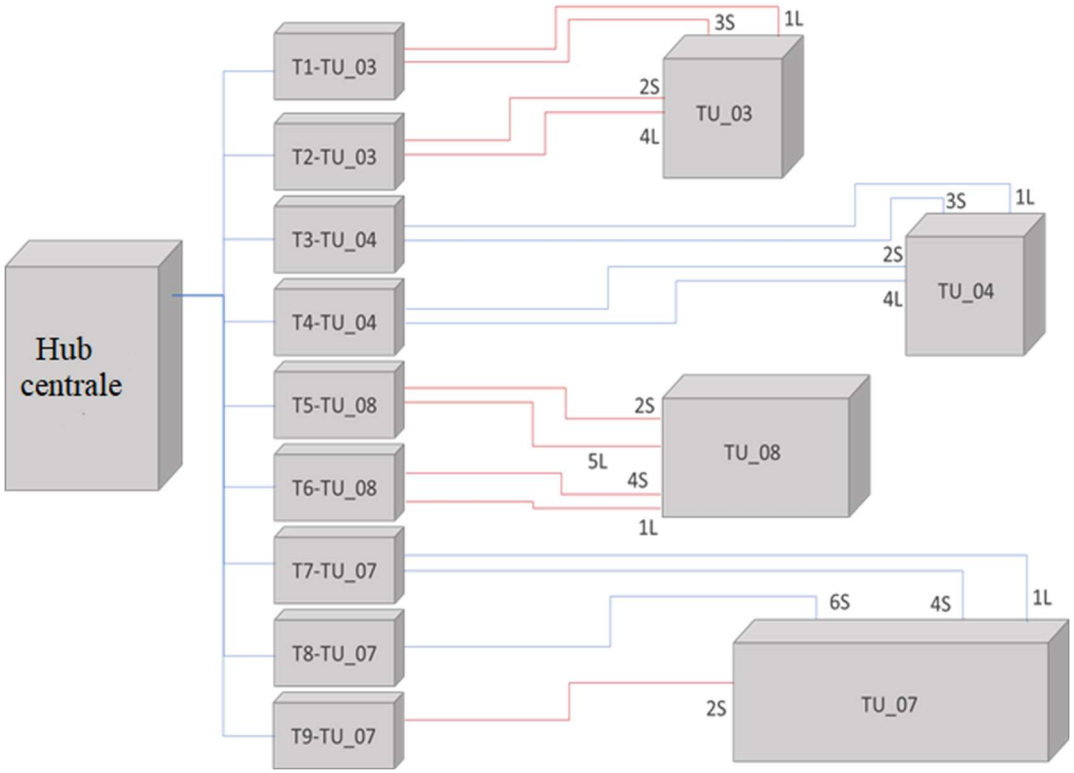

Figura 4 - Schema dell'installazione di Leon

Sulla destra vi sono i campioni di calcestruzzo, con all'interno i vari sensori che effettuano le misure; le schede di acquisizione sono poste, a gruppi di due, all'interno delle scatole elettriche, riempite di gel IP68, come visibile in figura 3. Due schede del blocco TU_07, per ragioni di spazio e mobilità, sono state invece collocate singolarmente all'interno delle scatole. Tutti i collegamenti confluiscono poi nell'hub centrale, al cui interno sono posizionate le componenti necessarie al funzionamento del monitoraggio, fra cui, ad esempio, il computer, con installato

il sistema operativo Microsoft Windows, che verrà controllato da remoto e su cui verrà eseguito il programma per effettuare le misure. È presente, inoltre, un hub seriale rs232 a 32 porte, che consente di collegare al computer i molteplici cavi provenienti dalle schede utilizzando un'unica porta USB, e un router, che permette il collegamento alla rete Internet.

2.2 SensorPal

SensorPal è il software che consente di controllare le schede di acquisizione AD5490BIOZ tramite un'interfaccia grafica, la quale permette di predisporre l'impostazione della misura attraverso la definizione di una serie di parametri e, successivamente, di visualizzare l'andamento dell'impedenza in forma grafica. Le funzionalità offerte da SensorPal sono visibili in figura 5.

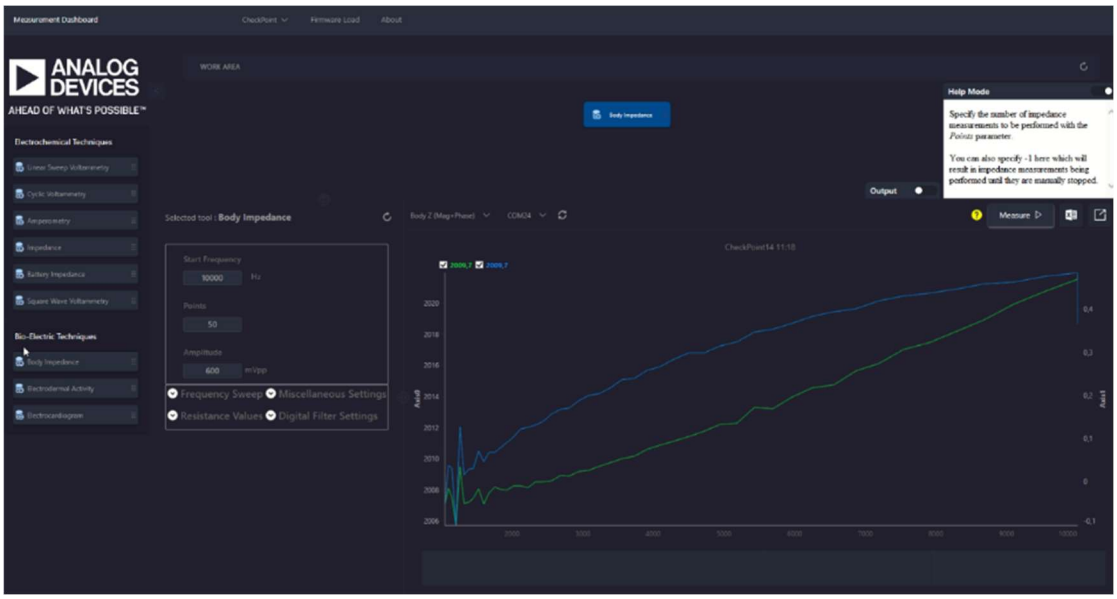

Figura 5 - Interfaccia grafica di SensorPal

Innanzitutto, tramite il menu localizzato sulla sinistra, è necessario indicare il tipo di misurazione che si vuole eseguire; nel caso dell'impedenza elettrica dei blocchi di calcestruzzo, va selezionata la tipologia "body impedance", che si trova sotto la voce "bio-electric techniques". La finestra situata al centro dell'interfaccia, leggermente spostata a sinistra, permette la variazione di una serie di parametri riguardanti la misura, come, ad esempio, la frequenza cui svolgere la misurazione e l'ampiezza del voltaggio da applicare. Vi è inoltre la possibilità di eseguire uno sweep in frequenza, una modalità che consiste nell'effettuare una serie di misure a frequenze comprese fra una iniziale e una finale, anch'esse parametrizzabili. Selezionando poi la porta COM cui è collegata la scheda di interesse, è possibile innescare la misura e visualizzarne i risultati in tempo reale nel grafico; il software permette di scegliere la curva da mostrare, se modulo, fase o entrambi, e di esportare i risultati ottenuti in un file Excel o CSV.

SensorPal costituisce il punto di partenza del sistema di monitoraggio, sia da un punto di vista progettuale che implementativo; contiene infatti le funzionalità base richieste dal progetto, ovvero la possibilità di effettuare le misure e di visualizzarne i risultati, ma le rende fruibili in modalità non adattabili ad un monitoraggio continuativo e automatico, in quanto le misure vanno attivate premendo un bottone, quindi con l'interazione di un utente, e il grafico che viene mostrato rappresenta l'andamento dei valori della misura appena eseguita, in tempo reale, senza alcun tipo di memoria. Inoltre, essendo SensorPal sviluppato appositamente per le schede AD5940BIOZ, non supporta il controllo della temperatura, che è invece un parametro rilevante per il sistema di monitoraggio.

L'idea da cui parte il progetto è dunque quella di estendere le funzionalità di SensorPal e strutturare il funzionamento del sistema in modo da consentire un monitoraggio duraturo ed efficace.

È necessario quindi sfruttare le API fornite da SensorPal descritte precedentemente per automatizzare il processo di esecuzione delle misure e implementare poi una logica per immagazzinare i dati; l'ultimo passo è costituito dalla realizzazione di un'interfaccia, idealmente simile a quella di SensorPal, tramite la quale poter visualizzare i dati raccolti nel corso del tempo.

CAPITOLO 3

Obiettivi del progetto

Il progetto illustrato in questa tesi, come già ampiamente discusso, ha come obiettivo lo sviluppo di un sistema di monitoraggio, in particolare la creazione dell'applicazione software che effettua le misure, interagendo con i sensori e con le schede di acquisizione, e ne gestisce la fruizione dei dati.

L'obiettivo del progetto, come suggerito dal concetto stesso di monitoraggio, può essere quindi riassunto nella necessità di ottenere l'andamento nel tempo dell'impedenza elettrica e della temperatura misurate nei blocchi di calcestruzzo realizzati con i materiali innovativi descritti nel capitolo 2.

Volendo tuttavia dettagliare meglio i macro-obiettivi che hanno guidato lo sviluppo dei software che compongono l'architettura nel suo complesso, si può far riferimento al semplice diagramma UML[1] dei casi d'uso riportato in figura 6.

[1] [1] Il linguaggio di modellazione UML (Unified Modeling Language) permette di visualizzare e analizzare sistemi software tramite l'utilizzo di specifici diagrammi, come quello dei casi d'uso.

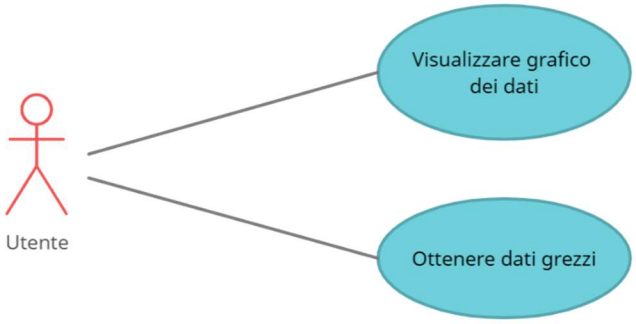

Figura 6 - Casi d'uso del sistema di monitoraggio

Un caso d'uso, in ambito informatico, rappresenta una modalità di utilizzo del software da parte degli attori che ne usufruiscono. La loro elaborazione deriva direttamente dall'analisi dei possibili fruitori dell'applicazione. Posizionandoli in un diagramma è possibile avere una visione d'insieme delle funzionalità che un dato sistema informatico deve implementare.
Un qualsiasi utente del software, rappresentato dall'uomo stilizzato sulla sinistra, deve avere la possibilità di effettuare due operazioni, che nel diagramma sono riportate all'interno degli ovali:

1. Visualizzare dei grafici che mostrino l'andamento nel tempo dell'impedenza elettrica o della temperatura di tutti i blocchi di calcestruzzo, con la possibilità di personalizzare il periodo di osservazione e regolare una serie di parametri, come, ad esempio, la frequenza di misurazione.
2. Ottenere i dati grezzi, quelli raccolti direttamente dai sensori, in modo da poterli analizzare a parte con strumenti più elaborati che esulano dal sistema di monitoraggio in sé.

Questi requisiti sono pensati per soddisfare le esigenze di chi deve analizzare i dati ottenuti dal sistema di monitoraggio, sia per averne un riscontro grafico, quindi immediato, sia per poterli elaborare e studiare a parte con più calma.
Lo sviluppo dell'architettura software è stato dunque guidato da questi casi d'uso e si è articolato nell'implementazione di una serie di funzionalità che, nel complesso, permettono di soddisfare le specifiche descritte.

3.1 Requisiti del software

Dopo aver definito i macro-obiettivi del progetto, la fase successiva è consistita nell'individuare i requisiti dell'architettura software del sistema.

La definizione dei requisiti è una fase cruciale nella produzione di un software; non è coinvolta alcun tipo di implementazione pratica, non viene deciso il linguaggio di programmazione o un particolare algoritmo da impiegare, ma ci si colloca ad un livello di astrazione più alto, in quanto si mira esclusivamente a descrivere ciò che un sistema deve realizzare. È altresì vero che tali decisioni tecniche si fondano proprio sui requisiti delineati in questa fase ed è perciò fondamentale effettuare un'analisi dettagliata che fornisca tutti gli elementi per compiere scelte corrette ed evitare quindi complicazioni nelle fasi successive della produzione.
I requisiti di un sistema si suddividono in due tipologie:

- Requisiti funzionali: descrivono, in senso stretto, ciò che un sistema deve fare, le sue funzionalità, il modo in cui reagisce a particolari tipi di input e come gestisce determinate situazioni. Considerando una stampante come un sistema esemplificativo, potremmo individuare come requisiti funzionali la possibilità di fotocopiare o scannerizzare un foglio.
- Requisiti non funzionali: caratterizzano le proprietà che il sistema deve possedere per soddisfare a pieno le richieste del committente. Rifacendosi all'esempio della stampante, un esempio di requisito non funzionale potrebbe essere la velocità, caratterizzabile in fogli stampati al minuto, oppure, semplicemente, l'essere economica, ovvero con un prezzo finale inferiore a quello delle stampanti concorrenti.

Requisiti funzionali e non funzionali sono, al di là della distinzione teorica, strettamente collegati: implementare tutte le funzionalità richieste ma senza rispettare le proprietà ad esse associate porta al non raggiungere gli obiettivi prefissati e, di conseguenza, al fallimento del progetto.
I requisiti del software che gestisce il sistema di monitoraggio sono stati dunque delineati basandosi su queste due categorie, con l'obiettivo di facilitare l'implementazione pratica dei vari moduli software che formano il sistema nel suo complesso.

3.2 Requisiti funzionali

I requisiti funzionali individuano il comportamento del sistema; la loro descrizione deve dunque essere sufficientemente dettagliata in modo da indicare chiaramente la strada da seguire in fase di sviluppo. In caso di problematiche, inoltre, essi devono costituire le linee guida cui fare affidamento mentre si ricerca una soluzione, per evitare che venga perso il focus sui veri obiettivi di ciò che si sta programmando.

3.2.1 Effettuare le misure

Il primo requisito consiste nell'effettuare le misure di impedenza elettrica e di temperatura nei blocchi di calcestruzzo. E' la funzionalità base su cui si fonda il sistema di monitoraggio, che non avrebbe senso di esistere senza una parte che si occupi di raccogliere i dati dei sensori.
Effettuare le misure significa interagire con le schede di acquisizione; è necessario quindi sfruttare le API fornite da SensorPal, descritte nel capitolo 2, e collegarsi serialmente alla scheda Arduino per azionare i sensori ed effettuare le misure.
Il processo di acquisizione delle misure deve essere automatico e deve avvenire un numero preciso di volte al giorno. L'intervallo di tempo che intercorre fra una misura e l'altra è stato poi fissato ad un'ora, delineando quindi la necessità di effettuare 24 misure al giorno, che devono essere compiute in tutti i blocchi di calcestruzzo di entrambe le installazioni.
In ultima istanza, la parte del sistema che si occupa di effettuare le misure deve essere in grado di gestire correttamente eventuali problemi che si possono verificare. Deve, ad esempio, individuare misure non andate a buon fine, assegnando conseguentemente valori di default, o gestire eventuali problematiche di accesso ad un sensore, senza che ciò blocchi l'acquisizione delle misure negli altri. È fondamentale, insomma, prevedere procedure sicure di gestione delle eccezioni, in modo da garantire misure integre e continuative, quindi veritiere.

3.2.2 Immagazzinare i dati

Il secondo requisito funzionale del sistema di monitoraggio è la capacità di immagazzinare in modo sicuro e facilmente fruibile i dati provenienti dalle misure effettuate dai sensori. Se l'obiettivo del sistema è quello di permettere di visualizzare l'andamento nel tempo delle grandezze misurate, è facile intuire l'importanza che svolge, in questo contesto, un'appropriata gestione dello storage dei dati.
E' fondamentale, innanzitutto, salvare più dati possibili, non limitandosi ai soli valori delle misure, ma includendo anche parametri statistici, con l'obiettivo, dopo un sufficiente periodo di monitoraggio, di essere in grado di ricostruire il più precisamente possibile le prestazioni e l'accuratezza del sistema sensoristico.
Un altro parametro da tenere in considerazione nel decidere come e con quali tecnologie implementare la gestione dei dati prodotti dal sistema è la sicurezza: si deve innanzitutto predisporre un backup che permetta di recuperare eventuali perdite di dati e gestire, inoltre, possibili problematiche che si riscontrino nell'accesso al database, senza che ciò impedisca il prosieguo dell'acquisizione delle misure e del salvataggio dei valori.

3.2.3 Fornire un'interfaccia

Il terzo e ultimo requisito funzionale riguarda la necessità di fornire ai fruitori del sistema di monitoraggio un'interfaccia tramite la quale interagire con i dati; in particolare, devono essere implementate le funzioni che permettono di espletare i casi d'uso precedentemente presentati: in primo luogo, visualizzare l'andamento nel tempo delle grandezze oggetto di monitoraggio sottoforma di grafici, semplici ma intuitivi, e, in secondo luogo, ottenere i dati grezzi per poterli analizzare a parte con più attenzione.
Il software che costituisce l'interfaccia deve inoltre dare la possibilità di personalizzare una serie di parametri nel momento in cui si accede ai dati, sia per visualizzarne il grafico, sia per estrarre i dati originali:

- Deve essere possibile selezionare il blocco di calcestruzzo i cui dati si vogliono visionare, con la possibilità di scegliere fra tutti quelli monitorati in entrambe le installazioni.
- Per ogni blocco, si deve poter visualizzare l'andamento dell'impedenza elettrica, sottoforma di modulo, fase, parte reale e parte immaginaria, e della temperatura.
- Deve essere possibile la selezione della frequenza di misurazione.
- Si deve poter configurare il periodo di osservazione, con la possibilità di scegliere anche una precisa fascia oraria.

In aggiunta a tutto ciò, l'interfaccia deve anche fornire una funzione che permetta l'applicazione di una media mobile semplice ai dati da visualizzare nel grafico, con una finestra temporale anch'essa selezionabile dall'utente.
Una media mobile semplice è uno strumento utilizzato nello studio di serie storiche, ovvero insiemi di valori che descrivono la dinamica di un fenomeno, come, nel caso del sistema di monitoraggio, l'andamento dell'impedenza elettrica misurata nei blocchi di calcestruzzo. Applicare una media mobile consiste nel sostituire ad ogni valore la media ponderata dei valori misurati prima e dopo, secondo un intervallo di tempo costante. Il risultato consiste in un grafico dall'andamento più morbido e regolare, insensibile a variazioni repentine, in quanto, di fatto, si è operato un filtraggio sui dati originali; questo effetto è tanto più accentuato quanto più la finestra temporale scelta è ampia.
Una media mobile consente di identificare più facilmente l'evoluzione delle grandezze osservate, riducendo l'effetto degli errori statistici che, per quanto insignificanti se presi singolarmente, possono influenzare il grafico nel suo complesso.

Dato che le misure vengono effettuate a distanza di un'ora una dall'altra, la finestra temporale configurabile deve anch'essa essere su base oraria, fino ad un massimo di 3 giorni.

3.3 Requisiti non funzionali

I requisiti non funzionali descrivono le caratteristiche che un software deve possedere al fine di soddisfare le esigenze per cui viene creato; non sono funzionalità da implementare, ma delle proprietà che devono caratterizzare il modo in cui il sistema informatico adempie le sue funzioni. Proprio per questo, i requisiti non funzionali risultano più complessi da garantire rispetto a quelli funzionali, ma la loro presenza determina, in ultima istanza, la vera riuscita del software.

3.3.1 Scalabilità

Per scalabilità, in ambito informatico, si intende la capacità di un software di garantire le sue proprietà, soprattutto a livello prestazionale, in occasione di variazioni della dimensione del sistema che gestisce.
Il software del sistema di monitoraggio, in particolare, deve garantire la possibilità di:

- variare il numero dei blocchi di calcestruzzo da monitorare in modo facile e senza intaccare il normale andamento del sistema;
- aggiungere o togliere parametri da personalizzare per la visione dei dati nell'interfaccia del sistema semplicemente inserendo il codice necessario e senza dover ripensare il modo in cui vengono immagazzinati i dati.

Tali requisiti richiedono, sostanzialmente, un software flessibile e adattabile, in modo che eventuali cambiamenti futuri necessità stravolgano ciò che è stato sviluppato e costringano quindi ad un ripensamento completo dell'architettura generale.

3.3.1 Manutenibilità

Il sistema di monitoraggio è pensato per garantire un periodo lungo di osservazione; a tal proposito, è fondamentale che il software che lo gestisce sia facilmente mantenibile, ovvero che sia semplice e intuitivo, nel caso di malfunzionamenti o necessità di aggiornamenti, mettere mano al suo codice e apportare le modifiche di cui si ha bisogno.
Un software, per essere mantenibile, deve essere organizzato secondo un'architettura modulare, caratterizzata da una serie di moduli a sé stanti, ognuno con delle precise funzionalità, ma che nel complesso realizzano le implementazioni necessarie. È sicuramente più facile, con

questo tipo di scelta strutturale, individuare i punti di interesse nel momento in cui si interagisce con il codice per applicare dei cambiamenti.
Un altro fattore chiave che determina la manutenibilità di un software è certamente un'adeguata documentazione, soprattutto nel caso in cui le persone che devono modificare il codice siano diverse da chi l'ha sviluppato.

3.3.3 Usabilità

Il concetto di usabilità è definito dall'ISO, l'organizzazione internazionale per la standardizzazione, come "l'efficacia, l'efficienza e la soddisfazione con le quali determinati utenti raggiungono determinati obiettivi in determinati contesti" [4]. Nell'ambito di prodotti informatici descrive quindi quanto un utente riesca a usufruire agevolmente dei servizi offerti, i quali devono quindi essere adeguati al fine di rendere il migliore possibile l'esperienza utente.
Forzando un po' la definizione e adattandola al caso del sistema di monitoraggio, l'interfaccia tramite cui visionare i dati deve essere chiara e facile da utilizzare, con le varie funzionalità ben visibili e presentate con uno stile elegante, minimale e di semplice comprensione.
Un software usabile ha diversi benefici: se, da una parte, migliora l'esperienza utente, rendendone più piacevole l'utilizzo, dall'altra alleggerisce anche il codice da sviluppare, il quale sarà infatti privo di meccanismi complessi e fini a sé stessi.

CAPITOLO 4

Strumenti Software

Dopo aver individuato le caratteristiche del sistema di monitoraggio, definite in termini di funzionalità e di proprietà, la fase successiva dello sviluppo è consistita nella delineazione delle tecnologie da utilizzare per implementare il sistema. Tali decisioni sono state guidate dai requisiti, funzionali e non funzionali, analizzati precedentemente, con l'obiettivo di creare dei moduli software che li rispettassero a pieno.

Il primo passo è stato quello di progettare l'architettura del sistema, ovvero identificare la sua struttura generale, organizzata in singoli moduli che comunicano secondo protocolli ben definiti.

Definire l'architettura è fondamentale nella progettazione di un software: realizzare le singole componenti avendo ben chiaro il loro collocamento nel sistema è sicuramente più corretto rispetto ad implementarne le varie parti senza consapevolezza del ruolo che svolgono, ma soprattutto vi sono miglioramenti generali nella qualità del sistema sviluppato, che risulta più facilmente manutenibile e scalabile, come peraltro richiesto nei requisiti non funzionali del sistema di monitoraggio.

Successivamente sono state definite le tecnologie con cui sviluppare le varie componenti del sistema, definendo quindi quali soluzioni adottare a livello di linguaggi di programmazione, di DBMS, di protocolli con cui effettuare la comunicazione fra i vari moduli, e, in ultima istanza, anche quali strumenti software adoperare per scrivere il codice vero e proprio.
In questa sezione dell'elaborato, dopo aver presentato brevemente l'architettura del sistema, la quale verrà ripresa e analizzata più accuratamente nel capitolo successivo, verranno descritte le tecnologie utilizzate, cercando di puntualizzarne i punti di forza che hanno determinato la loro scelta.

4.1 Architettura del sistema

Si possono individuare due livelli in cui suddividere l'architettura del sistema di monitoraggio. Il primo è responsabile della comunicazione con i sensori e dell'archiviazione dei dati. È essenzialmente composto da un programma, costantemente in esecuzione sulle macchine localizzate nei siti di monitoraggio, il quale:

- Effettua le misure, comunicando con i sensori attraverso le API delle schede di acquisizione.
- Immagazzina i dati raccolti in due database, uno situato in locale, risiedente come file presso la macchina stessa, l'altro invece implementato in cloud presso il provider di servizi web Amazon Web Services.

La scelta di ricorrere a due database è stata fatta per garantire una maggiore sicurezza nel mantenimento e nell'integrità dei dati raccolti, i quali di fatto si trovano duplicati in due istanze diverse e non collegate fra loro. In particolare, in caso di assenza di connessione alla rete Internet, circostanza plausibile in luoghi come le zone di monitoraggio, specialmente nel lungo periodo, e che impedirebbe la comunicazione con il database in cloud, si ha sempre la possibilità di preservare i dati provenienti dalle misure grazie a quello locale.
Il software è stato realizzato con il linguaggio di programmazione Python, mentre i database sono istanze PostgreSQL e SQLite, la prima delle quali risiede nei servizi di AWS.
Il secondo livello dell'architettura, invece, si occupa di gestire i dati raccolti e implementa l'interfaccia con cui gli utenti interagiscono per accedere ai dati del sistema di monitoraggio. La comunicazione fra l'applicazione e il database remoto avviene tramite un server, implementato grazie al servizio offerto da AWS chiamato Amazon EC2 (*Elastic Compute Cloud*), mentre l'interfaccia è costituita da un'applicazione mobile, la cui scelta è stata dettata dalla necessità di rendere la fruizione dei dati del sistema facile, accessibile e veloce, oltre che disponibile in

qualsiasi momento della giornata, senza quindi la necessità di utilizzare un computer o collegarsi ad una pagina web.
Il server è stato creato con Python, mentre per l'applicazione mobile è stato scelto il framework Flutter e il linguaggio di programmazione Dart.

4.2 Python

Python è un linguaggio di programmazione ideato e implementato dall'informatico olandese Guido van Rossum a partire dal 1989; attualmente il suo sviluppo è gestito dall'organizzazione no-profit Python Software Foundation. È un linguaggio che supporta più paradigmi di programmazione; sebbene sia stato pensato per essere prevalentemente orientato agli oggetti, è infatti in grado di gestire anche costrutti di tipo imperativo ed ha caratteristiche di programmazione funzionale [5]. È inoltre un linguaggio ad alto livello, caratterizzato ovvero da una rilevante astrazione rispetto al funzionamento del calcolatore, cosa che si riflette in una sintassi più vicina al linguaggio naturale che a quello macchina.
La filosofia che sta alla base di Python e del suo design è espressa nel manifesto "Zen of Python", redatto dall'ingegnere del software Tim Peters nel 1999 e pubblicato sulla mailing list ufficiale [6]. Si tratta di una raccolta di 19 frasi che rappresentano i principi fondanti del linguaggio; qui vengono riportati, in lingua originale, alcuni dei più significativi:

- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.

Python si distingue per un codice pulito e leggibile, per una sintassi leggera e comprensibile e per l'obiettivo dichiarato di adottare soluzioni semplici ma efficaci.

4.2.1 Caratteristiche del linguaggio

Python viene spesso descritto come un linguaggio interpretato, ma in realtà il codice sorgente, localizzato in file con l'estensione *.py*, viene prima compilato in un bytecode, i cui file sono invece caratterizzati dall'estensione *.pyc*, e solo a questo punto eseguito dall'interprete Python noto come PVM, Python Virtual Machine. Il processo è raffigurato in figura 7 [7].

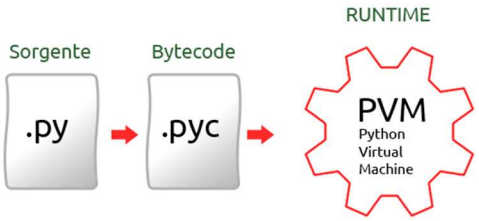

Figura 7 - Processo che porta all'esecuzione di un codice Python

Queste caratteristiche rendono il codice Python portabile, in quanto può essere eseguito in qualsiasi macchina con la stessa versione dell'interprete.

La PVM supporta inoltre un REPL (*Read Only Print Loop*), un modo d'uso che permette di eseguire singole espressioni direttamente da riga di comando visualizzando subito l'output prodotto, utile specialmente in fase di debug del codice.

Per quanto riguarda la sintassi, Python è caratterizzato da variabili non tipizzate e dalla mancanza di parentesi o di parole chiave per evidenziare i diversi blocchi di codice, che sono invece delineati da diversi livelli di indentazione. Il controllo dei tipi è comunque forte e viene eseguito a runtime, con le variabili che sono in realtà dei puntatori a oggetto. È inoltre presente un meccanismo di garbage collector per la gestione della memoria, che non necessita quindi dell'intervento esplicito del programmatore per essere allocata o liberata. Tutte queste caratteristiche rendono la velocità di esecuzione di programmi scritti in Python inferiore rispetto a quella di altri linguaggi compilati e a tipizzazione statica, come il C o il C++, ma, se comparato con altri linguaggi di scripting come, ad esempio, Perl, Python resta un'ottima soluzione anche a livello prestazionale. Esiste inoltre la possibilità di integrare delle porzioni di codice scritte in linguaggi più performanti all'interno di codice Python, con la possibilità quindi di sfruttarne la velocità di esecuzione nelle parti in cui si ha bisogno.

4.2.2 Punti di forza

Python è, innanzitutto, gratuito, quindi utilizzabile senza alcun tipo di restrizione o copyright. Nonostante ciò, è comunque caratterizzato da una comunità molto attiva, che lo mantiene costantemente aggiornato e al passo con i tempi, anche attraverso la produzione di una documentazione dettagliata, sottoforma di guide scritte e di tutorial video, che aiuta gli sviluppatori di qualsiasi livello ad acquisire o espandere le proprie conoscenze.

Un altro punto di forza di Python è la presenza di innumerevoli librerie che semplificano e riducono il codice a carico degli sviluppatori, specialmente nelle prime fasi di creazione del

software. I package Python permettono, ad esempio, di produrre grafici e diagrammi complessi, di interagire con il sistema operativo ed il file system o, ancora, di interrogare API esterne tramite il protocollo HTTP. Proprio in virtù del gran numero di moduli, sia presenti nella libreria standard che prodotti da terze parti, di cui Python è dotato, il suo utilizzo è diffuso in un gran numero di applicazioni, anche molto diverse tra loro, che vanno dallo sviluppo web, grazie a framework come Flask o Django, fino a software in ambito scientifico, sfruttando le potenzialità di librerie come Numpy o Scipy. Python è insomma un linguaggio molto versatile, ma in ciascuno dei suoi campi di applicazione mantiene la sua efficienza e la sua flessibilità.
Un'altra caratteristica che rende Python uno dei linguaggi più utilizzati al mondo è la sua sintassi, le cui linee guida sono già state presentate attraverso i principi espressi nel manifesto. Il codice Python si caratterizza per essere comprensibile e pulito; ciò facilita lo sviluppo dei software e rende più veloce il processo di comprensione e apprendimento, oltre a semplificare il lavoro di manutenzione del codice.
Tutte queste caratteristiche hanno determinato la scelta di Python come linguaggio con cui sviluppare il server del sistema di monitoraggio e il programma tramite cui effettuare le misure interagendo con i sensori.

4.2.3 Librerie utilizzate

Python si distingue per la presenza di numerose librerie e framework che ne ampliano e potenziano le possibilità. Riportate di seguito, vi sono quelli che hanno svolto un ruolo importante negli applicativi sviluppati per il sistema di monitoraggio.

4.2.3.1 Flask

Flask è un framework che permette lo sviluppo di servizi web tramite Python. È in realtà classificato come micro-framework, non perché sia caratterizzato da poche funzionalità, quanto per l'obiettivo di avere un nucleo fondamentale semplice ed essenziale, in modo da lasciare allo sviluppatore la possibilità di implementare funzioni più avanzate secondo le proprie necessità.
È stata utilizzata anche la libreria Flask-RESTful, un'estensione per Flask che aggiunge funzionalità per la costruzione di REST Api.

4.2.3.2 Numpy e Pandas

Numpy, il cui nome deriva dalla fusione delle parole "Numeric Python", è una libreria che fornisce il supporto per operazioni matematiche complesse su matrici e array di grandi dimensioni. È di fatto la libreria standard per qualsiasi implementazione di tipo scientifico in Python,

in quanto consente la stesura di programmi concisi e facili da leggere, ma allo stesso tempo veloci ed efficienti. Ciò avviene tramite tecniche come la vettorizzazione e il broadcasting, che sfruttano codice precompilato scritto in C per ottimizzare la velocità di esecuzione di calcoli matematici sulle strutture dati messe a disposizione da Numpy stesso.
Pandas è invece una libreria le cui funzioni vengono utilizzate per l'analisi e la manipolazione dei dati. In particolare, fornisce funzioni specifiche che facilitano l'importazione o l'esportazione di dati in vari formati, come CSV, Microsoft Excel o JSON.

4.2.3.3 Sqlite3 e Psycopg2

Sqlite3 e Psycopg2 sono le librerie che permettono l'interazione di programmi Python con i database, rispettivamente Sqlite la prima e PostgreSQL la seconda.
Sqlite3 è un modulo che gestisce l'accesso al file contenente il database, ne assicura il blocco per prevenire corruzione dei dati quando più utenti vogliono accedervi in scrittura e permette l'esecuzione di query SQL. Psycopg2, un wrapper di libpq, la libreria client ufficiale scritta in C, è il driver Python più popolare per interagire con un database PostgreSQL; la sua principale caratteristica è la capacità di gestire in modo sicuro ed efficiente applicazioni multi-threaded che lavorino in modo concorrente sul database.
Entrambe le librerie forniscono funzionalità SQL conformi con le specifiche espresse nel PEP[2] 249 dal titolo "Python Database Api Specification v2.0", il quale incoraggia l'implementazione di metodi simili per accedere a database diversi, basandosi quindi su interfacce comuni. Nella pratica, ciò si traduce in moduli caratterizzati da sintassi simili, facilitando così la comprensione e la portabilità del codice che li coinvolge.

4.3 Flutter e Dart

Le applicazioni mobile sono dei software realizzati per essere eseguiti su dispositivi mobili, come smartphone o tablet. L'evoluzione continua che ha caratterizzato il mercato di settore ha incentivato gli sviluppatori a studiare soluzioni sempre più complesse ed efficienti per produrre applicazioni; ciò ha determinato, nel corso del tempo, la nascita di più tipologie di app, le quali si caratterizzano per le diverse tecnologie con cui sono implementate.
Vi sono, innanzitutto, le applicazioni native, ovvero quello create appositamente per un particolare sistema operativo; i linguaggi di programmazione utilizzati sono Java o Kotlin per Android, mentre Swift o Objective-C sono quelli cui si fa ricorso per iOS. Le applicazioni native,

[2]Le PEP (Python Enhancement Proposals) sono dei documenti che descrivono nuove caratteristiche del linguaggio Python o che forniscono semplicemente delle informazioni alla comunità degli sviluppatori.

essendo specifiche per un sistema, si caratterizzano per offrire prestazioni ottimali e per la possibilità di interagire in modo completo con l'hardware dei dispositivi, sfruttando, ad esempio, le funzionalità dei sensori di cui questo è dotato. La produzione di app native risulta però lenta e costosa, in quanto, per essere fruibili su più piattaforme, richiedono la scrittura di codici diversi.

Una soluzione che si è posta l'obiettivo di ovviare all'assenza di un linguaggio di programmazione condiviso da tutti i sistemi operativi mobile è quella delle app ibride, le quali costituiscono un anello di congiunzione fra le app native e le web app; vengono realizzate con tecnologie tipiche dello sviluppo Web, come HTML, CSS e JavaScript, ma per funzionare devono comunque essere scaricate e installate sul dispositivo mobile. Sebbene introducano una notevole semplificazione, essendo caratterizzate da un unico codice eseguibile su tutte le piattaforme, le app ibride non riescono a sfruttare completamente le potenzialità dei dispositivi e sono dunque sconsigliate per applicazioni complesse o in cui siano richieste performance elevate.

Un'ulteriore tipologia di applicazioni mobile è quella delle app cross-platform, le quali costituiscono degli applicativi installabili sui dispositivi e, similmente a quelle ibride, funzionanti su più piattaforme senza bisogno di riscriverne il codice. Si distinguono per essere caratterizzate da ottime performance, paragonabili a quelle native, e per consentire, inoltre, l'integrazione di codice nativo nel caso in cui vi sia la necessità di interagire con un particolare componente hardware, altrimenti non raggiungibile, superando così un limite delle app ibride.

Flutter, un framework realizzato da Google a partire dal 2017, si colloca in questo scenario, offrendo la possibilità, attraverso il linguaggio di programmazione Dart, di sviluppare applicazioni cross-platform efficienti e con interfacce utente funzionali e facilmente componibili.

4.3.1 Flutter

Flutter è un software open source, rilasciato con una licenza BSD. È un framework leggero e facilmente integrabile in ambienti di lavoro tipici dello sviluppo mobile come Android Studio; semplifica la realizzazione del layout grafico, adattandolo alla piattaforma su cui l'app deve essere eseguita, e permette così allo sviluppatore di concentrarsi sull'implementazione delle funzionalità. Sebbene sia una tecnologia relativamente nuova, costituisce ormai uno standard nella realizzazione di applicazioni mobile ed è infatti già utilizzato da moltissime aziende, come Alibaba, la più grande società di commercio online al mondo, che ha impiegato proprio Flutter per creare la sua applicazione disponibile su Android e iOS, con più di 50 milioni di download [8].

4.3.1.1 Struttura

Flutter è caratterizzato da un'architettura a strati, rappresentata in figura 8; tutte le componenti di un livello dipendono da quello sottostante, senza che possano però accedervi in modo privilegiato.

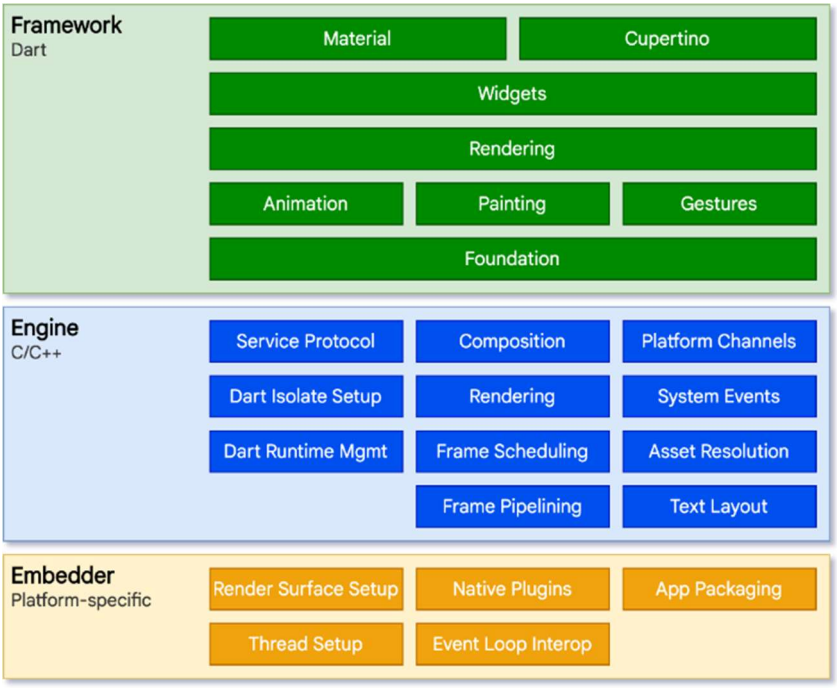

Figura 8 - Architettura del framework Flutter

Lo strato più esterno, denominato Framework, è scritto con il linguaggio Dart ed è quello con cui gli sviluppatori interagiscono per implementare le funzionalità ed il layout della propria applicazione. Sono presenti le librerie Material e Cupertino, le quali forniscono una vasta serie di funzioni e classi che permettono di personalizzare il design della propria app inserendo componenti tipici dei dispositivi Android ed Apple.

Lo strato Engine, il vero cuore di Flutter, implementa tutte le primitive necessarie alla creazione delle applicazioni. Sviluppato in C e C++, è responsabile dell'interazione con il file system e con la rete, gestisce il rendering grafico, che avviene tramite la libreria di Google Skia Graphics, e si occupa di integrare eventuali componenti native, le quali si collegano al codice Dart attraverso i Platform Channels.

L'interazione con il sistema operativo del dispositivo è invece coordinata dall'ultimo strato che compone la struttura, l'Embedder, scritto nel linguaggio specifico della piattaforma, come Java per Android o Objective-C per iOS. In questo modo, le applicazioni realizzate in Flutter sono viste come se fossero native dal sistema operativo sottostante, garantendo così performance elevate [9].

4.3.1.2 Caratteristiche

Le componenti fondamentali delle applicazioni Flutter sono i Widget, ovvero gli elementi visuali che compongono le interfacce utente. Sono rappresentati da classi immutabili e internamente sono gestiti attraverso alberi. Per dichiarare o modificare l'interfaccia di un Widget è necessario effettuare l'override del metodo build, il quale è appositamente implementato in modo che abbia un'esecuzione veloce e priva di effetti collaterali, così da poter essere richiamato più volte senza che vi sia un degrado delle prestazioni.
Ci sono due tipologie di Widget:

- Stateless Widget, che si caratterizzano per l'assenza di uno stato, non avendo delle proprietà che possono cambiare nel tempo.
- Stateful Widget, i quali sono invece dinamici, caratterizzati quindi da proprietà che variano, ad esempio, in seguito ad un'interazione dell'utente o al risultato di una chiamata di rete. Quando lo stato cambia, è necessario richiamare il metodo setState per notificare la modifica al framework e fargli costruire di nuovo il widget, eliminando l'istanza modificata dall'albero e aggiungendo quella nuova.

Il layout grafico si ottiene dunque attraverso la composizione di Widget, con o senza memoria, che possono essere innestati fra loro in modo tale da creare una struttura gerarchica facilmente modificabile.

4.3.2 Dart

Dart è un linguaggio di programmazione sviluppato da Google a partire dal 2011. Nato con l'intento di sostituire JavaScript come linguaggio per lo sviluppo Web, può essere attualmente utilizzato sia in questo ambito, attraverso la piattaforma Dart Web, che per realizzare applicazioni mobile o desktop; ciò avviene mediante la piattaforma Dart Native, una macchina virtuale che si caratterizza per la presenza di due tipologie di compilatori:

- Compilatore JIT, just in time, il quale permette la compilazione incrementale e la raccolta di metriche sulle prestazioni dell'app.
- Compilatore AOT, ahead of time, con il quale il codice sorgente dell'app viene compilato in codice binario per le diverse piattaforme su cui verrà eseguito; è la compilazione da utilizzare per effettuare la distribuzione del software.

4.3.2.1 Caratteristiche del linguaggio

Dart è un linguaggio orientato agli oggetti, ma supporta anche la programmazione funzionale e la riflessione. È un sistema fortemente tipizzato e "type safe": in fase di compilazione effettua infatti controlli statici sui tipi delle variabili e vengono lanciate eccezioni nel caso in cui valore e tipo di una variabile non corrispondano. Sono tuttavia presenti meccanismi di inferenza, che permettono l'omissione della dichiarazione del tipo di una variabile attraverso l'utilizzo della parola chiave dynamic.

Un'altra rilevante caratteristica del linguaggio Dart è il supporto alla null safety, tramite cui si impedisce alle variabili di assumere un valore null a meno che non lo si dichiari in modo esplicito. Viene quindi effettuata un'analisi statica sul codice, la quale previene eventuali null exceptions a runtime. [10]

4.3.2.2 Librerie Utilizzate

Dart è dotato di una vasta serie di librerie, sia prodotte dal team di sviluppo ufficiale che da terze parti, le quali arricchiscono le funzionalità offerte dal linguaggio.

I moduli principali utilizzati per sviluppare l'applicazione del sistema di monitoraggio sono:

- **http**: contiene tutte le classi e funzioni necessarie per effettuare richieste tramite il protocollo http. Viene utilizzato per comunicare con il server ed ottenere i dati immagazzinati nel database.
- **async:** permette l'implementazione di funzionalità asincrone all'interno dell'applicazione; introduce, in particolare, il tipo Future <T>, che costituisce il risultato di un'operazione asincrona il cui risultato è di tipo T.
- **charts-flutter:** consente la visualizzazione di dati attraverso grafici di molte tipologie; tutte le funzionalità che offre implementano i principi del Material Design. Nell'applicazione si fa ricorso a grafici di tipo time series, caratterizzati ovvero da valori temporali nell'asse delle ascisse.

4.3.3 Punti di forza

Il principale vantaggio che deriva dall'utilizzare Flutter e Dart come strumenti di sviluppo è quello di necessitare la creazione di un unico codice per ottenere applicazioni funzionanti in tutti i dispositivi mobile, sia Android che iOS, riducendo così i tempi e le difficoltà che si incontrano nella produzione. Flutter consente di velocizzare ulteriormente la scrittura dell'applicazione abilitando la possibilità di effettuare, in fase di sviluppo, l'hot reload, una particolare modalità di compilazione che avviene grazie all'interazione del framework con il compilatore

JIT di Dart. Ciò consiste nell'iniettare codice sorgente nei file che sono già in esecuzione nella macchina virtuale, dando così la possibilità di visualizzare in breve tempo gli effetti delle modifiche che vengono man mano applicate.
Le applicazioni Flutter si caratterizzano inoltre per performance ottimali, paragonabili a quelle di applicazioni scritte in codice nativo, e per una struttura del codice organizzata e modulare, quindi facilmente estensibile e mantenibile, ma soprattutto agilmente adattabile ai propri scopi. L'interfaccia grafica, organizzata tramite Widget innestati fra loro, risulta efficace e semplice da realizzare.
Flutter e Dart sono insomma tecnologie di facile apprendimento ma dalle grandi potenzialità; per questo sono stati individuati come strumenti tramite cui sviluppare l'applicazione del sistema di monitoraggio.

4.4 DBMS

I DBMS, ovvero Database Management System, sono dei software che consentono la gestione di una base di dati. Si pongono, dunque, come strato intermedio fra chi vuole accedere alle informazioni contenute in un database, sia esso un utente o un software, e i dati stessi, fornendone una rappresentazione logica interrogabile mediante delle istruzioni ben definite.
L'integrazione di un DBMS in un sistema informatico che debba immagazzinare e analizzare dei dati è di fondamentale importanza. Il sistema di monitoraggio fa ricorso a due DBMS, uno risiedente nella macchina che effettua le misure, SQlite, l'altro invece implementato in cloud, usufruendo dei servizi di AWS, denominato PostgreSQL.

4.4.1 SQlite

SQlite è un DBMS scritto in linguaggio C che si distingue per offrire la possibilità di creare e manipolare una base di dati risiedente in unico file. È in realtà una libreria e non può quindi essere eseguita come processo autonomo, ma deve essere integrata in un software sviluppato in linguaggi che ne consentano il binding tramite API. Il suo codice sorgente è open source e consultabile su Fossil, un sistema di controllo versione decentralizzato sviluppato da Richard Hill, lo stesso creatore di SQlite.
È definito "zero-configuration database" [11], in quanto, a differenza dei database che sfruttano il modello Client-Server, non implementa alcun tipo di controllo sugli accessi, i quali vengono invece gestiti dai permessi del file system dati al file stesso. Un'altra conseguenza della sua architettura server-less è quella di non supportare l'accesso contemporaneo di più processi in

scrittura al database, in quanto la gestione della concorrenza è affidata ai lock del sistema operativo che non implementano meccanismi di gestione del parallelismo.
SQlite implementa lo standard SQL per l'esecuzione delle query, ma si differenzia per un controllo debole sui tipi dei dati che vengono inseriti, non garantendo quindi l'integrità di dominio. Si caratterizza per essere molto efficiente, grazie alla leggerezza della libreria ma soprattutto alla veloce interazione con il file, e per garantire transazioni ACID (atomiche, consistenti, isolate e durabili); supporta database anche molto grandi, fino a una dimensione di 140 TB, ed è multipiattaforma.
Sebbene non sia la scelta ottimale per applicazioni multiutente o che richiedano elevati volumi di scrittura, SQlite rappresenta una delle scelte più popolari nell'ambito dei software che richiedano uno storage interno; esempi di programmi celebri che ne fanno uso sono i web browser Google Chrome e Mozilla, che utilizzano SQlite per immagazzinare informazioni sugli utenti, e il sistema operativo mobile Android.
Anche per il sistema di monitoraggio è stata decisa l'implementazione di un database SQlite sulla macchina in cui vengono effettuate le misure, con l'obiettivo di ottenere uno storage sicuro ed efficiente da utilizzare come backup in caso di perdita o corruzione dei dati risiedenti nel database remoto.

4.4.2 PostgreSQL

PostgreSQL è un DBMS ad oggetti-relazionale, in cui l'informazione, similmente ai linguaggi di programmazione che implementano questo paradigma, viene rappresentata sottoforma di oggetti. Inizialmente denominato Postgres, in quanto considerato il successore del gestore di database Ingres, il suo nome è stato cambiato in PostgreSQL per evidenziare il supporto al linguaggio SQL. Il codice che lo implementa, realizzato in linguaggio C, è libero e opensource.
È un database che utilizza il modello Client-Server. Una tipica sessione di lavoro consiste nella collaborazione fra due processi: il server, che comunica direttamente con la base di dati e gestisce le richieste di accesso e modifica che arrivano dai vari utenti, e il client, il quale richiede di accedere al database per leggere o modificarne i dati.
L'accesso concorrente dei client viene gestito con un approccio diverso da quello utilizzato da altri DBMS. PostgreSQL implementa infatti un controllo della concorrenza multiversione, che sostituisce i lock su righe e tabelle impiegati da altri sistemi. Questo metodo si basa sull'assegnare ad ogni transazione uno snapshot del database rappresentante i dati all'inizio della connessione, dando la possibilità di effettuare delle modifiche senza influenzare le transazioni condotte da altri client. Ciò avviene mediante l'aggiunta di un nuovo record ogni volta che si

effettua un aggiornamento dei dati e marcando come invalida la riga modificata, la quale resta però disponibile per gli snapshot che devono poterla vedere. In questo modo l'accesso simultaneo al database viene gestito efficientemente e viene garantito lo standard ACID per le transazioni.
PostgreSQL è un DBMS completo, caratterizzato da numerose funzionalità che lo rendono uno dei più popolari in assoluto: supporta l'esecuzione di query SQL complesse, include nativamente una grande varietà di tipi di dati, tra cui, ad esempio, il JSON, implementa le stored procedures, le quali, attraverso il linguaggio proprio di PostgreSQL chiamato PL/pgSQL, permettono di eseguire programmi direttamente nel database, semplificando il processo di comunicazione fra client e server nel caso di query complesse. Attraverso i WAL (write-ahead logs), una tecnica che consiste nella memorizzazione delle modifiche che avvengono sul database, fornisce protezione dalla perdita o dalla corruzione dei dati, che possono essere ripristinati ripercorrendo la storia del database.
È insomma un sistema di gestione flessibile ed efficiente e rappresenta quindi la scelta ideale per un'applicazione come il sistema di monitoraggio che richiede lo storage sicuro di una grande quantità di dati.

4.4 AWS

Amazon Web Services, abbreviato in AWS, è un'azienda che offre prodotti informatici secondo il modello del cloud computing, una modalità di erogazione di servizi che consiste nel mettere a disposizione dell'utente infrastrutture e risorse attraverso la rete Internet, quindi disponibili e configurabili da remoto [12]. Attivo ormai dal 2006, AWS implementa le sue tecnologie grazie a server farm dislocate in tutto il mondo; è di proprietà del gruppo Amazon, di cui rappresenta il 63% del fatturato totale [13].
I vantaggi legati all'utilizzo dei servizi offerti da AWS sono molteplici e di vario tipo. Innanzitutto, vi è la convenienza dal punto di vista economico, in quanto i costi previsti si basano sul modello "Pay as you go", che consiste nell'applicare tariffe in base al consumo effettivo dei servizi. In questo modo l'allocazione di risorse si adegua alle necessità che si fronteggiano nel corso del tempo e non alle previsioni fatte all'inizio della progettazione, annullando rischi di eccessivo provisioning o di capacità insufficiente. In secondo luogo, i servizi AWS si caratterizzano per una serie di benefici tecnici: sono facilmente scalabili, grazie alla solida infrastruttura di Amazon che permette di adattare le risorse alle proprie esigenze, garantiscono ottime prestazioni e sono estremamente flessibili, in quanto supportano una grande quantità di sistemi operativi, linguaggi di programmazione o DBMS; la fruizione dei servizi è inoltre agevolata

da una console di gestione efficace e da un'ampia documentazione presente sul sito. La sicurezza è, infine, uno dei cardini della piattaforma: l'infrastruttura è infatti dotata di certificazioni e audit riconosciuti in tutto il settore e garantisce l'integrità e la protezione dei dati [14].
I servizi cloud di AWS di cui si è usufruito per l'implementazione del sistema di monitoraggio sono due: Amazon EC2, per il server, e Amazon RDS, per ospitare il database.

4.4.1 Amazon EC2

Amazon Elastic Compute Cloud, noto con la sigla Amazon EC2, è un servizio web che fornisce il controllo e l'accesso a macchine virtuali personalizzabili su cui eseguire i propri software. È un servizio scalabile, in quanto fornisce capacità computazionali adattabili in maniera dinamica alle proprie esigenze.
La configurazione delle istanze virtuali avviene attraverso dei modelli noti come Amazon Machine Image (AMI), contenenti tutte le informazioni necessarie per avviare il server, compreso il sistema operativo e altre funzionalità aggiuntive. Amazon fornisce numerose AMI con configurazioni comuni, ma ogni sviluppatore può crearsene una personalizzata. Il login alle macchine virtuali è sicuro e sfrutta la crittografia asimmetrica, secondo cui ad ogni istanza è associata una coppia di chiavi, quella pubblica gestita da AWS e quella privata assegnata all'utente. Le macchine EC2 sono inoltre dotate di indirizzi IP elastici, ovvero degli indirizzi IP statici che possono essere rimappati, in caso di necessità, in modo dinamico, e di un firewall, che permette di specificare i protocolli, le porte e gli intervalli di indirizzi IP che possono raggiungerle attraverso un meccanismo noto come gruppi di sicurezza.

4.4.1 Amazon RDS

Amazon Relational Database Service, o Amazon RDS, è un servizio che consente la creazione e la gestione di un database relazionale in cloud. Il suo scopo è quello di facilitare le operazioni di impostazione e dimensionamento del database; gestisce infatti attività dispendiose in termini di tempo come il provisioning dell'hardware, gli aggiornamenti del DBMS e il backup dei dati, garantendo inoltre prestazioni ottimali nell'esecuzione delle query e protezione da perdite o corruzione dei dati.
Sono disponibili diverse istanze di database, fra cui PostgreSQL, quello utilizzato nel sistema di monitoraggio, MySql, MariaDB e il DBMS proprietario di Amazon, Aurora.

4.5 Tool di sviluppo

Lo sviluppo del software del sistema di monitoraggio è stato supportato da numerosi strumenti tecnologici che hanno facilitato e velocizzato la creazione degli applicativi. Sono stati innanzitutto utilizzati degli IDE, ovvero ambienti di sviluppo integrati, tramite cui è avvenuta la scrittura dei codici Python e Dart; si è fatto poi uso di programmi che forniscono interfacce grafiche per l'amministrazione dei database e di software che consentono la comunicazione con le macchine remote. In questa sezione verranno presentati i principali strumenti utilizzati e le caratteristiche che ne hanno determinato la scelta.

4.5.1 Spyder

Spyder è un ambiente di sviluppo integrato per la scrittura di codice Python; inizialmente creato dal programmatore Pierre Raybaut nel 2009, è ora open source e mantenuto da un'attiva comunità di sviluppatori che ne migliorano costantemente le caratteristiche. Spyder è progettato per supportare la programmazione scientifica e presenta, infatti, delle funzionalità che permettono di integrare le principali librerie utilizzate in questo ambito, come Numpy, Scipy e Pandas, ma le sue caratteristiche funzionali lo rendono una scelta popolare anche per progetti di altro tipo.

La figura 9, presente sul repository GitHub ufficiale del progetto [15], rappresenta la schermata principale di Spyder ed evidenzia tutte le funzionalità che l'IDE mette a disposizione degli sviluppatori.

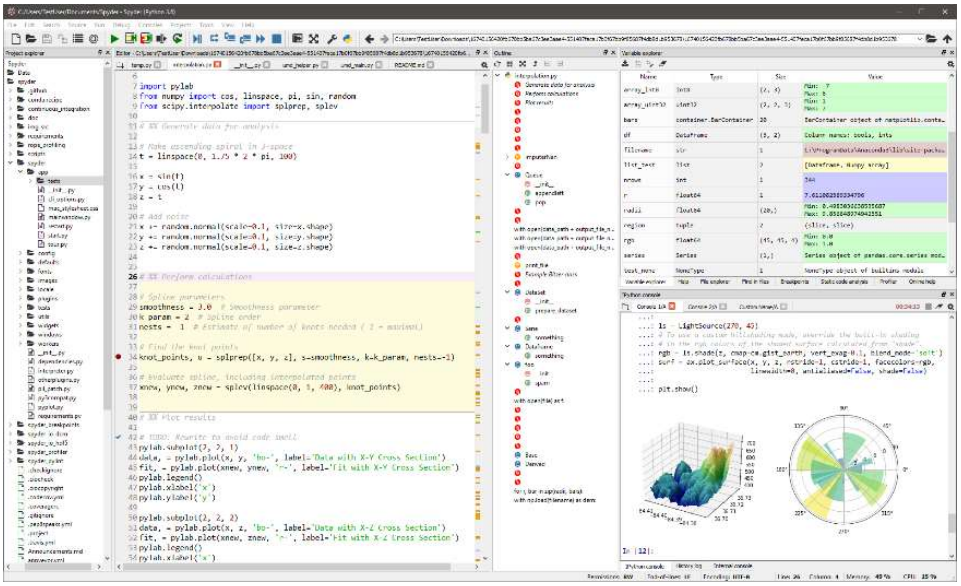

Figura 9 - Interfaccia dell'IDE Spyder

Al centro vi è l'editor di testo, caratterizzato da una colorazione della sintassi personalizzabile che facilita la lettura degli script; è presente l'analisi del codice automatica, che avverte in caso di errori sintattici, e la funzionalità di completamento automatico del codice, che ne velocizza ulteriormente la stesura.
A destra, in alto, è presente un variable explorer, una delle funzionalità più utili fra quelle offerte da Spyder e raramente supportato in altri IDE. Consiste in un'interfaccia che permette di interagire con le variabili del proprio codice, le quali possono essere analizzate ed eventualmente modificate in modo dinamico e separato rispetto all'esecuzione del programma. In basso è invece localizzata una console IPython, ovvero una shell interattiva che permette l'esecuzione di righe di codice e la visualizzazione dei risultati. Un'altra caratteristica importante di Spyder è la presenza di un debugger integrato, le cui funzionalità si attivano mediante i bottoni blu della toolbar; consente di eseguire il codice un'istruzione alla volta mediante l'inserimento di breakpoint e di interagire con il programma stesso attraverso la console IPython.

4.5.2 Android Studio

Android Studio è l'ambiente di sviluppo integrato ufficiale per il sistema operativo Android e presenta dei plugin che consentono la creazione di progetti Flutter e Dart. Basato sul software prodotto dall'azienda JetBrains IntelliJ IDEA, Android Studio mette a disposizione moltissime funzionalità che supportano lo sviluppo di applicazioni mobile.
L'editor di testo, innanzitutto, facilita la scrittura delle varie componenti con la colorazione della sintassi e meccanismi di completamento delle parole, fornendo inoltre funzionalità di refactoring automatico del codice; è presente, inoltre, un emulatore virtuale che permette di eseguire le proprie applicazioni direttamente all'interno di Android Studio. L'IDE supporta poi un sistema di controllo di versione del codice dettagliato ed efficiente, che permette di recuperare eventuali modifiche apportate nel corso del tempo, e implementa l'integrazione con il sistema di versioning online GitHub direttamente all'interno della piattaforma.

4.5.3 PgAdmin e Db Browser for Sqlite

PgAdmin e DB Browser for Sqlite sono due software, entrambi scritti in C++, che forniscono delle interfacce grafiche facili e intuitive per l'interazione con database di tipo PostgreSQL il primo e SQlite il secondo.
Entrambi implementano tutte le funzionalità di base per la gestione di un database, come la creazione e l'eliminazione di istanze, l'inserimento e la manipolazione di tabelle, oltre all'esecuzione di query SQL.

PgAdmin consente inoltre il collegamento con un database remoto, una funzione che è stata utilizzata nell'ambito del sistema di monitoraggio per accedere all'istanza PostgreSQL attivata tramite il servizio Amazon RDS.

4.5.4 FileZilla e AnyDesk

FileZilla e AnyDesk sono software che sono stati adoperati per interagire con le macchine remote; il primo, in particolare, per inviare file al server tramite il protocollo SFTP, mentre il secondo per accedere al computer situato nei siti di monitoraggio ed avviare il programma che effettua le misure.

FileZilla è open source, multipiattaforma e a licenza libera: supporta protocolli come l'FTP e l'SFTP per il trasferimento di file in rete. L'interfaccia consiste in una sezione dedicata ai messaggi che vengono ricevuti dal server cui si sta cercando di connettersi e da due finestre, rappresentanti i file system dei due sistemi, tramite le quali- è possibile effettuare il trasferimento dei file tramite drag and drop. La connessione ai server è facilitata grazie al site manager, una funzionalità che consente la creazione di una lista personalizzata di siti FTP facilmente selezionabili tramite un menu a tendina.

AnyDesk è invece un software proprietario, distribuito dall'azienda tedesca AnyDesk Software GmbH. Consente l'accesso a macchine remote con qualsiasi sistema operativo su cui sia installato lo stesso AnyDesk. Sfrutta la codifica video DeskRT, implementata dalla stessa azienda, la quale permette la trasmissione di immagini e video compressi che risultano fluidi anche con connessioni a banda stretta. Tutte le connessioni sono criptate attraverso l'algoritmo AES a 256 bit e sfruttano il protocollo sicuro TLS 1.2.

CAPITOLO 5

Software sviluppati

L'ultima fase dello sviluppo dei software che gestiscono il sistema di monitoraggio è consistita nell'effettiva stesura dei codici che implementano le varie componenti dell'architettura. La programmazione vera e propria è il momento cruciale e più complesso dell'intero processo di produzione, ma il lavoro di analisi preliminare descritto nei capitoli precedenti, che ha incluso la definizione dettagliata delle funzionalità da implementare e delle proprietà che devono caratterizzare i software, oltre che la scelta di strumenti opportuni con cui realizzarli, ha indirizzato lo sviluppo sui giusti binari.

Il primo passo è stato quello di definire in modo puntuale i vari moduli che formano l'architettura software, dettagliando funzionalità e protocolli di comunicazione. Le componenti del sistema sono state in realtà definite precedentemente, come illustrato nel primo paragrafo del capitolo quattro, con l'intento di individuare gli strumenti software e i linguaggi di programmazione necessari allo sviluppo. In questa fase si è tuttavia scesi ad un livello di dettaglio superiore, tale da permettere l'effettiva implementazione dei codici; quest'ultima ha seguito il seguente ordine:

1. Software che effettua le misure e salva i risultati sui database.
2. Server che interroga il database e fornisce i dati per la visualizzazione.
3. Applicazione mobile che realizza l'interfaccia.

La scelta di procedere in questo modo è stata dettata da due motivi: innanzitutto, vi era necessità di iniziare il monitoraggio il prima possibile, sia per poter cominciare ad analizzare dei dati ma anche per verificare che non vi fossero problematiche di alcun tipo, specialmente lato sensori e schede di acquisizione. Per tale ragione, era fondamentale avviare per prima la realizzazione del software che opera effettuando le misure e interagisce quindi con la parte hardware del sistema. In secondo luogo, le tre componenti dell'architettura dipendono, rispetto al modo in cui sono state presentate nella lista, dal software superiore, non da quello inferiore; ad esempio, le funzionalità del server possono essere implementate solo se il database contiene dei dati, quindi necessitano che il software che effettua le misure sia attivo e funzionante, mentre quest'ultimo non ha bisogno di altre componenti per essere eseguito in modo funzionale. Di conseguenza, nell'ottica di procedere con uno sviluppo consequenziale, è questo l'ordine naturale che si delinea analizzando le funzionalità dei vari moduli che compongono il sistema.
In questa sezione, dopo aver descritto più puntualmente l'architettura software, verranno presentati i vari programmi sviluppati per il sistema di monitoraggio; saranno in particolare analizzati gli algoritmi e le strutture dati che permettono l'implementazione delle funzionalità richieste.

5.1 Architettura del sistema

L'architettura software che gestisce il sistema di monitoraggio è caratterizzabile, come già presentato nel capitolo precedente, in due livelli. Il primo è quello che si occupa di effettuare le misure e salvare i dati nei database, mentre il secondo è quello che implementa l'interfaccia per la visualizzazione dei risultati del monitoraggio. La distinzione è ovviamente teorica, in quanto le funzionalità dei due strati sono strettamente correlate e coordinate. In particolare, il punto di contatto è costituito dal database remoto, in cui i dati vengono immagazzinati dal primo livello ed estratti dal secondo. Tuttavia, in virtù dell'asincronismo che caratterizza i loro processi, può essere funzionale presentarli ed esaminarli separatamente, specialmente in fase di analisi; le misure vengono infatti effettuate in modo costante, in particolare ogni ora, mentre il processo di visualizzazione dei dati si verifica ogni qualvolta un utente ne faccia richiesta attraverso l'applicazione mobile.

5.1.1 Primo livello

Il primo livello dell'architettura provvede all'esecuzione delle misurazioni di impedenza elettrica e di temperatura dei blocchi di calcestruzzo. Tale processo consiste di due parti; è necessario, infatti, interagire prima con le schede di acquisizione per realizzare le misure e collegarsi poi ai due database, uno locale e uno situato in remoto, per immagazzinare i dati raccolti.
Per quanto riguarda la scheda che gestisce le misure di impedenza, la AD5940BIOZ, vengono sfruttate le API del software SensorPal, che consentono di azionare direttamente la scheda, e quindi i sensori, per realizzare la misurazione e ottenere poi il risultato. Relativamente all'interazione con l'Arduino Nano, il quale è collegato al sensore di temperatura, viene invece effettuata una lettura sul collegamento seriale individuato dalla porta COM virtuale assegnata alla scheda stessa, in modo da leggere direttamente il valore misurato dal sensore. Il procedimento è riassunto in figura 10, mentre il codice Python che implementa queste interazioni verrà presentato nella sezione dedicata.

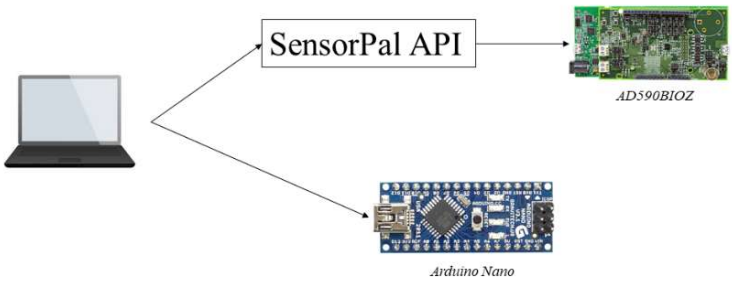

Figura 10 – Interazione con le schede di acquisizione

Una volta raccolti, i valori vengono inviati ai due database; quello remoto, un'istanza PostgreSQL, viene interrogato poi dal server, appartenente al secondo livello dell'architettura, per estrarre i dati da inviare all'interfaccia del sistema di monitoraggio, mentre quello locale, realizzato con il DBMS SQlite, ha la finalità di costituire un backup sicuro e completo dei dati, soprattutto nel caso in cui non avvenga il collegamento con il database remoto. La figura 11 esemplifica la seconda parte del primo livello dell'architettura.

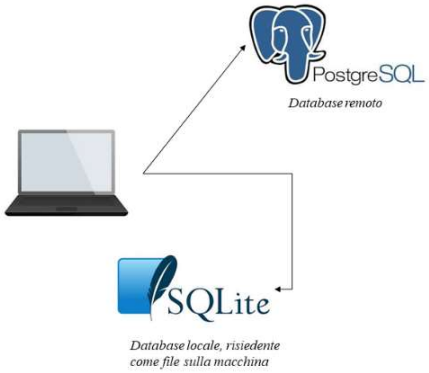

Figura 11 - Collegamento con i database

Il processo descritto si verifica per tutti i blocchi di calcestruzzo, in entrambe le installazioni, una volta ogni ora; l'automatizzazione avviene mediante un'applicazione, chiamata Utilità di pianificazione, la quale permette di pianificare l'esecuzione di programmi a determinati orari. Si trova preinstallata all'interno del sistema operativo Microsoft Windows, il quale viene utilizzato sulle due macchine, situate presso i siti di monitoraggio, per garantire la compatibilità con le API di SensorPal.

5.1.2 Secondo livello

Il secondo livello dell'architettura è sostanzialmente costituito da due software, il server e l'applicazione mobile, i quali implementano l'interfaccia tramite cui gli utenti possono visionare i risultati del monitoraggio. Come già descritto, i dati vengono recuperati dal database remoto attraverso il server ogni volta che si fa una richiesta di visualizzazione e vengono poi inviati all'applicazione mobile, la quale si occupa di gestirne la visualizzazione in forma grafica. La comunicazione fra le 3 componenti è rappresentata in figura 12, la quale mostra i passaggi che sono coinvolti in una richiesta da parte dell'utente.

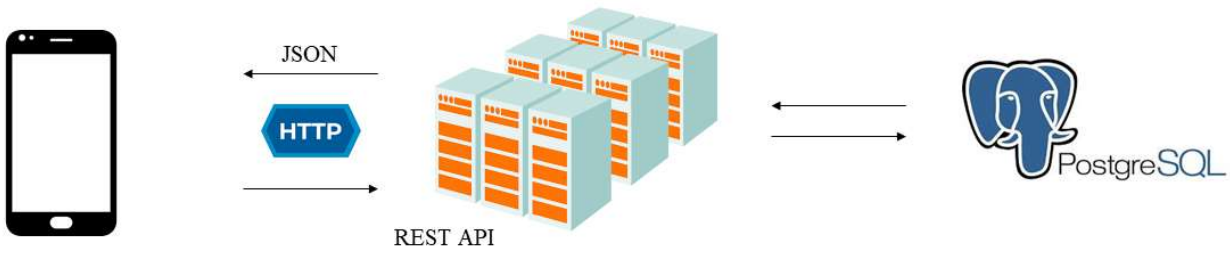

Figura 12 - Secondo livello dell'architettura

Il server espone un servizio di REST API, verso cui l'applicazione mobile, che funge da client, effettua una chiamata attraverso il protocollo HTTP specificando i parametri che delineano i dati richiesti dall'utente. Il server processa la richiesta, interroga il database e risponde con un JSON contenente i dati desiderati, pronti per essere visualizzati tramite grafico all'interno dell'applicazione. A questo proposito, il servizio annovera due endpoint, uno per la visualizzazione dei dati così come sono estratti dal database e l'altro per ottenere invece i dati filtrati attraverso una media mobile di una finestra temporale personalizzabile, così come richiesto dalle specifiche del progetto. Inoltre, sempre nell'ottica di implementare tutti i requisiti funzionali descritti nel capitolo 3, è presente un terzo endpoint, attraverso il quale l'utente può richiedere i dati grezzi in modo da poterli analizzare e studiare a parte, senza bisogno di visionarne l'andamento attraverso un grafico. La procedura, raffigurata in figura 13, è equivalente a quella mostrata in figura 12, ma il server, invece di rispondere al client con un JSON contenente i dati richiesti, invia, attraverso il protocollo SMTP, un file CSV contenente i dati estratti dal database alla mail dell'utente, la quale viene specificata in fase di login nell'applicazione e inviata al server contestualmente alla richiesta. Il client riceve comunque una risposta in cui viene notificato l'esito dell'operazione, sia positivo che negativo.

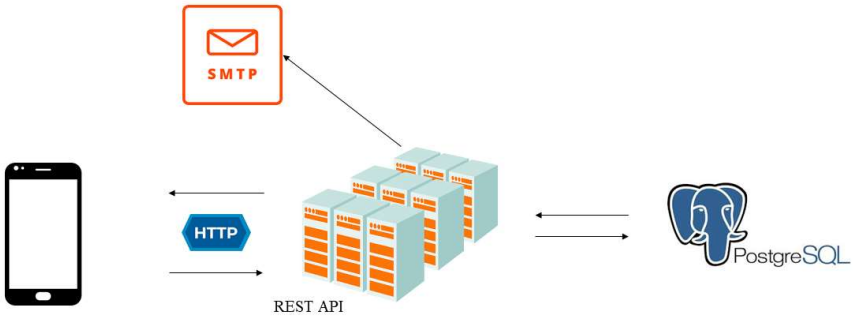

Figura 13 - Terzo endpoint del server

5.1.2.1 REST API

Il termine REST, acronimo di Representational State Transfer, è stato introdotto nel 2000 dalla tesi di dottorato dell'informatico Roy Fielding, nella quale viene descritto come un particolare stile architetturale implementabile nelle comunicazioni Web basate su HTTP. REST non è infatti un protocollo, ma un insieme di specifiche applicabili ad un'architettura software che opera sul Web per renderla scalabile e più efficiente. Fra queste, vi è ad esempio l'approccio Client-Server, che si fonda sulla separazione dei ruoli fra le varie componenti del sistema, e il

principio dell'uniformemente accessibile, basato sul rendere ogni risorsa individuabile da un'interfaccia comune ma caratterizzata da un indirizzo univoco, favorendo in questo modo la modularità e la manutenibilità del sistema.
Una REST API, anche detta API RESTful, è quindi un'API realizzata seguendo i principi REST. Ogni richiesta, come tutte quelle eseguite su HTTP, è inoltre caratterizzata da un verbo, o metodo, che caratterizza l'operazione che si vuole eseguire sulla risorsa individuata tramite lo specifico URL. Le API del sistema di monitoraggio supportano solo il metodo GET, in quanto in tutti i casi, come precedentemente descritto, il client, ovvero l'applicazione mobile, fa richiesta di dati da visualizzare, non per cancellarli, modificarne il valore o aggiungerne altri.

5.1.2.2 JSON

JSON, acronimo di JavaScript Object Notation, è un formato di dati impiegato nello scambio di informazioni fra servizi WEB. Si caratterizza per essere semplice da leggere e scrivere per le persone e altrettanto facile da realizzare e analizzare per le macchine.
Basato sulla sintassi del linguaggio JavaScript, è in realtà indipendente dal linguaggio di programmazione utilizzato; tutti i principali ne supportano infatti l'utilizzo, attraverso librerie o funzionalità integrate che implementano la gestione del parsing e della serializzazione.
JSON si basa su due strutture:

- L'oggetto, delimitato da parentesi graffe, è costituito da un insieme di coppie chiave valore, separate da virgole.
- L'array, il corrispettivo della struttura dati presente in tutti i linguaggi di programmazione, ovvero un insieme ordinato di elementi, codificato attraverso valori separati da virgole e delimitato da parentesi quadre.

I tipi di dato ammessi in JSON sono: le stringhe, ovvero sequenze di caratteri, i numeri, sia interi che decimali, valori booleani, quindi True o False, e, infine, il tipo null, che rappresenta un valore sconosciuto o indefinito.
Il sistema di monitoraggio utilizza la codifica JSON per inviare i dati dal server all'applicazione mobile. Un esempio di JSON scambiato è quello riportato di seguito in figura 14; la creazione verrà analizzata poi nella sezione relativa all'implementazione del server.

```
{
  "measures": [
        {
            "datetime": "20201008 13:55:20",
            "magnitude": 36279.79
        },
        {
            "datetime": "20201008 14:55:21",
            "magnitude": 36280.7
        },
        {
            "datetime": "20201008 15:55:21",
            "magnitude": 36140.78
        }
      ]
}
```


Figura 14 - Esempio di JSON

I JSON prodotti dal server sono caratterizzati da un oggetto contenente una coppia chiave valore, rispettivamente la stringa *measures* e un array, i cui elementi, a loro volta degli oggetti, rappresentano tutte le misure avvenute nell'intervallo temporale specificato contestualmente alla richiesta, in questo caso il giorno 08/10/2020 tra le 13:00 e le 16:00. All'interno di ogni elemento vi sono le informazioni sulla misura, organizzate anch'esse in coppie chiave valore: viene riportato il momento in cui è avvenuta la misurazione, specificato in data e ora attraverso una stringa, e il valore dell'attributo richiesto, in questo esempio il modulo dell'impedenza elettrica.

5.2 Database

I database costituiscono uno dei componenti chiave dell'architettura del sistema, in quanto consentono lo storage sicuro e facilmente accessibile dei dati ricavati dal monitoraggio dei blocchi di calcestruzzo. Come già descritto, due sono le tipologie di istanze utilizzate; in particolare, entrambe le macchine situate presso i siti di monitoraggio gestiscono un'istanza SQlite, conservata come file locale presso le macchine stesse, mentre il database remoto, un'istanza PostgreSQL, è comune alle due installazioni.

Sia SQlite che PostgreSQL sono DBMS basati sul modello relazionale, il più diffuso e utilizzato nell'ambito della gestione dei dati, che consiste nel rappresentare l'informazione attraverso delle tabelle, che rappresentano relazioni tra i dati. Ogni tabella è caratterizzata da un insieme di colonne, che costituiscono gli attributi dei dati che si vogliono salvare, e da un insieme, non ordinato, di righe, anche dette record, ovvero le tuple della relazione.

Il sistema di monitoraggio, in totale, consta quindi di tre database, tutti caratterizzate da tabelle aventi gli stessi attributi; ciò che varia sono ovviamente i dati che vengono salvati. In particolare:

- Le istanze locali contengono un'unica tabella in cui vengono salvati i dati relativi alle misure effettuate presso il sito in cui si trovano.
- Il database remoto è costituito da due tabelle, una per l'installazione di Gijon e l'altra per quella di Leon, le quali risultano identiche, sia a livello di attributi che di dati, a quelle locali, a meno di eventuali perdite di informazione.

Prima di analizzare la struttura della tabella, comune a tutti i database, è necessario comprendere in che modo avvengono le misure, in modo da delineare le ragioni che hanno determinato l'inserimento di ciascun attributo.

5.2.1 Esecuzione delle misure

All'interno di questo elaborato si è fatto riferimento al concetto di misura intendendola come una singola interazione con le schede di acquisizione. Ciò avviene nel caso dell'Arduino Nano, collegato al sensore di temperatura, ma non in quello delle schede AD5940BIOZ, responsabili delle misurazioni di impedenza, per le quali il valore effettivamente salvato nel database, sottoforma di modulo e fase, è la media ponderata di una serie di misure eseguite una dopo l'altra per un numero predefinito di volte, fissato ad 80. In aggiunta a ciò, ogni volta che il programma viene avviato, vengono in realtà effettuate 20 misure di impedenza su ogni scheda, ognuna caratterizzata, quindi, da 80 effettive interazioni con i sensori, ma eseguite a frequenze diverse, che vanno da 1 a 20 kHz con passo di 1 kHz.
I dati, tuttavia, prima di essere effettivamente inseriti nel database, subiscono un'elaborazione volta ad eliminare gli outlier dall'insieme degli 80 valori che vengono generati in una misurazione, sia per quanto riguarda il modulo che per la fase. Il termine outlier, in ambito statistico, indica un valore che, all'interno di un insieme, risulta lontano dagli altri; nel caso del sistema di monitoraggio, un outlier può essere dunque considerato una misurazione non corretta, o quantomeno imprecisa, in quanto numericamente lontana dalle altre. Di conseguenza, il valore effettivamente inserito nel database è la media ponderata dei valori delle singole misurazioni da cui sono però stati esclusi gli outlier. Tale processamento viene effettuato tramite il metodo denominato "2 sigma", che consiste nell'eliminare i valori che siano distanti dalla media, calcolata su tutto il set di misurazioni, più del doppio della deviazione standard, in valore assoluto. Riassumendo, ogni ciclo di misurazioni comporta la creazione di 20 nuovi record da inserire nel database per ogni scheda di acquisizione. Ognuno di questi è caratterizzato da una frequenza, dai valori di modulo e fase, ottenuti come media ponderata dell'insieme delle 80 misurazioni e filtrato tramite l'eliminazione degli outlier, e da un valore di temperatura, ottenuto interagendo serialmente con la scheda Arduino.

5.2.2 Struttura della tabella

Nella tabella 1 sono riportati gli attributi che compongono la tabella del database e il tipo di dato corrispondente.

Nome	Tipo di dato
id	stringa
com_port	stringa
date	stringa
time	stringa
frequency	intero
magnitude	reale
magnitude_devstd	reale
magnitude_mean_outlier	reale
magnitude_devstd_outlier	reale
phase	reale
phase_devstd	reale
phase_mean_outlier	reale
phase_devstd_outlier	reale
temperature	reale
samples_mean	intero

Tabella 1 - Attributi delle tabelle

Il DBMS SQlite supporta esclusivamente cinque tipologie di dato: il tipo nullo, per indicare un dato non definito, i tipi intero e reale, per gestire i dati numerici, il tipo stringa, per inserire valori testuali, e il tipo blob, utilizzato per descrivere un dato generico. Di conseguenza, nell'ottica di garantire compatibilità fra i due database utilizzati, è stato deciso di ricorrere solamente a queste tipologie di dato, anche per quanto riguarda l'istanza PostgreSQL, la quale ne permette in realtà un set molto più ampio. Fra quelli inclusi nativamente, ad esempio, vi sono i tipi data e timestamp, che avrebbero potuto codificare, nella tabella del sistema, gli attributi date e time, i quali sono stati invece considerati come semplici stringhe.
Per quanto riguarda gli attributi:

- Il campo id contiene il nome della scheda di acquisizione AD5940BIOZ che ha eseguito la misurazione. È una stringa, composta da tre parti e separate da un trattino basso: la prima è una sigla che identifica l'installazione, quindi "TU" o "MA" a seconda che si tratti del sito di monitoraggio localizzato presso Leon o di Gijon, la seconda è un

numero che individua il blocco di calcestruzzo e la terza è invece una stringa, composta un carattere e un numero, che identifica la scheda stessa.

- L'attributo com_port contiene l'informazione relativa alla porta COM cui è collegata la scheda di acquisizione.
- Gli attributi date e time rappresentano, rispettivamente, il giorno e l'ora in cui è avvenuta la misura.
- Il campo frequency contiene la frequenza in cui è avvenuta la misurazione.
- Gli attributi magnitude, phase e temperature rappresentano il modulo, la fase e la temperatura misurati al momento dell'acquisizione. Sono i valori che vengono effettivamente recuperati dal server e inviati all'applicazione mobile per visualizzare l'andamento del monitoraggio. Come descritto precedentemente, magnitude e phase sono il risultato della media ponderata calcolata sui valori delle singole misurazioni da cui sono però stati eliminati gli outlier per ottenere un dato meno rumoroso.
- I campi magnitude_devstd, magnitude_mean_outlier e magnitude_devstd_outlier contengono dei valori che non sono strettamente necessari alla visualizzazione o all'analisi dei risultati del monitoraggio, ma rappresentano dei parametri statistici, relativi al modulo dell'impedenza. Questi dati possono essere utilizzati, in un'ottica a lungo termine, per valutare l'accuratezza e le prestazioni del sistema di monitoraggio nel complesso, in particolare della parte hardware, quindi dei sensori e delle schede di acquisizione. Nel primo attributo viene salvata la deviazione standard dell'insieme di misure da cui sono stati eliminati gli outlier, mentre gli altri due contengono rispettivamente la media ponderata e la deviazione standard calcolate su tutte le 80 misurazioni effettuate, quindi compresi gli outlier. Questi campi sono stati inseriti per adempiere alle specifiche delineate nei requisiti funzionali, in particolare in quello relativo allo storage dei dati, che richiedeva la presenza di valori che permettessero appunto l'analisi delle performance e della precisione del sistema dopo un arco di tempo ragionevolmente lungo. La tabella presenta inoltre gli stessi attributi anche relativamente alla fase, in particolare phase_devstd, phase_mean_outlier e phase_devstd_outlier.
- L'attributo samples_mean rappresenta, infine, il numero di misure che rimangono dopo aver eliminato gli outlier.

5.3 Software per le misure

L'esecuzione delle misure viene effettuata tramite uno script Python, denominato measurements.py, che viene mandato in esecuzione ogni ora tramite l'applicazione Windows Utilità di

Pianificazione, la quale consente appunto di avviare programmi a determinati orari, anche in modo ricorrente.
Il codice è organizzato secondo dei blocchi funzionali che svolgono funzioni diverse e che vengono eseguiti in cascata, uno dopo l'altro. Tutte le parti sono necessarie al corretto funzionamento del programma, ma questa strutturazione consente una gestione degli errori tale da permettere che il verificarsi di un'eccezione in uno dei blocchi non pregiudichi la prosecuzione degli altri blocchi dello script, non annullando dunque l'intero set di misurazioni.
I blocchi di codice individuabili sono tre: il primo si occupa di stabilire le connessioni al database, il secondo gestisce l'interazione con le schede di acquisizione, il terzo effettua invece l'elaborazione sui dati descritta precedentemente e li invia infine ai due database, sia a quello situato in locale che a quello remoto.

5.3.1 Connessione ai database

Il programma, appena avviato, si connette ai due database. Per quanto riguarda l'istanza SQlite, quindi il database locale, la connessione avviene tramite la funzione connect, fornita dalla libreria Sqlite3, che accetta come parametro il nome del file, con estensione .db, su cui è localizzato il database e restituisce un oggetto di tipo connessione tramite cui è possibile interrogare la base di dati. La connessione viene effettuata un'unica volta all'inizio dell'esecuzione e viene poi chiusa al termine.
Il collegamento con il database remoto viene gestito diversamente, in quanto la connessione viene aperta e poi chiusa ogni volta che si interagisce con una nuova scheda, quindi più di una volta ad ogni esecuzione del codice, tante quante sono le schede presenti nel sito di monitoraggio. Il processo di acquisizione delle misure è infatti abbastanza lungo e risulta più funzionale e sicuro effettuare più volte la procedura di connessione, assicurando così che eventuali problemi sopraggiunti durante l'esecuzione del codice, come, ad esempio, la perdita di connessione alla rete Internet, non pregiudichino, perlomeno necessariamente, l'invio dei dati relativi alle schede successive al database.
La figura 15 contiene la porzione di codice che implementa la connessione al database remoto. Le prime tre righe sono eseguite esclusivamente una volta, mentre le restanti sono quelle che vengono ripetute ogni volta che si interagisce con una scheda diversa.

```
session = boto3.Session(profile_name='endurcrete')
client = boto3.client('rds')
token = client.generate_db_auth_token(DBHostname=ENDPOINT,
                                      Port=PORT,DBUsername=USR, Region=REGION)

try:
    conn_postgre = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME,
                                    user=USR, password=token)
    logging.info("Connected to PostgreSQL database")
except Exception as e:
    logging.error("Database connection failed due to {}".format(e))
```

Figura 15 - Connessione al database remoto

Il codice riportato utilizza le funzioni fornite dalle librerie boto3 e psycopg2; quest'ultima, descritta nel capitolo 4, è quella che implementa la funzione connect, responsabile dell'effettiva connessione al database PostgreSQL; similmente alla funzione omonima definita dal modulo SQlite3, restituisce un oggetto di tipo connessione, il quale fornisce dei metodi per interrogare e manipolare il database.

La libreria boto3, sviluppato da Amazon Web Services per semplificare l'integrazione dei loro servizi all'interno di codice scritto in Python, viene sfruttata invece per generare un token di autenticazione che viene poi passato come parametro alla funzione connect per eseguire correttamente la connessione. Gli altri parametri forniti, quali ENDPOINT, PORT, DBNAME e USR, contengono le informazioni relative al database e necessarie per stabilire la connessione, in particolare l'url e la porta del server AWS, il nome del database e lo username dell'utente che interagisce con la base di dati.

La gestione delle eccezioni viene effettuata tramite il costrutto definito dalle parole chiave try ed except; la libreria logging, invece, viene sfruttata per effettuare debug del codice in caso di errori che ne blocchino il funzionamento. Consente, infatti, di registrare su un file separato le informazioni volute, associando ad ogni messaggio un livello che identifica il contesto del log; i livelli info ed error, utilizzati nel codice, individuano, rispettivamente, un'informazione che si vuole memorizzare e un errore che ha impedito l'esecuzione di una determinata istruzione.

5.3.2 Esecuzione delle misure

Il secondo blocco del programma è quello che si occupa di effettuare le misure, quindi, nella pratica, di collegarsi alle schede di acquisizione, avviare la misurazione e salvare i risultati all'interno di strutture dati che saranno poi processate e inviate al database attraverso le funzionalità che verranno descritte nella sezione successiva.

Il codice si basa su 4 strutture dati, in particolare su 4 array di stringhe:

- panels_TU (o panels_MA, in base al sito considerato) contiene i nomi delle schede di acquisizione AD5940BIOZ.
- COM_ports_imp contiene le porte COM tramite cui interagire con le schede di acquisizione dell'impedenza.
- COM_ports_temp contiene le porte COM virtuali in cui sono collegati gli Arduino Nano.
- frequency contiene invece tutte le frequenze cui si vogliono effettuare le misurazioni, quindi da 1000 a 20000 con un passo di 1000.

I vettori panels_TU, COM_ports_imp e COM_ports_temp hanno la stessa lunghezza, pari al numero delle schede di acquisizione di impedenza presenti nello specifico sito di monitoraggio. Sebbene le schede Arduino siano in numero inferiore rispetto alle schede AD5940BIOZ e le corrispettive porte COM siano, di conseguenza, in quantità minore, l'array COM_ports_temp contiene comunque lo stesso numero di elementi degli altri vettori, in quanto il riferimento alla porta COM dell'Arduino situato in un blocco è ripetuto più volte in corrispondenza delle schede di acquisizione di impedenza collocate nello stesso campione.
L'esecuzione del codice si articola in due iterazioni, una innestata nell'altra: la prima viene effettuata sull'array panels_TU ed esamina quindi tutte le schede di acquisizione, la seconda itera invece sull'array frequency, determinando, ad ogni ciclo, una frequenza diversa. Ogni volta che viene conclusa l'iterazione interna i dati vengono inviati al database; in questo modo, ad ogni esecuzione del programma corrisponde l'inserimento nelle tabelle di 20 record per ogni scheda di acquisizione, ognuno contenente i dati relativi ad una misurazione effettuata ad una frequenza diversa.
La misura della temperatura viene eseguito all'inizio dell'iterazione esterna ed avviene dunque una sola volta per ogni scheda di acquisizione di impedenza. È infatti supposto che la temperatura del blocco resti la stessa durante la misurazione alle diverse frequenze.

5.3.2.1 Temperatura

La misurazione della temperatura consiste in una semplice lettura seriale sulla porta COM associata alla scheda Arduino. Il codice che implementa questa funzionalità è riportato in figura 16.

```
def measure_temp(com_port):
        serial_Arduino = serial.Serial(com_port,9600)
        temperature = serial_Arduino.readline()
        serial_Arduino.close()
        return temperature

Temperatures = []
for _ in range(3):
        try:
            temperature = func_timeout(15,measure_temp,args=[COM_ports_temp[i]])
            temperature = float(temperature.decode())
        except Exception as e:
            logging.error("Couldn't connect to Arduino due to {}".format(e))
            temperature = 0

        Temperatures.append(temperature)

temperature = max(Temperatures)
```

Figura 16 - Misurazione della temperatura

La problematica di interagire con la scheda Arduino è determinata dal fatto che numerose volte la connessione con il sensore si blocca e, fino al momento in cui viene ripristinata, anche semplicemente chiudendo e riaprendo la connessione con la porta COM, l'esecuzione della lettura seriale non si conclude, senza però che questa attesa generi eccezioni rintracciabili con un costrutto try except. Di conseguenza, è necessario un meccanismo tramite cui stabilire un tempo massimo oltre il quale chiudere la connessione e riprovare ad aprirla. A tal proposito, è stato sfruttato il modulo python func_timeout, da cui è stata importata la funzione omonima che viene utilizzata nel codice.

Partendo dall'inizio, la misura della temperatura viene effettuata sulla base di un ciclo composto da tre iterazioni, generato grazie alla funzione Python range; ogni iterazione corrisponde all'esecuzione di una misura di temperatura. I valori risultanti vengono aggiunti, di volta in volta, ad un array, e il valore finale, che verrà poi inserito nel database, è ricavato calcolando il massimo fra i tre che sono stati ottenuti. Nella maggior parte dei casi, verosimilmente, i valori sono identici, ma la scelta di considerare il massimo deriva dalla volontà di assegnare il valore 0, considerato di default, alle misure non andate a buon fine.

La funzione func_timeout stabilisce una soglia massima di tempo oltre la quale l'esecuzione di una funzione, passata come parametro, viene interrotta generando un'eccezione. Gli altri parametri di cui necessita sono un intero, ovvero il numero di secondi oltre il quale far scattare il timeout, e gli argomenti da passare alla funzione. La funzione func_timeout, inoltre, ha come valore di ritorno il dato restituito dalla funzione che viene monitorata, nel caso del codice la funzione measure_temp, la quale si occupa appunto di effettuare la lettura seriale sulla porta COM, passatale come parametro, e che restituisce il valore di temperatura rilevato. La variabile

i è quella tramite cui vengono effettuate le iterazioni sulle schede di acquisizione, quindi la stringa COM_ports_temp[i] rappresenta la porta COM cui è collegato l'Arduino Nano corrispondente alla scheda di acquisizione di impedenza corrente. Il valore di temperatura, tuttavia, prima di essere inserito nella struttura dati Temperatures, deve essere convertito in un numero reale. A tal proposito, dato che il valore viene restituito dalla lettura seriale come un tipo byte, viene applicato il metodo decode, che lo converte in stringa, e la funzione float, che lo converte appunto in un tipo float.

5.3.2.2 Impedenza

Le misurazioni di impedenza vengono effettuate sfruttando le funzionalità offerte dalle API di SensorPal, il software che consente di controllare direttamente le schede di acquisizione.

Il primo passo consiste dunque nell'importare il codice assembly che implementa tali API, localizzato in un file presente nella stessa cartella dell'eseguibile. Vengono utilizzate le librerie winreg e pythonnet: la prima consente di recuperare il percorso del file desiderato interagendo con il registro di sistema, ovvero la base di dati gestita dal sistema operativo che contiene tutte le informazioni sulle applicazioni installate, tra cui SensorPal, mentre la seconda fornisce delle funzionalità che permettono l'integrazione di codice C#, utilizzato per realizzare le API, all'interno di software Python.

Questa procedura viene eseguita all'avvio del programma e fornisce al software una classe, denominata SensorPalAPI, che contiene i metodi necessari per controllare direttamente le schede di acquisizione, azionando le misure e salvandone i risultati, come se si stesse interagendo direttamente con l'interfaccia grafica del software.

Analogamente a come accade quando si utilizza l'interfaccia, è possibile impostare dei parametri che determinano le modalità di esecuzione della misura. Tale predisposizione avviene tramite una struttura dati JSON, memorizzata su un file locale e contenente tutte le informazioni necessarie a definire la misura.

Il sistema di monitoraggio effettua tutte misure analoghe, che si differenziano unicamente per il valore della frequenza. Il parsing del JSON viene dunque effettuato all'inizio del programma, ma ad ogni iterazione sull'array frequency viene aggiornato il parametro relativo alla frequenza, in modo da effettuare, ogni volta, una misura ad una frequenza diversa. Tutti gli altri parametri sono definiti in modo statico e restano costanti durante l'esecuzione del programma.

Di particolare rilevanza, vi è il valore relativo alla chiave "technique", che determina la misura che si vuole effettuare, fissato a "Body Impedance", il valore assegnato alla chiave "plot",

ovvero “Body Z (Mag+Phase)”, che individua una misura di modulo e fase dell’impedenza, e il numero di misure che si vogliono effettuare, fissato, come descritto precedentemente, a 80. Dopo aver istanziato l’oggetto di tipo SensorPalAPI, assegnandogli, contestualmente alla definizione, i parametri che determinano le modalità di misurazione, e aver aperto la connessione con la porta COM corrispondente alla scheda di interesse, viene richiamato il metodo measure, definito internamente alla classe SensorPalAPI, che avvia la misura di modulo e fase dell’impedenza sulla scheda di acquisizione. I dati rilevati vengono poi recuperati grazie al metodo getGraphData, anch’esso proprio della classe SensorPalAPI, il quale popola due array, chiamati Magnitudes e Phases, con i valori corrispondenti; queste strutture dati, che risultano di 80 elementi, saranno poi processate per ottenere il record da inserire nel database.

5.3.3 Salvataggio dei dati

La procedura di elaborazione dei dati viene effettuata a partire dagli array Magnitudes e Phases, contenenti gli 80 valori ottenuti dalla misurazione di impedenza, ed è riportata in figura 17; dopo aver eseguito i calcoli opportuni, i dati vengono inviati ai due database. Se le misure non sono andate a buon fine, tuttavia, le due strutture dati non vengono riempite e l’esecuzione del codice genera degli errori; a tal proposito, prima di effettuare il processing, viene eseguito un controllo sulla lunghezza degli array. Nel caso in cui risultino vuoti, non viene eseguito alcun tipo di elaborazione, ma viene direttamente salvato il record nel database, assegnando il valore 0, anche in questo caso considerato di default, a tutti i campi relativi al modulo e alla fase. Vi è poi un algoritmo, lato server, che si occupa di gestire correttamente l’eventuale presenza di zeri nei dati da fornire all’interfaccia grafica per essere visualizzati.

Il processo inizia con la definizione di tre array; il primo, vect, contiene i due array Magnitudes e Phases, e viene utilizzato per iterare su tali strutture dati, mentre gli altri due, dev_std e mean, verranno riempiti con i valori della deviazione standard e della media calcolati sugli 80 valori ottenuti dalle misurazioni, sia per il modulo che per la fase.

```
vect = [Magnitudes,Phases]
devstd =[]
mean = []
for v in vect:
        v_mean = np.mean(v)
        v_dev = np.std(v)

        mean.append(v_mean)
        devstd.append(v_dev)

        index1 = np.where( v > v_mean +  2*v_dev )

        while len(index1[0]) != 0 :
            vect[0].pop(index1[0][0])
            vect[1].pop(index1[0][0])
            index1 = np.where( v > v_mean +  2*v_dev )


        index2 = np.where( v < v_mean -  2*v_dev )

        while len(index2[0]) != 0 :
            vect[0].pop(index2[0][0])
            vect[1].pop(index2[0][0])
            index2 = np.where( v > v_mean +  2*v_dev )

Magnitude = np.mean(Magnitudes)
Phase = np.mean(Phases)
devstd_new = [np.std(Magnitudes),np.std(Phases)]
```

Figura 17 - Elaborazione dei dati

Il ciclo for determina quindi due iterazioni, in cui il vettore v rappresenta prima l'array Magnitudes e poi l'array Phases. La media e la deviazione standard dell'array corrente, dopo essere state calcolate sfruttando le funzioni apposite fornite dalla libreria numpy, importata in questo codice con il nome np, vengono aggiunte alle rispettive strutture dati.

A questo punto comincia l'elaborazione vera e propria, che fa anch'essa uso del modulo numpy, in particolare della funzione where, la quale, data una condizione booleana relativa agli elementi di un array, restituisce le posizioni degli elementi per cui tale condizione risulta verificata; numpy semplifica notevolemente l'utilizzo di questa funzione permettendo il confronto fra un elemento di tipo array e un valore numerico, un'operazione normalmente non ammessa e che causerebbe perciò un'eccezione.

La variabile index1, che memorizza il risultato della funzione where, è una tupla, il cui primo elemento è costituito dall'array contenente le posizioni degli elementi dell'array v, quindi Magnitudes o Phases, che costituiscono gli outlier dell'insieme dei valori presenti nell'array, in quanto rispettano la condizione, passata come parametro alla funzione where, di essere maggiori del numero dato dalla somma della media e il doppio della deviazione standard.

Attraverso il ciclo while, poi, avviene l'eliminazione di tali outlier, sfruttando il metodo pop, il quale, richiamato su un array, ne elimina l'elemento che sta alla posizione passata come parametro. Nel codice, in particolare, la posizione è determinata dal primo elemento dell'array

contenente le posizioni degli outlier; dato che, ad ogni eliminazione, le posizioni degli outlier rimasti negli array Magnitude e Phases possono variare, è necessario aggiornare il valore della variabile index1 richiamando di nuovo la funzione where, in modo da ottenere le posizioni corrette. Inoltre, dato che l'array con le posizioni, ad ogni iterazione del ciclo while, ha una lunghezza inferiore di uno rispetto alla precedente, il ciclo viene fatto arrestare quando la dimensione dell'array diventa pari a 0, ovvero quando non vi sono più valori maggiori della media sommata al doppio della deviazione standard.

L'eliminazione di un outlier da un insieme di valori, sia esso quello delle fasi o dei moduli, comporta anche l'eliminazione dall'altro array del valore corrispondente, ovvero quello caratterizzato dalla stessa posizione, anche se quest'ultimo non costituisce un outlier del suo insieme; il valore dell'impedenza, infatti, è determinato sia dal modulo che della fase ed è necessario scartarli entrambi nel caso in cui anche solo uno dei due valori sia impreciso e debba essere eliminato.

La stessa procedura viene ripetuta una seconda volta con una condizione diversa, la quale determina l'eliminazione dei valori inferiori al numero dato dalla sottrazione fra la media e il doppio della deviazione standard.

Al termine del secondo ciclo while, dunque, i vettori memorizzati nella struttura dati vect rappresentano i valori di modulo e fase ripuliti dagli outlier. Ne viene poi calcolata la media e la deviazione standard, che vengono memorizzati in variabili apposite.

L'ultimo passo è costituito dall'invio dei dati ai due database, che avviene mediante gli oggetti connessione definiti nel primo blocco funzionale del software. I dati da salvare vengono immagazzinati all'interno di una tupla, la quale sarà unita alla query di tipo insert che provvederà all'inserimento effettivo dei dati nelle tabelle. Per quanto riguarda gli attributi, tutti quelli necessari sono stati ricavati nel corso dell'esecuzione del software; in particolare, le medie ponderate di modulo e fase sono contenute nelle variabili Magnitude e Phase, mentre i vari parametri statistici sono stati salvati nelle strutture dati presentate precedentemente. La temperatura è memorizzata nella variabile temperature, ricavata nella sezione relativa alla connessione con l'Arduino Nano, mentre la scheda di misura dell'impedenza e la corrispondente porta COM sono date dalle variabili panels_TU[i] e COM_ports_imp[i]. Il numero di valori rimasti dopo l'eliminazione degli outlier, infine, è ricavato semplicemente dalla dimensione degli array contenuti nella struttura dati vect dopo l'elaborazione ed è quindi ottenibile come len(vect[0]), dove len è una funzione predefinita di Python che restituisce la lunghezza dell'iterabile passato come parametro, mentre la data e l'ora della misurazione sono definite tramite la funzione now

della libreria datetime, che restituisce il timestamp, composto di data e ora, del momento in cui viene chiamata.
Tutti i valori numerici di tipo reale, prima di essere inseriti nel database, vengono inoltre arrotondati a due cifre decimali tramite la funzione round, standardizzando in questo modo i numeri che si trovano all'interno delle tabelle.

5.4 Server

Il server del sistema di monitoraggio gestisce l'interazione dell'applicazione mobile con il database, fornendole, in base alla richiesta effettuata, i dati necessari alla creazione dei grafici, predisposti secondo la struttura dati JSON riportata in figura 14. Implementa inoltre la possibilità di inviare, all'indirizzo e-mail dell'utente, un file CSV contenente i dati estratti dal database, nel caso in cui si abbia necessità di analizzarli e studiarli a parte, senza quindi bisogno di visualizzarli graficamente.
Le sue funzionalità sono codificate in un file Python denominato server.py, costantemente in esecuzione sulla macchina virtuale fornita del servizio EC2 di Amazon Web Services; la comunicazione con il client, come descritto precedentemente, avviene tramite delle Rest API di tipo GET. Il codice che ne permette l'implementazione sfrutta la libreria Flask e, in particolare, la sua estensione flask_restful; le funzionalità offerte da Flask, codificate attraverso i metodi di una classe omonima, consentono la comunicazione con l'esterno, gestendo, ad esempio, il reindirizzamento degli URL e la configurazione delle risposte. La libreria flask_restful, invece, mette a disposizione degli sviluppatori una classe, denominata Api, che permette l'esposizione di un servizio allineato ai principi Rest.
Il primo passo, dunque, consiste proprio nell'istanziare due oggetti, il primo di tipo Flask e il secondo di tipo Api; al costruttore del primo viene passato il nome del modulo corrente, recuperato attraverso la variabile __name__, predefinita in ogni script Python, mentre il secondo accetta come parametro proprio l'oggetto di tipo Flask.
Un'ulteriore configurazione necessaria è costituita dalla creazione della connessione al database remoto, che verrà utilizzata per recuperare i dati richiesti dall'applicazione. A tal proposito viene nuovamente impiegata la libreria psycopg2, in particolare la sua funzione connect, che restituisce un oggetto di tipo connessione, i cui metodi verranno utilizzati per eseguire le query al database. I parametri necessari a stabilire la connessione sono i medesimi rispetto a quelli descritti precedentemente nella sezione relativa al software che effettua le misure.
Il servizio viene attivato richiamando il metodo run dell'oggetto di tipo Flask, al quale vengono passati due parametri, l'host e la porta; il primo viene inizializzato a "0.0.0.0", in quanto si

vuole rendere il server accessibile a tutti i client, mentre al secondo, che rappresenta la porta su cui il servizio resta in ascolto, è stato assegnato un valore intero, superiore a 1024 per non occupare le porte conosciute. Una volta definito tale valore è stato necessario abilitare, tramite la console di gestione AWS, il security group dell'istanza EC2 ad accettare richieste dirette a quella porta, le quali sarebbero state altrimenti bloccate dal firewall presente di default.
La libreria flask_restful adotta un approccio ad oggetti per definire gli endpoint delle API che si stanno configurando. Introduce, in particolare, la classe Resource, che rappresenta una risorsa REST in senso astratto. Le risorse concrete, ovvero quelle interrogabili attraverso una richiesta HTTP, caratterizzata da un endpoint e da un metodo associato, sono dunque implementate come classi che ereditano dalla classe Resource. I verbi HTTP supportati da ognuna sono codificati attraverso i metodi pubblici della classe corrispondente; nel caso del sistema di monitoraggio, dunque, avendo la necessità di configurare tre endpoint cui fare richieste di tipo GET, sono state definite tre classi, ognuna caratterizzata da un unico metodo get.
Il servizio Flask viene notificato della presenza di risorse interrogabili attraverso il metodo predefinito dalla classe Api chiamato add-resource; i parametri che accetta sono due: la classe che rappresenta la risorsa e l'URL associato.

CLASSE	URL
Attribute	"/<panel_set>/<panel_id>/<start_date>/<end_date>/<start_time>/<end_time>/<attribute>/<frequency>"
AttributeFiltered	"filter/<panel_set>/<panel_id>/<filter_time>/<start_date>/<end_date>/<start_time>/<end_time>/<attribute>/<frequency>"
SendCsvFile	"raw/<panel_set>/<email>/<panel_id>/<start_date>/<end_date>/<start_time>/<end_time>/<frequency>"

Tabella 2 - Risorse ed Endpoint dell'applicazione

La tabella 2 contiene il nome delle classi che identificano le risorse esposte dal server del sistema di monitoraggio e l'URL corrispondente; questi ultimi sono composti da parti fisse e da valori variabili, riportati tra parentesi angolari. Rappresentano gli attributi che possono essere personalizzati contestualmente alla richiesta e costituiscono i parametri formali definiti nei vari metodi get delle classi risorse.

5.4.1 Attribute

Il metodo get della classe Attribute implementa le funzionalità necessarie alla creazione di un JSON contenente le informazioni sulle misurazioni di temperatura o di impedenza. Si occupa

quindi di effettuare una query al database per recuperare i dati richiesti dal client, ovvero dall'applicazione mobile, e, dopo averli elaborati opportunamente, di inviarli indietro predisposti in modo tale da essere pronti per essere visualizzati sottoforma di grafico.
I parametri personalizzabili associati all'URL della richiesta definiscono ciò che l'applicazione mobile, successivamente all'interazione dell'utente, vuole visualizzare e, di conseguenza, specificano la query che va effettuata:

- panel_set rappresenta l'installazione cui appartiene la scheda di acquisizione di cui si vogliono visualizzare i dati. Il suo valore può essere "tu" o "ma", per identificare, rispettivamente, il sito di Leon e quello di Gijon.
- panel_id caratterizza il nome della scheda di acquisizione richiesta.
- start_date, end_date, start_time e end_time identificano il periodo temporale in cui devono essere state effettuate le misure. I primi due codificano i giorni, gli ultimi due si riferiscono invece all'intervallo espresso in ore.
- attribute definisce l'attributo che si vuole visualizzare; è possibile ottenere il valore della temperatura e quelli di modulo, fase, parte reale e parte immaginaria per quanto riguarda l'impedenza elettrica.
- frequency, infine, rappresenta la frequenza di misurazione. Può assumere 21 valori, di cui 20 sono quelli che effettivamente costituiscono le frequenze a cui avvengono le misure di impedenza, mentre il restante è dato dal valore "all", che codifica la richiesta di ottenere le misurazioni effettuate a tutte le frequenze possibili; quest'ultimo attributo, sebbene non sia strettamente necessario in fase di visualizzazione, risulta utile in caso di problematiche riscontrate su una scheda di acquisizione per visionare tutte le misure effettuate in modo consecutivo. La frequenza viene ignorata nel caso in cui l'attributo richiesto sia la temperatura.

Una volta ricevuta la richiesta e i parametri che la caratterizzano, dunque, il primo passo consiste nel comporre la query, di tipo SELECT, che permette di estrarre i dati dal database. La variabile panel_set, innanzitutto, identifica la tabella da cui richiedere i dati; poi, viene effettuato un controllo sull'attributo che è stato richiesto: se corrisponde alla parte immaginaria o alla parte reale dell'impedenza, infatti, è necessario estrarre sia il valore del modulo che della fase, in quanto entrambi sono necessari per calcolarne i valori, mentre, se l'attributo desiderato è uno fra modulo, fase e temperatura, la query estrae direttamente i valori di quel campo. Gli altri attributi che devono essere recuperati dal database sono date e time, per identificare l'istante temporale in cui sono avvenute le misurazioni.

I vincoli che definiscono le righe che devono essere estratte sono costituiti dal valore della variabile panel id, che contiene il nome della scheda di acquisizione che si vuole visualizzare, e dall'intervallo temporale, sia in termini di giorni che di ore, in cui devono essere avvenute le misurazioni.
Dato che, come descritto nella sezione relativa al database, le informazioni temporali sono memorizzate come stringhe, è necessario effettuare una conversione di tali valori in degli oggetti Date e Timestamp, che rappresentano, nel database, date e marche temporali e supportano quindi operazioni di confronto. A tal proposito, vengono utilizzate le funzioni definite nel linguaggio SQL to_date e to_timestamp, le quali, data una stringa e il formato in
cui la data o la marca temporale sono codificate, la convertono in un valore Date e Timestamp.
La porzione della query che definisce i vincoli è riportata in figura 18.

```
where id = %s and to_date(date,'DD-MM-YYYY') between to_date(%s,'DDMMYYYY')
and to_date(%s,'DDMMYYYY') and to_timestamp(time,'HH24:MI:SS')
between to_timestamp(%s,'HH24:MI:SS') and to_timestamp(%s,'HH24:MI:SS')
```

Figura 18 - Vincoli della query

I valori %s sono dei placeholder e rappresentano ognuno un parametro passato alla query; vengono utilizzati per evitare di comporla direttamente con un input esterno. Nell'ordine in cui si trovano, le variabili che si sostituiscono ai placeholder sono: panel_id, start_date, end_date, start_time e end_time.
Inoltre, dato che la direttiva SELECT restituisce le righe richieste in ordine casuale, è necessario specificare la necessità di ottenerle in ordine cronologico, in modo da poter essere poi configurate correttamente senza bisogno di un ulteriore ordinamento. La query effettuata è composta quindi anche dalla clausola ORDER BY, che permette appunto di ordinare il risultato secondo dei parametri che devono essere specificati di seguito. Nel caso del sistema di monitoraggio, essendo l'ordine desiderato quello cronologico, le righe devono essere ordinate rispetto all'anno, al mese, al giorno e infine all'orario. Viene dunque utilizzata la funzione PostgreSQL date_trunc, che permette di considerare solamente una specifica parte della data passata come parametro. L'ordinamento effettuato è mostrato nel codice riportato in figura 19.

```
order by DATE_TRUNC('year', to_date(date,'DD-MM-YYYY')),
DATE_TRUNC('month', to_date(date,'DD-MM-YYYY')),
DATE_TRUNC('day', to_date(date,'DD-MM-YYYY')),time
```

Figura 19 - Ordinamento delle righe estratte

La query viene poi effettuata sfruttando il metodo execute fornito dall'oggetto di tipo connessione instanziato all'avvio del servizio. Le righe estratte vengono recuperate attraverso il metodo fetchall, definito nella proprietà cursor dell'oggetto connessione, che restituisce un array i cui elementi, a loro volta degli array, rappresentano le righe stesse.
Prima di procedere alla creazione del JSON che verrà inviato al client, viene effettuata una prima elaborazione sull'array ottenuto. I dati richiesti devono essere visualizzati graficamente e non è necessario, nel caso in cui i parametri specificati determinino un grande numero di righe estratte, inviare tutti i dati all'applicazione, in quanto conferirebbero al grafico un livello di dettaglio troppo elevato e ne appesantirebbero solamente la renderizzazione. In figura 20 è presente l'algoritmo applicato, che riduce, in modo proporzionale, il numero delle misurazioni a un valore vicino a 500.

```
if len(rows_all) > 500:
    temp = np.arange(0,len(rows_all),int(round(len(rows_all)/500)))
    rows = [e for i, e in enumerate(rows_all) if i in temp]
else:
    rows = rows_all
```

Figura 20 - Algoritmo per ridurre le misurazioni

La variabile rows_all memorizza il valore restituito dal metodo fetchall. Viene effettuato un controllo sulla sua lunghezza; se la sua dimensione risulta minore di 500, infatti, non vi è bisogno di ridurne gli elementi e si procede con la configurazione del JSON. Se la condizione è verificata, però, si effettua l'elaborazione descritta. Attraverso la funzione arange, fornita dalla libreria Numpy, viene generato un array, memorizzato nella variabile temp, i cui elementi costituiscono le posizioni degli elementi dell'array rows_all che non devono essere eliminati, quindi le misure i cui dati verranno inviati al client. I parametri che la funzione accetta sono tre: i primi due definiscono l'intervallo cui devono appartenere gli elementi dell'array generato, di cui il secondo, ovvero il limite destro, non incluso, mentre il terzo costituisce il passo che deve separare gli elementi dell'array. Nel caso del codice presentato, i limiti dell'intervallo, dovendo rappresentare le posizioni dell'array rows_all, sono dati dal valore 0 e dalla sua

lunghezza, mentre il passo è dato dall'arrotondamento ad intero del quoziente fra la dimensione dell'array e il valore 500. In questo modo i valori generati risultano distribuiti in modo proporzionale all'interno dell'intervallo specificato. Ad esempio, se l'array rows_all fosse composto di 2000 elementi, l'array temp sarebbe costituito di 500 elementi spaziati di 4, ovvero il risultato della divisione fra 2000 e 500.
L'array generato viene poi utilizzato per la creazione del vettore rows, quello a partire dal quale verrà configurato il JSON finale; rows conterrà quindi gli elementi di rows_all che si trovano alle posizioni contenute all'interno di temp. Per ottenere questo risultato si fa ricorso al costrutto Python chiamato list comprehension, che consente la creazione di array con una sintassi ridotta e semplificata. Si sfrutta inoltre la funzione enumerate, anch'essa predefinita in Python, la quale consente di effettuare un ciclo su una struttura dati restituendo, ad ogni iterazione, i valori presenti nell'array e la posizione corrispondente. La terza istruzione contenuta nella figura 20, quindi, implementa questo meccanismo, inserendo nell'array rows gli elementi di rows_all caratterizzati da una posizione contenuta nell'array temp.

Il passo successivo consiste nel creare la struttura dati JSON che verrà inviata al client. Come mostrato nella figura 14, tale JSON è formato da un array di oggetti che rappresentano le misure, ognuno caratterizzato dai valori relativi al momento in cui è avvenuta la misurazione e all'attributo richiesto. Il corrispettivo del tipo di dato oggetto JSON, in ambiente Python, è costituito dal dizionario, una struttura dati anch'essa valorizzata attraverso delle coppie chiave valore. La configurazione del JSON consiste dunque nella creazione di un array contenente dei dizionari, ognuno dei quali rappresenta una misura e deve essere riempito con le informazioni relative a partire dagli elementi dell'array rows.
In figura 21 è riportato il codice che implementa la creazione dell'array. Viene effettuato, innanzitutto, un controllo sull'attributo richiesto, in quanto la parte reale e la parte immaginaria, a differenza degli altri attributi che possono essere inseriti direttamente nel dizionario, richiedono di essere calcolate. Sono qui omesse le porzioni di codice relative alla parte immaginaria e agli altri attributi, ma, al di là del calcolo, diverso o assente, sono funzionalmente identiche a quella presentata.
Una volta delineato l'attributo, viene eseguito un ciclo for per un numero di volte pari alla lunghezza dell'array rows; ogni iterazione comporta l'aggiunta di un elemento all'array result, quello che sarà poi inserito nel JSON e inviato al client. Le informazioni sulla misura corrente vengono strutturate nel dizionario measure, utilizzato come variabile di appoggio in ogni iterazione.

```
if attribute == "real" :
        for i in range(len(rows)):
            measure = {}

            date = rows[i][3]
            date = date[6:10]  + date[3:5] + date[0:2]
            complete_date = date + " " + rows[i][2]

            real = rows[i][0] * math.cos(rows[i][1])
            measure = { "real" : real , "datetime" : complete_date}

            result.append(measure)

elif attribute == "imaginary" :

else:
```

Figura 21 - Creazione del JSON

Viene, in prima istanza, creata la stringa complete_date, contenente la data e l'istante temporale separate da uno spazio, che rappresenta il momento in cui è stata effettuata la misura; le operazioni svolte sulla stringa date sono necessarie alla comunicazione con l'applicazione mobile, la quale richiede che la data venga codificata nel formato "yyyyMMdd". Successivamente, viene calcolato il valore della parte reale dell'impedenza, data dal prodotto fra il modulo e il coseno della fase, ottenuto grazie alla funzione cos del modulo Math. Le informazioni, strutturate come coppie chiave valore, vengono poi inserite all'interno della variabile measure, la quale è poi aggiunta all'array result. Questo, al termine del ciclo for, sarà quindi composto da un insieme di dizionari, ognuno contenente i dati delle singole misurazioni.

Prima di inviare la risposta al client, tuttavia, è necessario sottoporre l'array result ad un'ulteriore fase di processing. Come descritto nella sezione relativa all'inserimento dei dati nel database, nel caso in cui la misura effettuata non sia andata a buon fine, viene comunque salvato un record nelle tabelle caratterizzato dal valore 0 in tutti campi relativi all'impedenza, in particolare in quelli del modulo e della fase. Se la richiesta effettuata dall'applicazione mobile comprende una di queste misurazioni, il grafico che si vuole visualizzare risulterebbe compromesso e, di conseguenza, privo di significato per chi usufruisce dell'applicazione.

Prima di comporre il JSON finale, dunque, viene richiamata una funzione denominata evaluate_measure, con l'obiettivo di eliminare le irregolarità presenti nelle misure. I parametri formali della funzione sono due: il primo è un array di elementi iterabili, contenente i dati su cui deve essere effettuata l'elaborazione, il secondo è invece l'attributo specifico i cui valori devono essere, eventualmente, corretti. I parametri attuali con cui la funzione viene attivata sono l'array result e l'attributo richiesto dal client.

Gli zeri trovati nell'array vengono eliminati secondo i seguenti criteri, in base alla posizione in cui vengono rilevati:

- Se il valore della misura iniziale è uno zero, gli viene sostituito il valore dell'array diverso da zero più vicino a livello di posizione. La stessa procedura si applica anche agli elementi immediatamente successivi se anche questi risultano pari a zero.
- Se l'ultima misura dell'array, o quelle immediatamente precedenti, sono uguali a zero, si applica il medesimo meccanismo e viene loro assegnato il valore diverso da zero nella posizione più vicina.
- Se lo zero è rilevato in una posizione centrale dell'array, gli viene assegnato il valore dato dalla media ponderata degli elementi adiacenti, purché questi siano diversi da zero. Nel caso in cui vi siano più zeri consecutivi, invece, viene assegnato a ciascun elemento un valore tale per cui vi sia una progressione lineare tra gli elementi diversi da zero adiacenti al sottoinsieme dei valori nulli.

La tabella 3 riporta degli esempi di funzionamento della funzione evaluate_measure, uno per ogni caso presentato. Per ragioni di semplificazione, gli esempi sono riportati utilizzando come input un array di array e l'indice su cui vuole essere effettuata l'elaborazione, ma nel caso del sistema di monitoraggio, come puntualizzato precedentemente, l'input è dato dall'array result, quindi un array di dizionari, e dall'attributo richiesto dal client, ad esempio "magnitude". Il funzionamento resta, in ogni caso, il medesimo.

Input	Output
([[1,0,4],[3,1,4],[5,4,4]],1)	[[1,1,4],[3,1,4],[5,4,4]]
([[1,0,4],[1,0,4],[1,3,4],[3,2,4],[5,4,4]],1)	[[1,3,4],[1,3,4],[1,3,4],[3,2,4],[5,4,4]]
([[1,3,4],[3,2,4],[5,0,4]],1)	[[1,3,4],[3,2,4],[5,2,4]]
([[1,2,4],[1,2,4],[1,3,4],[3,0,4],[5,0,4]],1)	[[1,2,4],[1,2,4],[1,3,4],[3,3,4],[5,3,4]]
([[1,2,4],[1,2,4],[1,0,4],[3,4,4],[5,0,4]],1)	[[1,2,4],[1,2,4],[1,3,4],[3,4,4],[5,0,4]]
([[1,3,4],[1,0,4],[1,0,4],[3,0,4],[5,4,4]],1)	[[1,3,4],[1,3.25,4],[1,3.5,4],[3,3.75,4],[5,4,4]]

Tabella 3 - Funzione evalutate_measure

L'implementazione della funzione è riportata in figura 22. Il codice si struttura su quattro iterazioni; le prime due, effettuate tramite cicli while, hanno come obiettivo quello di correggere eventuali zeri presenti all'inizio e alla fine dell'array, assegnando a tali valori quelli adiacenti finché non ne viene trovato uno diverso da zero. Queste iterazioni vengono interrotte se il contatore utilizzato per incrementare la posizione di controllo diventa pari alla lunghezza

dell'array, ovvero nel caso in cui tutti i valori presenti siano pari a zero. In tal caso, non essendo possibile ricavare alcun tipo di valore, la funzione viene interrotta restituendo la struttura dati così come era stata passata.

Il terzo ciclo itera di nuovo sull'array a, inserendo all'interno della struttura dati denominata pos le posizioni degli elementi di valore pari a 0, quindi quelli su cui è necessario intervenire; l'elaborazione viene poi svolta all'interno dell'ultimo ciclo for, il quale itera appunto sugli elementi dell'array pos.

```
def evaluate_measure(a,index):
        k=0
        while a[len(a)-1][index] == 0:
                a[len(a) -1][index] = a[len(a)-1-k-1][index]
                k += 1
                if k == len(a)-1:
                        return a
        k=0
        while a[0][index] == 0:
                a[0][index] = a[k+1][index]
                k += 1
             if k == len(a)-1:
                    return a
        pos = []

        for i in range(0,len(a)):
                if a[i][index] == 0:
                        pos.append(i)

        for j in pos:
        if a[j] != 0 :
                        break
                cont = 1
                for i in range(j+1,len(a)):

                        if a[i][index] != 0 :
                            break
                        else :
                            cont +=1

                if cont == 1:
                        a [j][index] = ( a[j-1][index] + a[j + 1][index] )  / 2
                else:
                    a1 = a[j+cont][index]
                        a2 = a[j-1][index]
                        val = (a1-a2) / (cont+1)
                        for z in range(j,len(a)):
                            if a[z][index] == 0:
                                    a[z][index] = a[z-1][index] + val
                            else:
                                    break
        return a
```

Figura 22 - Funzione evaluate_measure

Per ogni posizione, rappresentata dalla variabile j, viene inizializzato un contatore, valorizzato poi attraverso un ulteriore ciclo for innestato, per memorizzare il numero di elementi pari a 0 riscontrati nelle posizioni successive; il valore iniziale del contatore è 1, poiché la posizione j è appunto quella di un elemento pari a 0.
Se il valore del contatore resta 1, ovvero non vi sono altri elementi adiacenti valorizzati a 0, all'elemento di posizione j viene assegnato il valore dato dalla media ponderata degli elementi contigui, i cui valori sono sicuramente diversi da 0: quello successivo perché altrimenti il contatore non sarebbe stato pari a 1, quello precedente poiché il ciclo che individua le posizioni le aggiunge all'array pos in ordine crescente e il primo elemento dell'array è sicuramente diverso da 0 grazie al ciclo while svolto all'inizio della funzione.
Se, invece, il valore del contatore è maggiore di 1, si entra nel ramo else della struttura condizionale; vengono inizializzate due variabili, a1 e a2, contenenti i valori degli elementi adiacenti al sottoinsieme di valori pari a 0; le posizioni che li individuano sono date da j+cont e da j-1 e, per le stesse ragioni indicate nel caso precedente, sono sicuramente di valore diverso da 0. Viene poi calcolata la differenza fra a1 e a2 normalizzata rispetto al numero di elementi che li separano, memorizzata nella variabile val, e, attraverso un ulteriore ciclo for, vengono valorizzati tutti gli elementi pari a zero, assegnando, ad ognuno, il valore del precedente, che nel caso del primo elemento coincide con a1, sommato a val; il ciclo for si interrompe quando non vi sono più elementi pari a 0.
Tramite questo meccanismo si ottiene una variazione proporzionale e, a livello di misurazioni, plausibile, fra gli elementi, che consente di mantenere nel database quanti più dati possibili, anche tenendo conto di misure non andate a buon fine, ma permettendo comunque una visualizzazione dei grafici corretta e ragionevole.
Il metodo get della classe Attribute, a questo punto, ha terminato le elaborazioni necessarie e si procede dunque alla creazione del JSON da inviare al client, all'interno del quale viene inserito l'array restituito dalla funzione evaluate_measure.
L'ultima istruzione è costituita dalla direttiva return, seguita dal risultato della funzione definita nel modulo Flask jsonify, la quale trasforma in una struttura JSON valida l'array di dizionari contenenti le informazioni sulle misure passato come parametro. L'invio vero e proprio viene poi gestito internamente all'oggetto Flask istanziato all'avvio del servizio.

5.4.2 AttributeFiltered

La classe AttributeFiltered viene attivata nel momento in cui l'utente dell'applicazione mobile richiede la visualizzazione di un grafico filtrato attraverso una media mobile di una finestra

temporale selezionabile. I parametri formali del suo metodo get, riportati nella seconda riga della Tabella 2, sono gli stessi di quelli accettati dall'omonimo metodo definito per la classe Attribute e presentati nella sezione precedente; rispetto a quelli, tuttavia, vi è un parametro aggiuntivo, rappresentato dalla variabile filter_time, la quale identifica il valore della finestra temporale scelta dall'utente e che deve essere incluso nella richiesta inviata al server per poter effettuare il calcolo opportuno.

La modalità con cui i dati vengono recuperati dal database è funzionalmente identica a quella implementata nel metodo get della classe Attribute; i parametri formali, che identificano i valori che vengono inviati dal client, definiscono la query di tipo SELECT che permette l'estrazione dalle tabelle delle righe richieste. La dimensione della struttura dati così ottenuta viene poi ridotta, con una procedura di elaborazione analoga a quella presentata precedentemente.

Si procede quindi alla creazione delle strutture dati su cui avverrà il calcolo della media mobile. A differenza di ciò che accadeva nella classe Attribute, i valori che devono essere inviati al client, ovvero l'attributo richiesto e l'identificatore del momento in cui è avvenuta la misura, vengono inizialmente inseriti in due array separati, chiamati, rispettivamente, values e dates. Sul primo, poi, viene richiamato il metodo evaluate_measure_vect, che implementa le funzionalità del metodo evaluate_measure estendendone il supporto anche ad array di valori numerici, come appunto values. In questo modo vengono corretti i valori pari a zero che identificano le misure non andate a buon fine.

Il calcolo della media mobile avviene sfruttando le funzioni messe a disposizione dalla libreria Pandas. Il codice che lo implementa è riportato nella figura 23.

```
df_prepare = {"values": values, "dates" : dates}
df = pandas.DataFrame(df_prepare)
filter_time = int(filter_time)
df[attribute] = df["values"].astype(float).rolling(window=filter_time).mean()

for j in range(0,len(values)):
        measure = {}

        measure = { attribute : df.iloc[j][attribute], "datetime" : df.iloc[j]["dates"]}
        result.append(measure)
```

Figura 23 - Calcolo della media mobile

È necessario, innanzitutto, configurare un oggetto di tipo DataFrame, la struttura dati predefinita in Pandas che è possibile manipolare con le funzionalità offerte dalla libreria stessa. Un DataFrame può essere inizializzato in più modalità. Nel caso del codice riportato, viene

popolato attraverso un dizionario, identificato dalla variabile df_prepare, al cui interno sono stati inseriti gli array values e dates contenenti le informazioni sulle misure.
Dopo aver effettuato la conversione ad intero del valore della finestra temporale, necessario poiché tutti i parametri vengono ricevuti come stringhe, viene effettuato il calcolo della media mobile; in particolare, vengono attivati i metodi rolling e mean, i quali, richiamati su una colonna di un DataFrame, ne restituiscono la media mobile caratterizzata da una finestra temporale passata come parametro al metodo rolling. Viene quindi aggiunta una colonna al DataFrame, il cui nome è dato dall'attributo richiesto dal client, che viene valorizzata dal calcolo della media mobile sulla colonna values, ovvero quella contenente i valori delle misure così come sono stati estratti dal database.
Si procede poi alla creazione del JSON da inviare al client, che è strutturato come quello presentato in figura 14. La procedura è analoga a quella che avviene per la classe Attribute: viene predisposto un array di dizionari, anche questo denominato result, dove ogni elemento rappresenta una misura avvenuta e viene poi eseguito un ciclo for per un numero di volte pari alla lunghezza dell'array values, quindi al numero delle misure che stanno per essere inviate al client. Ad ogni iterazione, il dizionario measure, utilizzato come variabile di appoggio, viene popolato e inserito nell'array result.
I valori che identificano la misura devono essere estratti dal DataFrame, in particolare dalle colonne attribute, ovvero quella contenente i valori filtrati, e dates, contenente invece le stringhe che identificano l'istante temporale; viene dunque utilizzata la funzione iloc, che consente di recuperare gli elementi contenuti in un DataFrame tramite indicizzazione.
Al temine del ciclo for, dunque, l'array result risulta correttamente popolato e il JSON, configurato tramite la funzione jsonify del modulo Flask, viene inviato al client.

5.4.2 SendCsvFile

Il metodo get della classe SendCsvFile implementa le funzionalità che consentono l'invio di un file CSV, contenente i dati sulle misure estratti dal database, all'email dell'utente dell'applicazione; questa funzione è stata introdotta, come specificato nei requisiti funzionali descritti nel capitolo 3, per permettere agli utenti di analizzare e studiare i risultati del monitoraggio a parte, quindi senza visualizzarli in un grafico all'interno dell'app stessa. I dati vengono infatti forniti in un formato facilmente integrabile con altri strumenti software, come, ad esempio, Matlab, e già strutturati in modo da essere pronti per l'analisi senza necessitare ulteriori configurazioni.

I parametri formali che caratterizzano il metodo get sono analoghi a quelli delle altre classi, ma presentano due differenze; la prima risiede nel parametro aggiuntivo identificato dalla variabile e-mail, che rappresenta l'indirizzo dell'utente al quale dovrà essere inviato il file, la seconda consiste invece nell'assenza del parametro attribute, in quanto, ad ogni richiesta, vengono sempre estratti ed inviati i valori di modulo, fase e temperatura, senza possibilità di selezionarne solo uno. Inoltre, a differenza di ciò che accade nei metodi get delle altre classi, il parametro panel_id, che contiene il nome della scheda di acquisizione, può assumere anche il valore "all", che codifica la richiesta dell'utente di ottenere i dati relativi a tutte le schede; questa possibilità non era implementabile per le altre due richieste, in quanto il grafico dell'impedenza misurata su campioni di calcestruzzo diversi non avrebbe alcun significato.
I dati vengono estratti dal database mediante una query di tipo SELECT, composta in base ai parametri inviati dal client, e vengono poi inseriti all'interno di un DataFrame Pandas. Tramite il suo metodo to_csv, viene creato e successivamente riempito con i valori relativi alle misure il file CSV, chiamato "measures.csv", che viene salvato in locale, quindi sulla macchina presso cui risiede il servizio, e poi inviato al client tramite la funzione send_mail, riportata in figura 24. La funzione viene richiamata passando come parametro la variabile email, contenente l'indirizzo del client cui inviare il file.

```
def send_mail(receiver):

    msg = MIMEMultipart()
    msg['From'] = sender
    msg['Date'] = formatdate(localtime = True)
    msg['Subject'] = 'Measures'
    msg.attach(MIMEText("Measures"))

    part = MIMEBase('application', "octet-stream")
    part.set_payload(open("measures.csv", "rb").read())
    part.add_header('Content-Disposition', 'attachment; filename="measures.csv"')
    msg.attach(part)

    try:
        server = smtplib.SMTP()
        server.connect('smtp.gmail.com', '587')
        server.starttls()
        server.login(sender, password)
        server.sendmail(sender, receiver, msg.as_string())
        message = "Email sent correctly to "+ email +"!"

    except Exception as e:
        message = str(e)

    return message
```

Figura 24 - Funzione send_mail

L'invio dell'e-mail avviene tramite il protocollo SMTP, acronimo di Simple Mail Transfer Protocol, il cui utilizzo è reso possibile dalle funzionalità messe a disposizione dalla libreria

Python smtplib. Prima di effettuare la connessione con il server, è necessario configurare un oggetto di tipo MIMEMultipart(), che consente l'estensione delle funzionalità base del protocollo SMTP, fornendo, ad esempio, il supporto all'invio di file allegati alla mail stessa. Vengono inizializzate le sue proprietà, specificando, in particolare, il mittente, codificato nella variabile sender, definita al di fuori del metodo, la data, ottenuta grazie alla funzione formatdate, e l'oggetto della mail, cui viene assegnata la stringa "Measures". Tramite il metodo attach, poi, vengono uniti all'oggetto msg due sotto parti: la prima, un oggetto di tipo MIMEText, rappresenta il corpo della mail, che, in questo caso, è costituito esclusivamente dalla stringa "Measures", mentre la seconda, un oggetto di tipo MIMEBase, identifica l'allegato della mail, ovvero il file "measures.csv". Il contenuto viene inizializzato mediante il metodo set_payload, al quale viene passato come parametro il risultato della lettura del file stesso, mentre le modalità di configurazione sono definite nell'header del messaggio tramite il metodo add_header.
All'interno di un costrutto try-catch viene poi stabilita la connessione con il server Gmail sulla porta 587. La comunicazione viene cifrata tramite il protocollo starttls, attivato mediante l'omonima funzione.
Dopo aver effettuato il login, attraverso le credenziali di un account creato appositamente, si procede all'invio della mail, richiamando la funzione sendmail, alla quale vengono passati i valori di sender, receiver e della variabile msg convertita in stringa. Il valore di ritorno della funzione send_mail è una stringa che notifica il successo o il fallimento dell'invio; in particolare, nel caso in cui si verifichi un'eccezione, viene restituito il messaggio incluso nell'eccezione stessa.
Il metodo get termina infine inviando al client un JSON, strutturato grazie alla funzione jsonify, contenente la stringa restituita dalla funzione send_mail, in modo che l'esito, positivo o negativo, dell'invio della mail venga comunicato anche all'utente.

5.5 Applicazione mobile

L'applicazione mobile costituisce l'interfaccia del sistema di monitoraggio, ovvero il software con cui l'utente finale deve interagire per visualizzare i risultati e l'andamento delle grandezze oggetto dell'osservazione. Come descritto nel capitolo relativo agli strumenti utilizzati, l'applicazione è stata sviluppata sfruttando il framework Flutter e il linguaggio di programmazione Dart, tecnologie innovative che consentono la produzione di un'unica code base in grado di compilare per tutti i sistemi operativi mobile principali, come Android e iOS.
La struttura dell'applicazione, semplice e lineare, ne rende la fruizione intuitiva ed immediata, oltre ad essere organizzata in modo tale da essere facilmente estendibile e modificabile. In

figura 25 è riportata la mappa dell'applicazione, contenente le schermate che la compongono e i collegamenti che le uniscono.

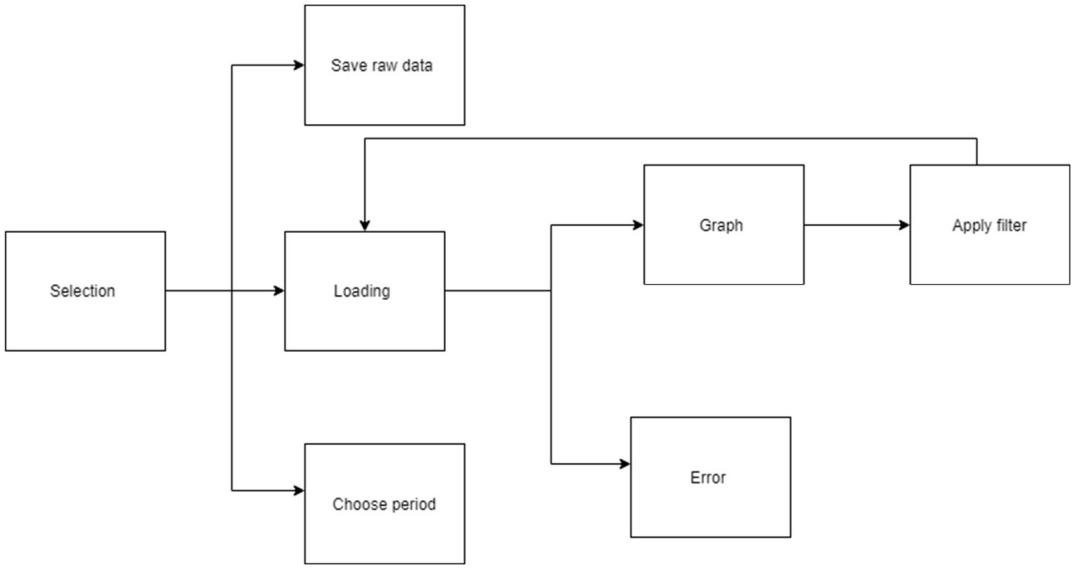

Figura 25 - Mappa dell'applicazione

La versione ufficiale consta in realtà di un'ulteriore schermata, quella iniziale, in cui è possibile effettuare la registrazione o, se si è già utenti, il login all'interno dell'applicazione; la sua realizzazione, tuttavia, non è stata effettuata nell'ambito di questo elaborato, in quanto rappresenta un pattern unico utilizzato anche per altre applicazioni del progetto EnDurCrete, e, per tale ragione, non è stata inclusa nella mappa presentata di seguito. Una sua porzione di codice verrà comunque analizzata, in quanto modificata appositamente per lo sviluppo di questa applicazione; una delle funzionalità implementate, ovvero quella di inviare un file contenente i dati relativi alle misure, necessita infatti dell'indirizzo e-mail dell'utente, che viene sempre richiesto in fase di login o di registrazione. A tal proposito, per evitare che l'utente debba reinserirlo una seconda volta, sono state aggiunte delle funzionalità alla logica che implementa l'autenticazione per rendere fruibile l'indirizzo e-mail anche ad altre parti dell'applicazione, in particolare alla classe che gestisce la comunicazione con il server.

Ogni rettangolo della figura 25 rappresenta una schermata dell'applicazione; la prima con cui l'utente interagisce, dopo essersi registrato o aver effettuato il login, è quella di selezione, che rappresenta il nodo centrale da cui hanno inizio le possibili azioni.

Al suo interno, infatti, è possibile scegliere i parametri che individuano il grafico che si vuole visualizzare, tra cui, ad esempio, la scheda di acquisizione, la frequenza di misurazione e l'attributo che si vuole visionare; vi è inoltre l'opzione per indicare il periodo di osservazione, che

può essere definito attraverso dei parametri statici, selezionabili nella schermata stessa, o può essere personalizzato attraverso dei widget interattivi implementati in un'altra schermata dell'applicazione, denominata appositamente "choose period".
Dopo aver configurato i parametri necessari, l'utente può scegliere se visualizzarne il grafico o se richiedere all'applicazione l'invio del file contenente i dati selezionati; a tal proposito, sono presenti due bottoni che rappresentano le due possibilità. Nel secondo caso, l'utente viene ridiretto alla schermata "save raw data", dalla quale è possibile effettuare la richiesta al server. Poi, tramite un Alert Dialog, viene mostrato un messaggio contenente l'esito, positivo o negativo, dell'invio dell'e-mail.
Se l'utente richiede invece la visualizzazione del grafico, viene attivata una schermata di caricamento, la quale, mentre all'utente viene mostrato un widget che rappresenta l'operazione di caricamento, effettua in background la richiesta al server e aziona le funzionalità necessarie alla predisposizione del grafico. Si occupa inoltre di gestire eventuali errori, mostrando in tal caso una schermata appositamente definita in cui viene notificata all'utente la tipologia di eccezione riscontrata. Se le operazioni avvengono con successo, l'utente viene ridiretto alla schermata contenente il grafico richiesto. Al suo interno vi è poi la possibilità di richiedere il filtraggio dei dati secondo una media mobile di una finestra temporale selezionabile. Ciò avviene mediante una finestra modale attivabile tramite un bottone. Nel momento in cui l'utente richiede il grafico filtrato, viene di nuovo mostrata la schermata di caricamento e si attiva una nuova richiesta diretta al server.

5.5.1 Servizi

Nell'ottica di separare, per quanto possibile, la gestione dell'interfaccia grafica e l'implementazione delle funzionalità offerte dall'applicazione, sono state definite due classi, inserite all'interno di un package denominato services, che contengono i metodi necessari a garantire il funzionamento dell'app ma che non sono coinvolte nella realizzazione grafica delle schermate.
La prima, denominata graphService, implementa i metodi che supportano la creazione del grafico, in particolare quelli legati alla configurazione dello scheletro del grafico stesso; quest'ultima è infatti determinata da una serie di parametri che devono essere ricavati in modo dinamico in base ai dati richiesti dall'utente, come il numero di spunte che devono essere mostrate nell'asse delle ascisse o il tipo di formattazione da applicare alle etichette ad esse associate. I metodi di questa classe verranno analizzati nella sezione relativa alla creazione del grafico, in modo da evidenziarne le caratteristiche in funzione delle ragioni per cui sono stati definiti.

DownloadService, invece, è la classe che gestisce l'interazione con il server e viene quindi attivata nel momento in cui l'utente richiede la visualizzazione di un grafico o l'invio del file contenente i dati relativi alle misure.

5.5.1.1 DownloadService

Un oggetto di tipo DownloadService deve essere istanziato nel momento in cui si ha la necessità di interagire con il server. Le sue proprietà, che devono essere inizializzate attraverso il costruttore all'atto della chiamata, rappresentano i parametri che l'utente ha selezionato e sono quindi utilizzate per comporre gli URL verso cui indirizzare la richiesta. Risultano, di fatto, analoghe ai parametri formali definiti per i metodi get delle classi che implementano il servizio REST descritto nella sezione precedente.

Vi è poi una proprietà aggiuntiva, che non deve essere inizializzata in quanto costituita da un valore statico e prefissato, che rappresenta, tramite una stringa, l'indirizzo IP pubblico della macchina EC2 su cui risiede il server, seguito, dopo i due punti necessari a comporre un URL corretto, dalla porta su cui il servizio si trova in ascolto.

La classe presenta quattro metodi, di cui tre, pubblici, sono quelli che mappano le API esposte dal server. Tuttavia, dato che le richieste per ottenere i dati su cui costruire il grafico, filtrato o meno, sono funzionalmente identiche e si differenziano unicamente per la composizione dell'URL, i metodi corrispondenti si limitano a costruire l'indirizzo corretto e a passarlo come parametro al quarto metodo della classe, denominato processData, che è quello che si occupa di effettuare la richiesta e di processarne la risposta.

L'implementazione del metodo processData è mostrata in figura 26; il suo nome inizia in realtà con un underscore, la modalità utilizzata da Dart per identificare metodi e proprietà private.

Utilizza le funzionalità offerte dalla libreria http, che consente la gestione della comunicazione sul protocollo omonimo, e dal modulo async, che fornisce il supporto all'esecuzione di operazioni asincrone, proprio come le chiamate ad API esterne effettuate all'interno dell'applicazione. Per questa ragione, nell'intestazione del metodo figura la parola chiave async e il tipo di ritorno è wrappato all'interno del tipo di dato Future, introdotto appositamente dalla libreria per individuare i risultati di metodi asincroni.

In particolare, la funzione restituisce una lista di oggetti Measure, la classe che viene utilizzata per mappare il concetto di misura all'interno dell'applicazione e che ne costituisce dunque il modello principale. La classe Measure contiene unicamente due proprietà, una di tipo double, che rappresenta il valore dell'attributo richiesto dall'utente, e l'altra di tipo DateTime, che identifica invece l'istante in cui la misura è avvenuta.

```
Future<List<Measure>> _processData(String url) async {
  List<Measure> measures = [];

  http.Response response = await http.get(url).timeout(
    Duration(seconds: 5),
    onTimeout: () {
      throw 'Can\'t reach the server';
    },
  );

  if (response.statusCode == 200) {
    var data = jsonDecode(response.body);
    var allMeasures = data["measures"];

    for (var measure in allMeasures) {
      Measure m = Measure(
          time: DateTime.parse(measure["datetime"]),
          value: measure[attribute]);
      measures.add(m);
    }
  } else {
    throw Exception('Your request was incorrect');
  }

  return measures;
}
```

Figura 26 - Funzione processData

Dopo aver inizializzato una lista di oggetti Measure, inizialmente vuota, viene richiamato il metodo get del modulo http, che effettua una richiesta di tipo GET all'url passato come parametro; nel caso del codice, il parametro è costituito dalla variabile url, a sua volta un parametro formale del metodo processData. Alla richiesta viene associato un timeout di cinque secondi, che definisce l'istante temporale massimo oltre il quale, se il server non fornisce una risposta, viene generata un'eccezione specifica che viene poi mostrata all'utente. Se la risposta è invece ricevuta correttamente e il codice di stato associato è pari a 200, si procede alla serializzazione del JSON ottenuto. Viene utilizzato il metodo jsonDecode, predefinito in Dart, che converte il contenuto del body in un oggetto JSON, manipolabile con le direttive del linguaggio stesso. Si estrae così il valore corrispondente alla chiave "measures", ovvero l'array contenente le informazioni sulle misure richieste e, tramite un costrutto for each, viene eseguita un'iterazione su tutti gli oggetti contenuti all'interno dell'array, identificati localmente dalla variabile measure. Per ognuno di essi viene istanziato e poi aggiunto alla lista definita inizialmente un oggetto di

tipo Measure, le cui proprietà vengono inizializzate con i valori contenuti all'interno della variabile locale measure.
Il valore che identifica l'istante temporale in cui la misura è avvenuta viene ricevuto come stringa, ma deve essere convertito in un oggetto DateTime per poter definire correttamente l'oggetto Measure. A tal proposito, si sfrutta il metodo parse, definito nella classe DateTime, che trasforma una stringa, codificata nello specifico formato "yyyyMMdd HH:mm:ss", nel corrispondente oggetto DateTime. Questo è la ragione per cui, in fase di creazione del JSON lato server, dopo aver estratto i dati dal database è necessario manipolare le stringhe contenenti la data e l'ora in cui è avvenuta la singola misura.
Al termine del ciclo, la lista risulta formata da oggetti che rappresentano le misure richieste dall'utente e il metodo termina quindi restituendo tale lista, pronta per essere impiegata nella realizzazione del grafico.

Il metodo callApiForDownload implementa invece le funzionalità necessarie ad effettuare la richiesta al server di inviare all'email dell'utente il file CSV contenente i dati relativi alle misure. Il suo funzionamento, costituito sostanzialmente dalla richiesta di tipo GET, è simile a quello del metodo processData riportato in figura 26, ma si differenzia, oltre che per la composizione dell'URL, anche per il diverso tipo di dato restituito, una stringa, che rappresenta il messaggio inviato dal server contenente l'esito della procedura attivata; anche in questo caso, il tipo di ritorno risulta wrappato nel tipo di dato Future, in quanto anche questo metodo dà luogo ad operazioni asincrone che devono essere segnalate al framework per gestirne correttamente l'esecuzione.
Per effettuare la richiesta, il metodo necessita di conoscere l'indirizzo e-mail dell'utente, in quanto quest'ultimo rappresenta un parametro che va fornito al server per portare a termine correttamente l'operazione di invio. L'utente, al di là di questa funzionalità, inserisce il suo indirizzo ogni volta che entra nell'applicazione, come descritto in precedenza, sia per effettuare il login che per registrarsi; è quindi necessario implementare un meccanismo che renda disponibile tale valore anche al metodo callApiForDownload. A tal proposito, è stato utilizzato il package Provider, le cui classi forniscono dei metodi che permettono di rendere accessibile lo stato di una particolare classe a tutte le componenti dell'applicazione, aggiornandone il valore delle proprietà nel caso in cui queste vengano modificate; tale classe, nell'applicazione del sistema di monitoraggio, è denominata Email e parte della sua implementazione è riportata in figura 27.

```
class Email extends ChangeNotifier {
  String email = '';

  void changeEmail(String newEmail) {
    this.email = newEmail;
    notifyListeners();
  }
}
```

Figura 27 - classe Email

La classe eredita, tramite la parola chiave extends, dalla classe definita dal modulo Provider chiamata ChangeNotifier, la quale attiva un listener interno sullo stato della classe nel momento in cui viene istanziata.

Quando l'utente termina l'inserimento dell'e-mail in fase di autenticazione, viene richiamato, tramite il metodo of del modulo Provider, il metodo changeEmail, al quale viene passata come parametro la stringa che codifica l'indirizzo e-mail inserito. Al suo interno, dopo aver aggiornato il valore della proprietà email, viene richiamato il metodo della classe ChangeNotifier notifyListener, il quale notifica il framework e i listener associati riguardo la variazione dello stato della classe.

Il valore risulta quindi accessibile anche all'esterno, in particolare nel metodo callApiForDownload. Per recuperare l'indirizzo e-mail è necessario richiamare, sempre attraverso il metodo of della libreria Provider, il metodo getEmail della classe, qui non riportato, che restituisce il valore della variabile email, ovvero l'indirizzo dell'utente inserito da quest'ultimo in fase di autenticazione.

5.5.2 Interfaccia grafica

In questa sezione verranno analizzate le schermate che compongono l'applicazione.

5.5.2.1 SelectionScreen

La classe selectionScreen implementa l'interfaccia tramite cui l'utente può selezionare i parametri che determinano il grafico che vuole visualizzare o i dati da inserire nel file che verrà eventualmente inviato tramite e-mail. La figura 28 mostra l'interfaccia della schermata.

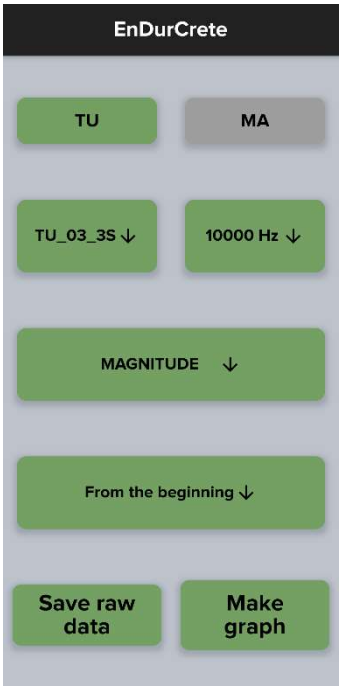

Figura 28 – selectionScreen

Ad ogni componente è associata una proprietà della classe, ad esclusione della coppia di widget collocata nella parte superiore che ne individua solamente una. La variazione dello stato di un widget, che segue l'interazione dell'utente, si riflette sul valore della variabile corrispondente. Richiamando il metodo setState, l'interfaccia viene inoltre aggiornata dinamicamente e, nel momento in cui l'utente decida di procedere alla visualizzazione del grafico o alla richiesta di invio del file, i valori delle proprietà, che individuano quindi la selezione effettuata dall'utente, vengono assegnati alle proprietà analoghe definite nelle altre classi, in modo che da quest'ultime possa essere eseguita la richiesta determinata dai parametri specificati.

SelectionScreen eredita inoltre dal Mixin AutomaticaKeepAliveClientMixin, il cui metodo wantKeepAlive, se sovrascritto, fa sì che il widget non venga distrutto e rimosso dall'albero nel momento in cui l'utente si sposta in un'altra schermata; in questo modo, lo stato del widget resta preservato e, quando la schermata di selezione viene nuovamente caricata, le sue componenti risultano settate secondo l'ultima scelta effettuata, a meno che l'applicazione sia stata chiusa forzatamente.

Le componenti collocate nella parte superiore sono realizzate tramite dei GestureDetector, widget in grado di rilevare un'interazione e di associarle una funzione. Cliccando su uno di essi, l'utente può quindi selezionare l'installazione che vuole visionare e, contestualmente, i colori dei widget e il nome delle schede disponibili variano, in modo che sia evidenziata

l'installazione scelta e siano presenti, nel widget relativo alle schede di acquisizione, solamente quelle appartenenti al sito selezionato.

I 4 widget centrali, invece, sono implementati tramite dei DropdownChoice, ovvero dei bottoni che, quando vengono cliccati, mostrano una lista di valori tra cui l'utente deve scegliere; quello selezionato costituisce poi il testo del bottone stesso. La lista relativa alle schede di acquisizione varia in modo dinamico in base all'installazione selezionata dall'utente e mostra i nomi delle schede attive nel sito scelto. La seconda è invece statica e contiene tutte le frequenze in cui sono state effettuate le misure, oltre al valore all che le identifica tutte. Il terzo widget consente la scelta del parametro che si vuole visualizzare, codificato nelle stringhe "magnitude", "phase", "real", "imaginary" e "temperature", mentre l'ultimo Dropdown implementa la selezione del periodo di osservazione; vi sono quattro scelte statiche, ovvero "from the beginning", "1 week", "2 week" e "one month", e una quinta opzione, "choose your period", la quale, se selezionata, porta l'utente ad una schermata in cui è possibile personalizzare a piacimento la data e l'orario di osservazione. Ques'ultima è realizzata attraverso tre widget, posti secondo un layout verticale, di cui due, di tipo DatePicker, consentono la selezione delle date di inizio e di fine, mentre il terzo implementa la scelta dell'orario, i cui possibili valori sono definiti secondo soglie di trenta minuti. L'interfaccia, con delle selezioni esemplificative, è riportata in figura 29.

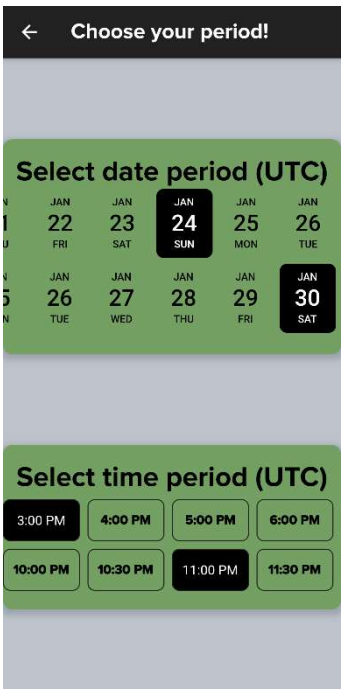

Figura 29 - Selezione del periodo di osservazione

Le ultime due componenti della schermata di selezione sono costituite da due bottoni, implementati attraverso dei widget denominati FlatButton; la funzione associata al bottone collocato a sinistra attiva la schermata implementata dalla classe saveRawDataScreen, all'interno della quale è possibile effettuare la richiesta al server di invio del file CSV, mentre il bottone destro, nel momento in cui viene cliccato, ridirige l'utente alla schermata di caricamento, implementata dalla classe LoadingScreen, successivamente alla quale viene mostrato il grafico richiesto. Entrambe le funzioni effettuano la navigazione alle schermate successive mediante la classe Navigator e il suo metodo push, passando alle classi che vengono richiamate i valori delle variabili associate ai widget così come sono state selezionate dall'utente.

5.5.2.2 SaveRawDataScreen

Le proprietà della classe SaveRawDataScreen rappresentano i parametri che identificano la richiesta da effettuare al server e vengono valorizzate all'atto della chiamata.

L'interfaccia della schermata, riportata in figura 30, è costituita da due componenti; la prima, un widget di tipo Switch, associa il suo stato ad una variabile booleana, che determina la richiesta dell'utente di ricevere il file contenente le misure compiute da tutte le schede di acquisizione. Nella figura, il widget risulta selezionato e, di conseguenza, alla proprietà della classe che definisce la scheda da richiedere viene assegnata la stringa "all".

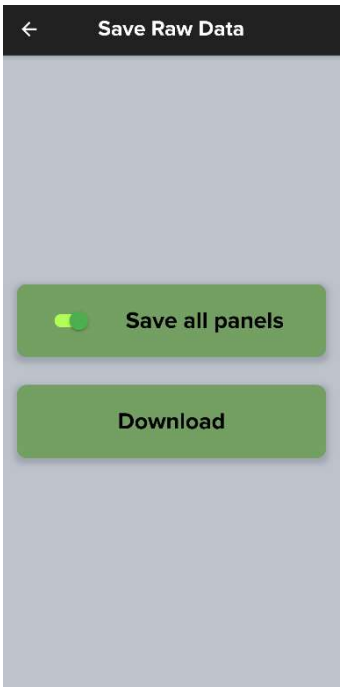

Figura 30 – SaveRawDataScreen

Poi, tramite il bottone Download, anche in questo caso implementato con un FlatButton, viene istanziato un oggetto di tipo DownloadService e viene richiamato il suo metodo callApiForDownload, al quale vengono passati i parametri che identificano la richiesta codificati nelle proprietà della classe; la stringa restituita dal metodo, contenente il messaggio che individua l'esito dell'operazione, viene mostrata all'utente tramite un AlertDialog.

5.5.2.3 LoadingScreen

La classe LoadingScreen implementa una semplice schermata, costituita unicamente da un widget, in particolare uno SpinKitDoubleBounce, che mostra un'animazione raffigurante un caricamento.

La sua funzionalità consiste nell'effettuare la richiesta al server e configurare correttamente la classe che realizza l'interfaccia contenente il grafico, intercettando eventuali errori verificatisi nel processo. Le proprietà della classe sono sostanzialmente le stesse definite nelle classi SelectionScreen e SaveRawDataScreen; identificano i parametri selezionati dall'utente e che devono caratterizzare la richiesta.

Vi sono due proprietà aggiuntive: una, denominata filter, è di tipo booleano e risulta vera se il grafico che si vuole visualizzare deve essere filtrato; l'altra, chiamata invece filterTime, è di tipo intero e identifica il valore della finestra temporale su cui applicare la media mobile. La classe LoadingScreen, infatti, può essere istanziata sia successivamente alla schermata di selezione iniziale, la quale assegnerà alle due variabili i valori false e 0, che alla richiesta di filtrare il grafico, a seguito della quale le variabili assumeranno i valori true e il valore dell'intervallo temporale specificato dall'utente. È quindi necessario, al fine di richiedere al server i dati corretti, specificare attraverso i valori attribuiti alle due variabili il tipo di grafico che l'utente ha richiesto.

Viene effettuato l'override del metodo initState, definito per tutti gli Stateful widget ed eseguito ogni volta che il widget stesso viene istanziato. Al suo interno, viene richiamato getData, un metodo privato della classe; è asincrono e nella sua intestazione figura quindi la parola chiave async. Le funzionalità di comunicazione con la classe Download sono state inserite all'interno di questo metodo e non direttamente in initState in quanto quest'ultimo, essendo un metodo ereditato, non può contenere direttamente operazioni asincrone.

Il metodo getData inizia istanziando un oggetto di tipo Dowload e assegnando alle sue proprietà i valori selezionati dall'utente memorizzati a loro volta nelle proprietà della classe LoadingScreen; all'interno di un costrutto try catch, poi, viene effettuata la richiesta. Le righe di codice che la implementano sono riportate in figura 31.

```
List<Measure> data = !widget.filter
    ? await download.callApiForGraphNoFilter()
    : await download.callApiForGraphFiltered(
        filterTime: widget.filterTime,
      );

if (data.isEmpty) {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => ErrorScreen(
        text: 'No data returned. Check your parameters and try again.',),),);
} else {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => GraphDisplay(
        seriesList: GraphSetting.dataForGraph(data),
        attribute: widget.attribute,
        panel: widget.panel,
        endDate: widget.endDate,
        startDate: widget.startDate,
        endHour: widget.endHour,
        startHour: widget.startHour,
        frequency: widget.frequency,),),);
}
```

Figura 31 - Esecuzione della richiesta

La prima istruzione richiama il metodo della classe Download che esegue la richiesta; il metodo corretto viene individuato tramite un operatore ternario, basato sul valore della variabile filter; nel caso in cui l'utente abbia richiesto un grafico filtrato, viene passato al metodo corrispondente il valore della finestra temporale, in modo che questo sia in grado di comporre correttamente l'URL dell'API.

Entrambi i metodi restituiscono una lista di oggetti Measure, sulla cui dimensione viene effettuato un controllo condizionale. Se la struttura dati ottenuta risulta vuota, viene mostrata la schermata di errore con un messaggio che invita l'utente a controllare i parametri selezionati; altrimenti, sfruttando il metodo push della classe Navigator, l'utente viene ridiretto all'interfaccia in cui il grafico viene mostrato, assegnando alle proprietà della classe tutti i dati relativi ai parametri selezionati dall'utente, necessari a comporre correttamente la schermata. Le variabili sono precedute dalla parola chiave widget, in quanto, come accade per tutti gli Stateful widget, i metodi sono contenuti all'interno di una classe che implementa lo stato del widget e non all'interno del widget stesso.

La variabile seriesList viene inizializzata dal valore restituito dal metodo statico della classe GraphService datForGraph, il quale, passato come parametro la lista di valori che si vuole visualizzare, restituisce una lista di Series, un oggetto definito nel package charts_flutter, le cui

componenti sono state utilizzate per realizzare i grafici, necessario alla creazione del grafico stesso.

Vi sono due clausole catch che seguono il blocco try; la prima individua l'eccezione denominata SocketException, che si verifica in seguito a problemi di connessione, sia lato client che lato server, mentre la seconda è generica e gestisce tutte le eccezioni che si possono verificare, come, ad esempio, quella lanciata dal metodo processData se il server non invia una risposta entro cinque secondi. In tutti i casi, viene sempre mostrata la schermata di errore, implementata dalla classe errorScreen, la cui interfaccia è mostrata in figura 32.

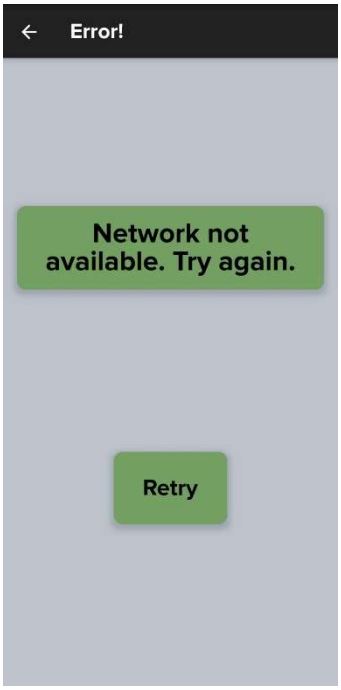

Figura 32 - ErrorScreen

Il messaggio visibile è quello che si ottiene se si prova ad effettuare una richiesta senza avere una connessione ad Internet attiva sul proprio smartphone. Tramite il bottone, implementato tramite un FlatButton, è possibile ritornare alla schermata di selezione iniziale.

5.5.2.4 GraphScreen

La classe GraphScreen realizza l'interfaccia nella quale viene mostrato il grafico richiesto dall'utente. È uno stateless widget e le sue proprietà sono quelle inizializzate dal metodo getData riportato in figura 31; un'ulteriore proprietà è costituita da un oggetto di tipo GraphService, istanziato internamente al costruttore, i cui metodi vengono utilizzati per gestire dinamicamente la creazione del grafico. La schermata è riportata in figura 33.

La stringa che costituisce il titolo varia in base ai parametri richiesti dall'utente, specificando la scheda di acquisizione, l'attributo e l'unità di misura dei valori sull'asse delle ordinate. Il

widget che realizza il grafico, importato dalla libreria charts_flutter, è di tipo TimeSeriesChart, utilizzato per implementare grafici in cui l'asse delle ascisse contiene dei valori temporali.

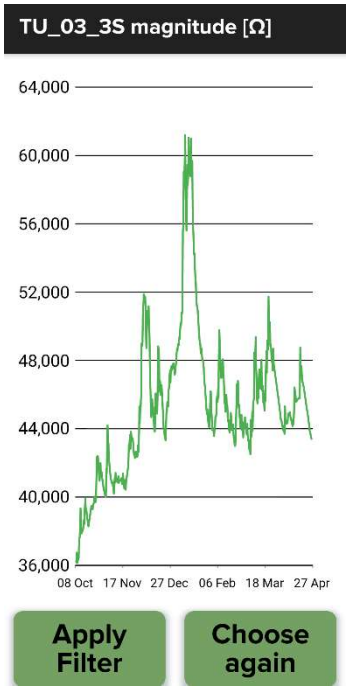

Figura 33 - GraphScreen

Il primo parametro con cui il widget deve essere inizializzato è la lista di oggetti Series che contengono i dati da visualizzare, memorizzati nella proprietà della classe seriesList, definita all'atto della chiamata da parte del metodo getData. L'aspetto del grafico può essere poi personalizzato tramite la riscrittura delle proprietà del widget, come ad esempio quelle dell'oggetto domainAxis, che identifica l'asse delle ascisse. In particolare, sono state ridefinite le sue proprietà tickProviderSpec, che determina quali etichette vanno posizionate sull'asse, e tickFormatterSpec, che identifica invece la formattazione che deve caratterizzare le etichette stesse. Il numero di etichette presenti sull'asse delle ascisse viene fornito alla proprietà relativa mediante il metodo della classe GraphService chiamato getList; quest'ultimo restituisce una lista di TickSpec, gli oggetti che possono essere assegnati alla proprietà tickProviderSpec, la cui dimensione è legata alla distanza temporale che intercorre fra i limiti dell'intervallo richiesto dall'utente; se, ad esempio, viene specificato un intervallo di tre giorni, vengono mostrate tre etichette, una per giorno, se l'intervallo è invece di 5 giorni, ne vengono posizionate 4, in modo che il grafico resti comprensibile ma non siano presenti informazioni ridondanti. Il numero massimo di etichette è pari a 6, come mostrato in figura; in tutte le casistiche, comunque, risultano distribuite equamente e separate da intervalli regolari.

La formattazione è invece determinata dal risultato del metodo getFormatter della classe GraphService; vi sono due tipi di formati possibili, anche in questo caso determinati dalla distanza che intercorre fra i limiti dell'intervallo selezionato dall'utente: se la differenza fra la data di fine e quella di inizio è minore di un giorno, vengono mostrati gli orari in cui sono avvenute le misure, secondo il formato "HH:mm", altrimenti la formattazione è quella visibile in figura, ovvero "dd MMM".
La schermata termina con i due bottoni situati nella parte inferiore; quello a destra consente all'utente di ritornare all'interfaccia di selezione, dalla quale può eventualmente scegliere nuovi parametri e richiedere la visualizzazione del grafico corrispondente, quello a sinistra attiva invece una finestra modale, al cui interno è possibile richiedere il filtraggio del grafico. L'interfaccia del widget che la implementa è riportata in figura 34.

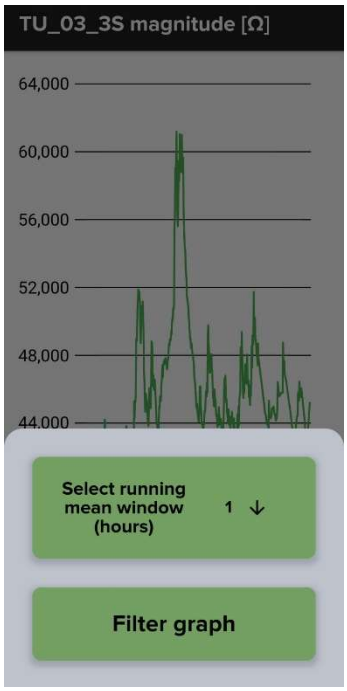

Figura 34 - Richiesta di filtraggio del grafico

È costituita da un DropdownChoice, contenente i numeri da 1 a 36 che identificano la finestra temporale oraria su cui applicare la media mobile, e da un bottone, implementato con un FlatButton. Alla sua proprietà onClick è associata una funzione, la quale, richiamando l'interfaccia di caricamento, permette la visualizzazione del grafico filtrato. Le proprietà filter e filterTime della classe LoadingScreen vengono inizializzate con i valori true, ad indicare appunto la

richiesta di un grafico filtrato, e con la stringa relativa alla finestra temporale selezionata dall'utente fra quelle disponibili nella lista associata al widget Dropdown.

In figura 35 è riportata la schermata che si visualizza nel momento in cui si richiede il filtraggio del grafico mostrato in figura 33 secondo una finestra temporale di 36 ore, il massimo selezionabile.

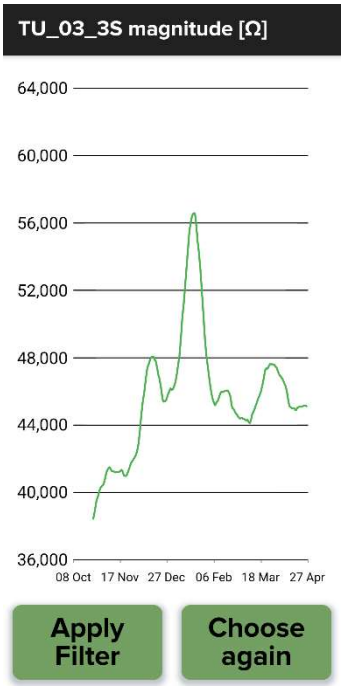

Figura 35 - Grafico filtrato

CAPITOLO 6

Conclusione

Il lavoro presentato in questo elaborato di tesi ha portato allo sviluppo di un sistema di monitoraggio di elementi in calcestruzzo, in particolare dell'architettura software che gestisce l'esecuzione automatica delle misure, il salvataggio dei dati ottenuti e la loro fruizione da parte degli utenti, sia sottoforma di grafici che attraverso dei file opportunamente strutturati. Le grandezze oggetto di monitoraggio sono l'impedenza elettrica, della quale vengono analizzati gli andamenti di modulo, fase, parte reale e parte immaginaria, e temperatura, entrambe misurate attraverso dei sensori, posizionati sui campioni di calcestruzzo, con cui i software interagiscono per effettuare le misurazioni.

Al momento della stesura dell'elaborato, il sistema di monitoraggio risulta attivo da più di sei mesi, durante i quali i software sviluppati hanno continuato a funzionare come previsto, senza interruzioni; i database risultano popolati con un gran numero di dati e di valori, il cui andamento è facilmente fruibile grazie ai grafici realizzati all'interno dell'applicazione mobile e che rappresentano un importante elemento di valutazione nell'analisi dei materiali con cui sono stati realizzati i campioni di calcestruzzo monitorati. Gli obiettivi funzionali del progetto, identificati nel Capitolo 3 dell'elaborato, sono stati quindi soddisfatti, anche se il sistema, nel suo complesso, risulta migliorabile e alcune delle sue funzionalità espandibili.

Innanzitutto, l'applicazione mobile è attualmente disponibile solo per smartphone Android, anche se il framework Flutter, tramite cui è stata realizzata, permette di compilare lo stesso codice per tutti i sistemi operativi mobile; creare l'applicativo per uno smartphone iOS, infatti, richiede delle componenti hardware specifiche, su tutte un computer Mac, non disponibili al momento dello sviluppo. Un miglioramento che si potrebbe apportare al sistema può quindi consistere proprio nella realizzazione dell'eseguibile per iOS, un'operazione che di per sé necessita di pochi passaggi, in quanto il codice risulta già realizzato, ma che vanno eseguiti sulle macchine corrette.

Un ulteriore potenziamento della solidità del sistema di monitoraggio può essere effettuato sviluppando un meccanismo in grado di confrontare costantemente i database locali, salvati sulle macchine da cui vengono eseguite le misure, e il database remoto, quello da cui l'applicazione mobile, tramite il server, recupera i dati da mostrare all'utente.

Nel corso della trattazione, infatti, si è fatto riferimento ai due database locali, gestiti attraverso il DBMS Sqlite, considerandoli come istanze di backup, in cui i dati vengono salvati anche in caso di assenza di connessione internet e quindi di impossibilità di comunicazione con il database remoto. Non esiste, tuttavia, un meccanismo che implementi l'operazione di aggiornamento di quest'ultimo nel caso in cui ci sia un'effettiva discrepanza fra i dati contenuti nelle istanze; se, da una parte, i dati mancanti risultano comunque accessibili, in quanto salvati e consultabili nelle macchine presso cui risiedono i database locali, risulterebbe più efficiente avere tutti i dati anche all'interno del database remoto, del quale sono già state implementate e testate le modalità di interrogazione necessarie.

Per quanto riguarda l'applicazione mobile, inoltre, una funzionalità che potrebbe essere implementata per arricchire le opzioni già presenti è rappresentata dalla possibilità di visualizzare un grafico contenente le curve di più attributi, ognuno caratterizzato dal proprio asse delle ordinate. Ciò fornirebbe agli utenti uno strumento per confrontare l'andamento di grandezze diverse e individuarne eventuali correlazioni.

Il sistema di monitoraggio è un progetto complesso e sicuramente migliorabile, sotto molti aspetti, ma lo stato attuale della realizzazione, descritto in questo elaborato, rappresenta un punto di partenza affidabile e funzionale, nonché già efficientemente operativo.

Bibliografia

[1] http://www.endurcrete.eu – consultato a marzo 2021

[2] https://it.wikipedia.org/wiki/International_Protection – consultato a marzo 2021

[3] https://www.arduino.cc/en/Guide/Introduction - consultato a marzo 2021

[4] https://it.wikipedia.org/wiki/Usabilità - consultato a marzo 2021

[5] http://www.python.it/about/ - consultato ad aprile 2021

[6] https://www.python.org/dev/peps/pep-0020/ - consultato ad aprile 2021

[7] https://www.html.it/pag/15610/linterprete-python-e-lidle/ - consultato ad aprile 2021

[8] https://flutter.dev/showcase - consultato ad aprile 2021

[9] https://flutter.dev/docs/resources/architectural-overview - consultato ad aprile 2021

[10] https://dart.dev/overview - consultato ad aprile 2021

[11] https://www.sqlite.org/zeroconf.html - consultato ad aprile 2021

[12] https://it.wikipedia.org/wiki/Cloud_computing - consultato ad aprile 2021

[13] https://www.geekwire.com/2021/amazon-web-services-posts-record-13-5b-profits-2020-andy-jassys-aws-swan-song/ - consultato ad aprile 2021

[14] https://aws.amazon.com/it/what-is-aws/ - consultato ad aprile 2021

[15] https://github.com/spyder-ide/spyder - consultato ad aprile 2021