



**UNIVERSITÀ
DI TORINO**

Università degli Studi di Torino

Corso di Laurea in Informatica

**Sviluppo di un sistema per l'analisi di dati
di lavorazioni aziendali**

Tesi di Laurea

Relatrice:

Baroglio Cristina

Candidato:

Eusebio Bergò Stefano

Matricola 952388

Anno Accademico 2022/2023

DICHIARAZIONE DI ORIGINALITÀ

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.

Indice

DICHIARAZIONE DI ORIGINALITÀ	2
Indice	2
1 Introduzione	4
Contestualizzazione del Progetto	4
Motivazioni e Rilevanza	4
Struttura della Tesi	4
2 Analisi del progetto	5
Dispositivi di Raccolta Dati	6
3 Configurazione e Gestione dell'Applicazione	8
Funzionalità Chiave del Sistema di Configurazioni	9
Strutture delle Configurazioni	10
Funzionamento del configuratore	13
Esempio di Utilizzo	15
Configurazione delle Funzionalità	15
Struttura dell'Applicazione	15
4 Sviluppo dell'Applicazione	16
Architettura Generale	16
Sviluppo del Front-end	17
Implementazione ed Utilizzo delle Configurazioni di Sistema	17
Struttura dell'applicazione	18
Sviluppo futuro	27
5 Utilizzo e Applicazioni	28
Perardi	28
Lab Green	30
Acquedotto di Bienca	31
6 Tecnologie e Metodologie	33
Metodologia di Sviluppo	33
Tecnologie Utilizzate	33
Flutter/Dart	34
Golang	34

S.O.L.I.D.	34
Architettura Esagonale	35
Database Non Relazionale	36
MQTT	36
Docker e Kubernetes	36
GitLab	37
Esempio di Tecnologia e Metodologia in Azione	37
7 Conclusioni	38
Descrizione	38
Risultati del Progetto	38
Lezioni Apprese	38
Prossime Tappe	38
8 Bibliografia	39

1 Introduzione

Contestualizzazione del Progetto

L'aumento dei prezzi per la corrente elettrica ed il gas hanno portato le aziende a riconoscere l'importanza della gestione efficiente dei consumi. L'obiettivo principale del progetto è stato lo sviluppo di un'applicazione innovativa che consenta alle aziende interessate di visualizzare e analizzare in modo efficiente i dati relativi alle loro attività di lavorazione. Questo progetto è stato condotto in risposta all'esigenza delle aziende di migliorare la gestione dei dati di consumo e di rendere più accessibili le informazioni critiche.

In questo contesto il progetto di sviluppo dell'applicazione per la visualizzazione e l'analisi dei dati assume un'importanza cruciale per migliorare l'efficienza operativa, facilitare la presa di decisioni basate sui dati e garantire una gestione più efficace delle risorse all'interno dell'azienda. Nel corso di questa tesi esploreremo in dettaglio come questa applicazione è stata sviluppata, implementata e utilizzata per migliorare il funzionamento di alcuni impianti.

Motivazioni e Rilevanza

La scelta di affrontare questo progetto è stata guidata da diverse motivazioni. In primo luogo la possibilità di sviluppare un'applicazione con un impatto importante sulle aziende che la utilizzano ed anche la possibilità di apprendere l'utilizzo di nuove tecnologie. Inoltre, la necessità di adattare l'applicazione per servire diverse aziende clienti con esigenze e modelli di business diversi, ha reso il progetto ancora più stimolante.

La creazione di un sistema di configurazioni codeless per personalizzare l'applicazione è stata un elemento chiave in questo progetto.

L'importanza di questo progetto risiede nel fatto che un'applicazione efficiente per la visualizzazione e l'analisi dei dati relativi a produzioni e consumi può consentire di prendere decisioni più informate, migliorare la produttività e, in ultima analisi, aumentare la competitività nel mercato.

Struttura della Tesi

La presente tesi è organizzata in modo da approfondire gli aspetti più significativi del progetto.

Nel capitolo 2, verrà fornita una panoramica del contesto aziendale, i dispositivi di raccolta dati utilizzati e le tecnologie adottate per la gestione dei dati. Nel capitolo 3, verrà esaminato il funzionamento del sistema di configurazioni e dell'applicazione per l'editing delle

configurazioni. Il capitolo 4 illustrerà il processo di sviluppo dell'applicazione, con un focus specifico sul front-end e sull'utilizzo del sistema di configurazioni. Il capitolo 5 porterà in esempio come l'applicazione viene utilizzata nelle aziende cliente e i benefici derivanti dal suo impiego. Il capitolo 6 analizzerà le tecnologie e le metodologie utilizzate durante lo sviluppo. Infine, nel capitolo 7, verranno presentate le conclusioni del progetto, i contributi personali e le prospettive future del progetto.

In questo modo, questa tesi si propone di fornire una visione completa e dettagliata del processo di sviluppo, implementazione e utilizzo dell'applicazione per la visualizzazione dei dati aziendali.

2 Analisi del progetto

Lo scopo di questa attività di stage è stato l'implementazione della parte di front-end per un'applicazione sviluppata dalla Robson.

L'azienda ne aveva sviluppato una versione basilare per poter accedere ai dati forniti dal back-end. All'inizio della mia attività in azienda era già presente una consolidata struttura per il back-end e la raccolta dati.

Un aspetto importante richiesto da questa applicazione è la scalabilità intesa come la possibilità di essere utilizzata da molte aziende cliente, ognuna con le proprie richieste specifiche, senza impattare in modo negativo sullo sviluppo dell'applicazione.

Questo punto viene analizzato più in dettaglio successivamente ma è quello che ha portato all'utilizzo di una struttura dinamica per il front-end che sfrutta dei file di configurazione presenti sul server cloud utilizzato dalla Robson. Per utilizzare questo sistema è stata necessaria la creazione di un'altra applicazione che permettesse di creare e modificare questi file di configurazione in modo più semplice ed efficiente.

Per quanto concerne la sicurezza mi è stato richiesto che ogni componente del front-end fosse visibile solo agli utenti che posseggono un determinato permesso; la stessa cosa mi è stata richiesta per quanto riguarda l'utilizzo di particolari componenti grafici da parte dell'utente.

Al fine di comprendere meglio questa tesi, introduco ora alcune delle aziende clienti in modo da poter contestualizzare eventuali esempi.

- **Luxottica Group**¹: È un'azienda italiana specializzata nella produzione e nel commercio di occhiali. È presente in oltre 150 paesi sparsi per tutto il mondo ed è la più grande

¹ "Luxottica - Wikipedia." Ultimo accesso ottobre 3, 2023. <https://it.wikipedia.org/wiki/Luxottica>.

produttrice mondiale di montature per occhiali da vista e da sole. La sede è situata nella città di Milano.

La richiesta iniziale è stata la possibilità di monitorare un sistema di ventilazione utilizzato per asciugare la verniciatura sulle montature degli occhiali in uno degli impianti dell'azienda.

- **Perardi & Gresino²**: È un'azienda di lavorazioni e costruzioni meccaniche che opera nel settore della componentistica automotive con focus sui settori industriali e nel campo dei riduttori compreso l'impiego per l'energia eolica e solare.

Il motivo originale per l'utilizzo di questa applicazione è stata la necessità di monitorare un sistema per la produzione di aria compressa ed i risparmi ottenuti all'interno di uno specifico impianto.

- **Istituto tecnico di Biella "Gae Aulenti"**: L'indirizzo di agraria gestisce un orto, un vigneto ed un frutteto didattici che vengono gestiti con le tecnologie 4.0.

Nella tesi e nell'applicazione, l'impianto relativo a questa realtà si chiama "Lab Green".

Il back-end di questa applicazione si compone di due parti principali: i dispositivi per la raccolta dati ed il server cloud Robson su cui sono presenti i processi per la loro elaborazione e le API con cui il front-end si interfaccia per visualizzarli e gestirli.

Dispositivi di Raccolta Dati

Uno degli aspetti chiave del progetto è la presenza di dispositivi avanzati per la raccolta dei dati all'interno degli impianti. La Robson ha già da tempo a disposizione dei dispositivi di proprietà chiamati "GRIMMY" che si occupano principalmente della lettura di sensori di campo o a bordo macchina e di interagire con della componentistica a loro collegata. I dati raccolti possono riguardare sia delle linee di produzione che delle zone di produzione energetica che valori ambientali o di sistemi presenti in azienda.

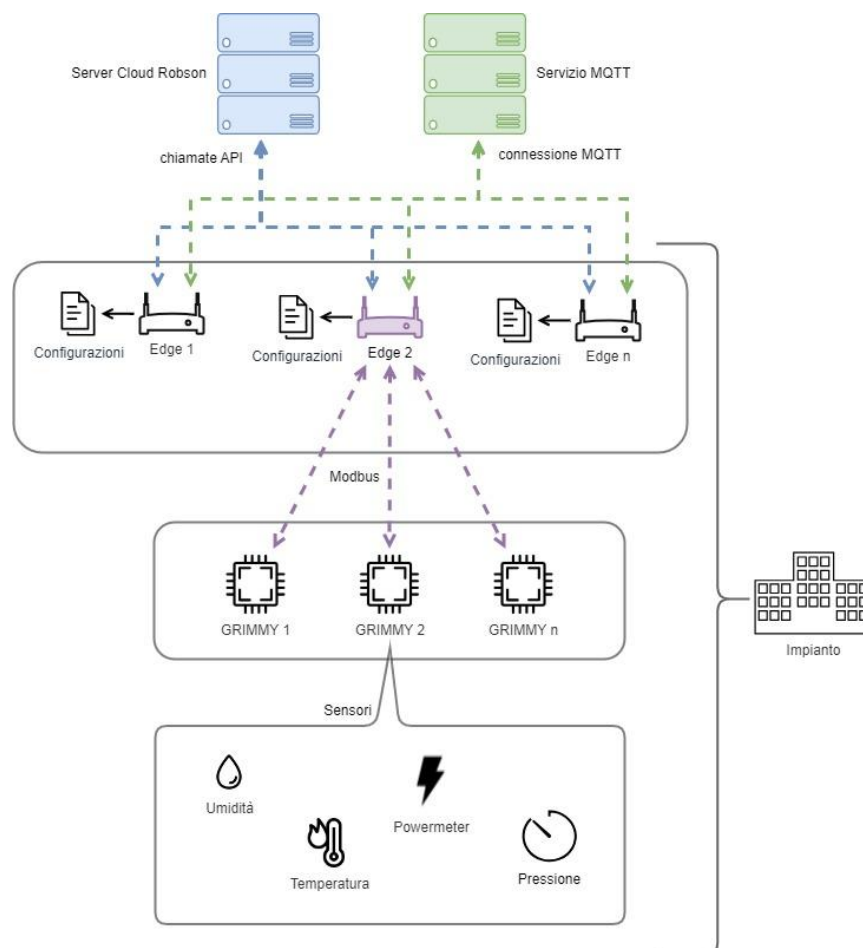
All'interno degli impianti sono presenti molteplici GRIMMY che si occupano della lettura dei sensori, ognuno di essi è assegnato ad un dispositivo chiamato "Edge" con cui comunica per esempio per l'invio dei dati campionati.

² "Perardi e Gresino S.r.l. | LinkedIn." <https://sg.linkedin.com/company/perardi-&-gresino>. Ultimo accesso: 23 ott. 2023.

Gli Edge sono dispositivi creati dalla Robson il cui componente principale è un computer in miniatura su cui vengono avviati programmi per la gestione della comunicazione con il web e per gestire la raccolta dati ed il loro flusso da e verso il web.

Gli Edge vengono spesso utilizzati per suddividere un impianto in zone di interesse, per esempio i locali dell'edificio, le diverse catene di produzione o i vari sistemi presenti.

All'avvio ogni Edge deve scaricare la sua configurazione dal cloud Robson ed utilizzarla per configurare sé stesso ed i GRIMMY di suo interesse. Il compito principale degli Edge è di richiedere periodicamente ai GRIMMY assegnatogli di effettuare il campionamento di determinati valori; una volta ottenuti i dati campionati li allinea secondo le regole definite nella sua configurazione e li bufferizza in attesa di inoltrarli verso il servizio MQTT ed il cloud Robson che si occupa della loro scrittura sul database.



Questo insieme di dispositivi ed i processi presenti sul cloud Robson sono progettati per svolgere cinque compiti principali:

1. **Acquisizione dati:** Robson installa una vasta gamma di sensori avanzati distribuiti in punti di interesse per monitorare parametri critici come temperatura, pressione,

potenza prodotta, umidità, consumi sia elettrici che di altra natura ed altre variabili rilevanti al contesto.

2. **Allineamento dei dati:** I dati provenienti dai sensori vengono allineati per garantire che siano coerenti e confrontabili. Questo processo è cruciale per l'accuratezza delle successive elaborazioni ed analisi.
3. **Pubblicazione tramite protocollo MQTT:** I dati allineati vengono quindi pubblicati su un sistema di comunicazione cloud basato sul protocollo MQTT (Message Queuing Telemetry Transport). Questo protocollo è stato scelto per la sua efficienza nella trasmissione di dati in tempo reale.
4. **Storicizzazione tramite API REST e database non relazionale:** I dati allineati, oltre ad essere pubblicati attraverso MQTT, possono anche venire storicizzati su un database non relazionale utilizzando un servizio di API REST dedicato. Questo database è essenziale per l'archiviazione e l'accesso a lungo termine dei dati raccolti.
5. **Svolgimento di attività automatiche:** I dispositivi possono anche ricevere una richiesta di svolgere determinate attività, possono per esempio dover aprire e chiudere delle valvole sotto richieste da parte degli utenti o di sistemi automatici. Gli Edge ricevono le richieste, le elaborano e successivamente provvedono a richiedere al GRIMMY adeguato lo svolgimento dell'attività o di parte di essa.

3 Configurazione e Gestione dell'Applicazione

Come già riportato, il vincolo principale per il front-end è stata la possibilità di scalare l'utilizzo dell'applicazione a numerosi clienti, ovvero la possibilità di aggiungere nuovi impianti in modo semplice.

Per esempio per l'impianto Perardi è stato richiesto che il front-end avesse tre schermate, ognuna rappresentante un preciso contesto e contenente componenti grafici specifici. La struttura di navigazione del front-end, così come i componenti grafici di ciascuna schermata, non sono sempre le stesse tra gli impianti; questo vuol dire che l'applicativo non può definire una struttura a priori.

Per risolvere questo problema si è deciso di utilizzare un sistema già presente in azienda e di estenderlo ad un nuovo utilizzo; il sistema in questione è un servizio API presente sul cloud Robson che con l'ausilio di un database fornisce delle configurazioni ai dispositivi GRIMMY ed Edge. Il servizio espone delle API anche per la creazione e modifica di queste configurazioni.

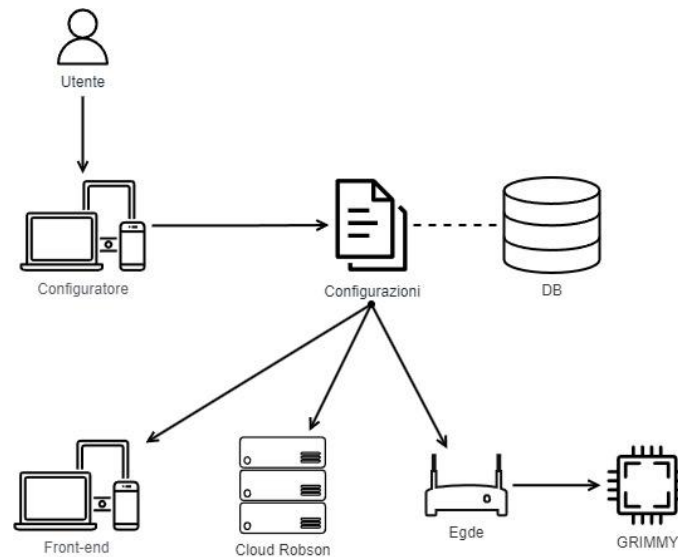
Il sistema di configurazioni è un elemento critico dell'applicazione sviluppata. Esso consente non solo di personalizzare l'applicazione per diverse aziende ed impianti ma anche di configurare in modo flessibile il comportamento dell'applicazione stessa e dei dati raccolti. Questo sistema è stato realizzato dalla Robson seguendo i principi di modularità ed utilizzando un approccio "codeless" che permette anche a utenti senza competenze di programmazione di creare e modificare le configurazioni in quanto sono rappresentate da dei file JSON.

Per l'utilizzo di questo sistema da parte del front-end si è deciso di implementare un'altra applicazione per browser web che consente di creare e modificare in modo semplice queste configurazioni. Lo scopo di questo editor è quello di ridurre il tempo che intercorre tra le richieste di modifica da parte di un'azienda e l'attuazione della modifica richiesta.

Funzionalità Chiave del Sistema di Configurazioni

Il sistema di configurazioni ora permette l'utilizzo di diverse funzionalità chiave:

- **Personalizzazione:** Permette di creare istanze uniche dell'applicazione per diverse aziende clienti, ciascuna con le proprie impostazioni e personalizzazioni.
Per "istanze" dell'applicazione si intende un set minimo e funzionante di configurazioni, sia per il front-end che per il back-end, che consente l'utilizzo dell'applicazione per un certo impianto da parte dell'azienda cliente.
- **Configurazioni delle Funzionalità:** Consente di definire e personalizzare il comportamento e le funzionalità dell'applicazione. Ad esempio, è possibile configurare i tipi di dati da raccogliere, le notifiche da inviare o le autorizzazioni degli utenti.
- **Descrizione delle Interazioni Utente:** Permette di creare una struttura dinamica dell'applicazione che determina come gli utenti navigano attraverso le pagine dell'applicazione stessa e come i dati vengono visualizzati. Questo aspetto è fondamentale per la replicabilità dell'applicazione e per garantire un'esperienza utente fluida ed efficiente.



Strutture delle Configurazioni

Tutte le configurazioni appartengono ad una tipologia, ad esempio: grafico, gauge, link, dashboard, etc.; ogni istanza di una configurazione è identificata da una chiave e dalla tipologia della configurazione. La chiave è una stringa spesso composta dal nome di un package e da un nome simbolico assegnato alla configurazione.

In un file JSON differente sono definite le strutture per molte delle tipologie di configurazione presenti. Quando una configurazione viene creata o modificata, questa deve rispettare la struttura che è stata definita per la tipologia a cui appartiene.

Di seguito viene riportata la definizione di una struttura e poi un esempio di un'istanza della configurazione che la rispetta.

Definizione di parte di una struttura:

```
"app_gauge_axis": {
  "annotations": {
    "cardinality": "**",
    "info": "struttura per personalizzare l'etichetta del gauge",
    "is_custom": true,
    "name": "app_gauge_axis_annotation"
  },
  "maximum": {
    "cardinality": "1",
    "default": 10,
    "info": "valore massimo rappresentabile dal gauge.",
    "is_custom": false,
    "name": "float32"
  }
}
```

```

    },
    "minimum": {
        "cardinality": "1",
        "default": 0,
        "info": "valore massimo rappresentabile dal gauge",
        "is_custom": false,
        "name": "float32"
    },
    "pointers": {
        "cardinality": "*",
        "info": "struttura rappresentante le lancette del gauge",
        "is_custom": true,
        "name": "app_gauge_axis_pointer"
    },
    "ranges": {
        "cardinality": "*",
        "info": "struttura per definire la suddivisione in colori del gauge",
        "is_custom": true,
        "name": "app_gauge_axis_range"
    },
    "show_labels": {
        "cardinality": "1",
        "default": false,
        "info": "flag per visualizzare i valori lungo il gauge",
        "is_custom": false,
        "name": "bool"
    },
    "show_ticks": {
        "cardinality": "1",
        "default": false,
        "info": "flag per visualizzare le tacche intermedie nella scala del gauge",
        "is_custom": false,
        "name": "bool"
    }
}
},

```

Istanza di configurazione che rispetta la struttura:

```

{
    "gauge_settings": {
        "axes": [
            {
                "annotations": [
                    {
                        "angle": 90,
                        "font_size": 16,
                        "font_weight": "w700",
                        "label_font_size": "text_1_5",
                        "label_font_weight": "w700",
                        "number_format": "###0",
                        "pattern": "{{value}} NI/min",

```

```

        "position_factor": 0.5
    },
    ],
    "maximum": 24000,
    "minimum": 0,
    "pointers": [
        {
            "type": "needle"
        }
    ],
    "ranges": [
        {
            "color": "#4caf50",
            "end_value": 12000,
            "end_width": 10,
            "start_value": 0,
            "start_width": 10
        },
        {
            "color": "#ffeb3b",
            "end_value": 20000,
            "end_width": 10,
            "start_value": 12000,
            "start_width": 10
        },
        {
            "color": "#f44336",
            "end_value": 24000,
            "end_width": 10,
            "start_value": 20000,
            "start_width": 10
        }
    ],
    "show_labels": true,
    "show_ticks": true
}

],
"title": "Air Flow",
"title_font_size": "text_1_5",
"title_font_weight": "w400",
"title_size": 15,
"title_weight": "w400"
},
"topic":
2456eec1-1d67-4999-a782-2966a93bb568/DATALOG/1298dd91-6f7b-4ba4-801a-dda96b5a4aba/de26a021-8c6c-480f-b686-8
e1b4dcbb0cb/1d454242-4e62-46d9-a807-dd10da0f409c/CR_OUTPUT_FLOW"
}

```

Funzionamento del configuratore

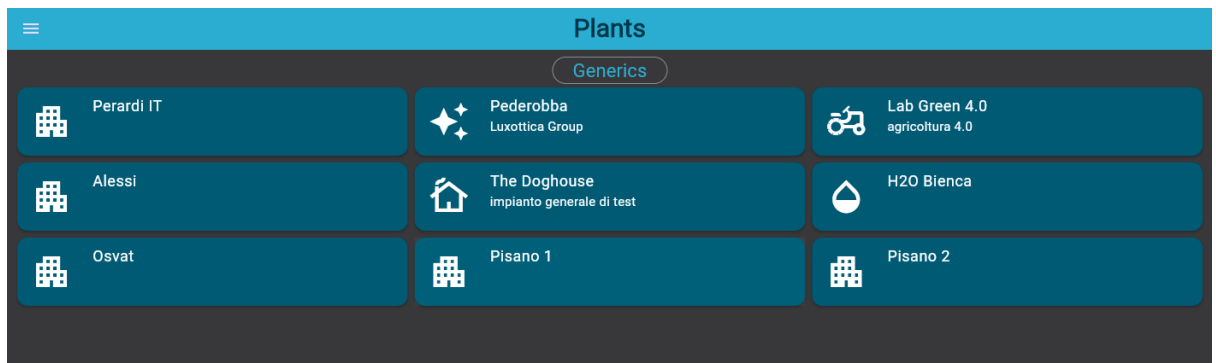
Tramite un servizio di API presente sul cloud Robson, il configuratore è in grado di leggere e scrivere le configurazioni presenti sul DB.

Ogni configurazione ha un numero di versione che viene incrementato ad ogni modifica.

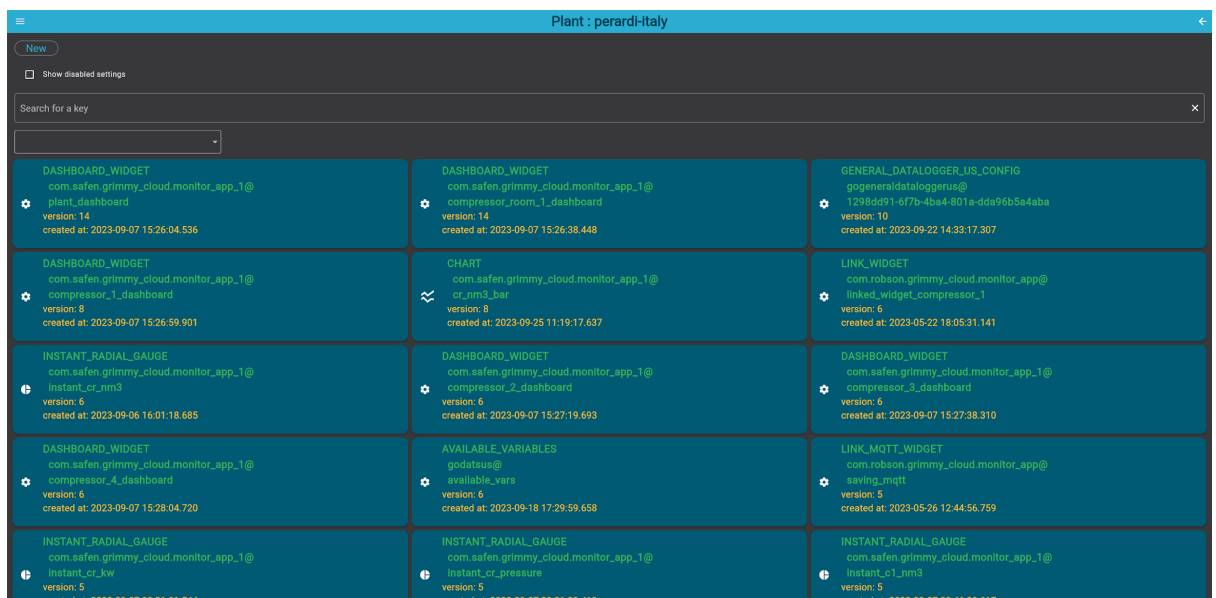
Quando viene richiesta una specifica configurazione, il servizio di API restituisce quella con il numero di versione maggiore tra quelle che sono in stato “abilitato”.

Ora verranno descritti i passi più comuni nell'utilizzo del configuratore:

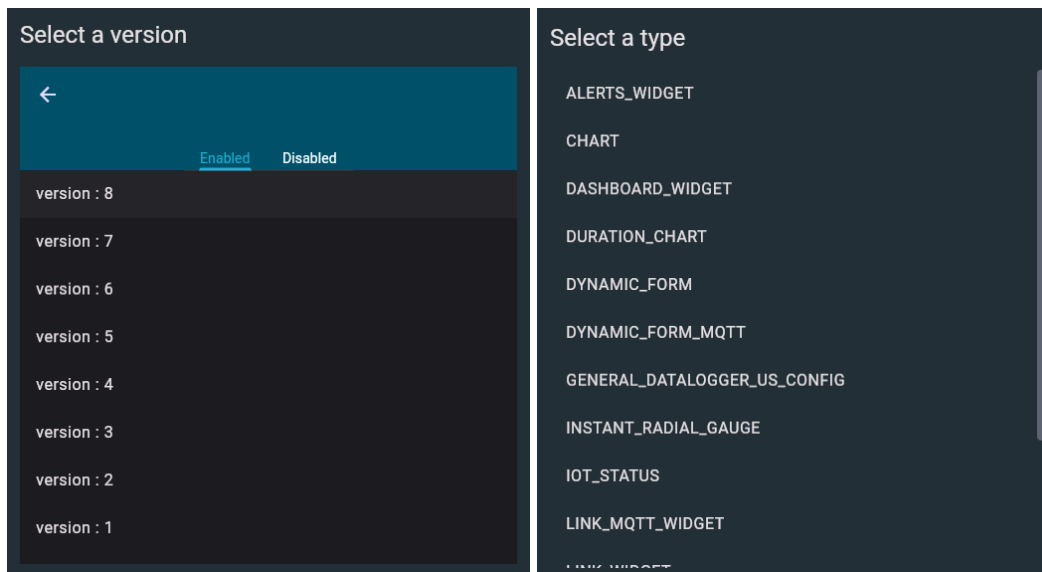
1. Dopo il login dell'utente, in configuratore richiede al cloud tutte le configurazioni di tipo “impianto” assegnate all'utente loggato in modo da popolare la schermata di scelta dell'impianto.



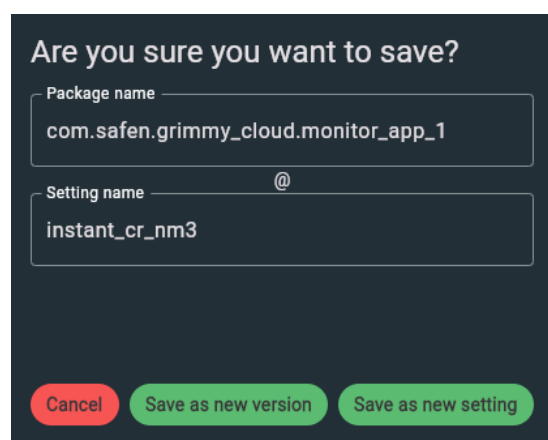
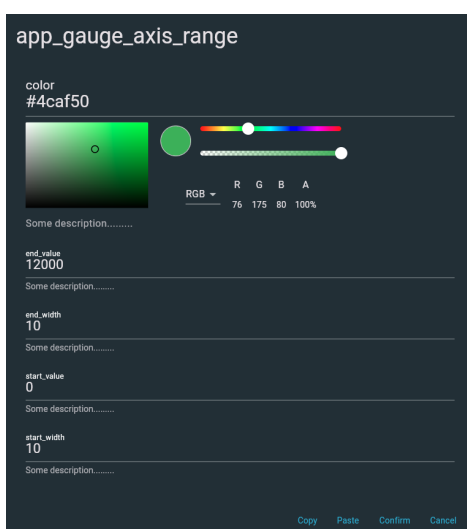
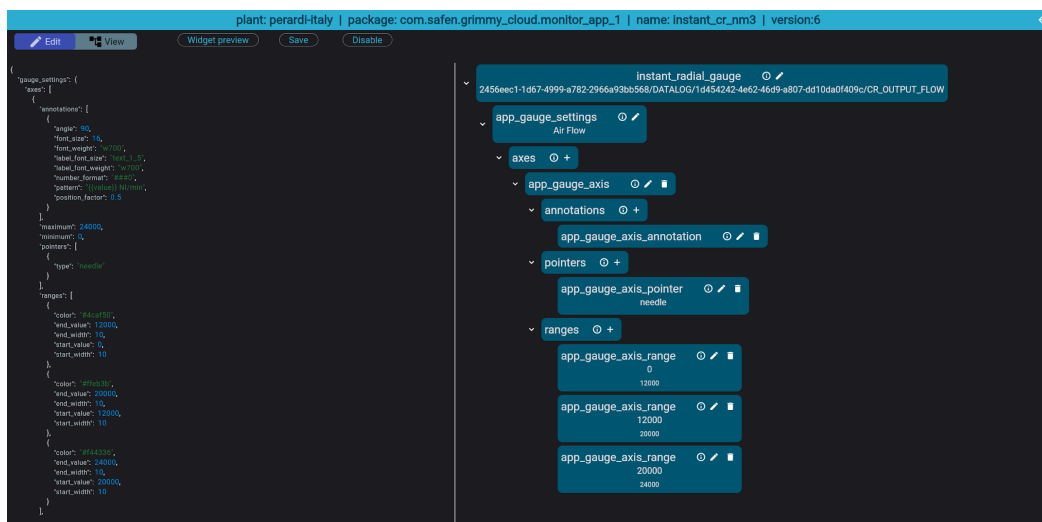
2. Quando l'utente decide di focalizzarsi su un impianto specifico provvede a richiedere tutte le configurazioni che lo riguardano e le utilizzerà per popolare la schermata successiva.



3. A questo punto l'utente potrà decidere di modificare una versione di una configurazione già esistente o di crearne una nuova.



- Una volta effettuata la scelta, l'utente viene reindirizzato all'editor della configurazione e l'utente potrà effettuare diverse azioni:



- Abilitare o disabilitare la configurazione selezionata

- Utilizzare la sezione di destra per modificare la configurazione a suo piacimento; un utente più esperto può anche utilizzare direttamente la sezione di sinistra, molto utile in caso serva modificarne il contenuto a partire da un file esterno
- Se la configurazione rispetta la struttura della sua tipologia, l'utente può decidere di salvarla. La configurazione può venire salvata in due modi:
 - come una nuova versione incrementale della configurazione aperta
 - oppure si può specificare un nuovo nome per il package o per la configurazione e salvarla come la prima versione di una nuova

Esempio di Utilizzo

Per comprendere meglio come il sistema di configurazioni è stato utilizzato, prendiamo in considerazione un esempio specifico:

Scenario: Perardi opera nell'industria manifatturiera ed è uno dei clienti che utilizza l'applicazione sviluppata. Inizialmente hanno richiesto la possibilità di monitorare i consumi energetici ed il flusso d'aria compressa prodotti da una serie di compressori.

Configurazione delle Funzionalità

Tipo di Dati da Monitorare: Nel sistema di configurazioni, è stata creata una nuova configurazione per il monitoraggio della sala compressori.

È stata specificata la tipologia di dati da raccogliere, i sensori da utilizzare, le frequenze di campionamento e le unità di misura di ogni dato.

All'avvio dell'Edge della sala compressori vengono scaricate le configurazioni rilevanti e l'Edge istruisce sé stesso ed i GRIMMY assegnatogli su come dovranno svolgere il loro lavoro.

Notifiche Personalizzate: È stata configurata una regola per l'invio automatico di notifiche quando la potenza assorbita da ciascun compressore supera una soglia specifica; questo aiuta a prevenire il sovraccarico del dispositivo.

Questa configurazione specifica verrà scaricata e gestita dall'Edge della sala compressori che provvederà ad inviare le notifiche sul servizio MQTT in caso i valori superino la soglia stabilita.

Struttura dell'Applicazione

Interfaccia Utente Personalizzata: È stata definita una struttura dell'interfaccia utente che consente agli utenti di visualizzare in modo chiaro e intuitivo i dati.

Tramite le configurazioni è stato definito che nell'impianto sono presenti un certo numero di locali adibiti alla produzione di aria compressa e per ciascun locale sono stati definiti quali compressori ne fanno parte.

Nella schermata principale di questo impianto è possibile vedere i dati relativi a consumi e flussi cumulati di tutto l'impianto, sia istantanei che storici; mentre all'interno delle schermate dei singoli locali sono presenti gli stessi dati ma inerenti al singolo locale. L'utente può poi selezionare un compressore specifico per vedere i dati relativi al singolo dispositivo.

Accesso Autorizzato: In base alle configurazioni solo il personale autorizzato può visualizzare queste informazioni garantendo la sicurezza di eventuali dati sensibili. Per esempio è stato richiesto che tramite una sezione apposita fosse possibile visualizzare le notifiche di allarme; questa sezione è resa disponibile, sempre tramite configurazioni, solo a certe tipologie di utente autorizzate.

In questo esempio, il sistema di configurazioni ha permesso a Perardi di adattare l'applicazione alle proprie esigenze specifiche, garantendo il monitoraggio efficace dei consumi e dei flussi d'aria nel loro impianto. Questa flessibilità ha reso l'applicazione altamente adattabile ad eventuali modifiche all'interno dell'impianto ed ha consentito all'azienda cliente di ottenere informazioni significative dall'utilizzo dell'applicazione.

4 Sviluppo dell'Applicazione

Architettura Generale

L'architettura dell'applicazione sviluppata è stata progettata per soddisfare le esigenze complesse di visualizzazione e analisi dei dati. L'applicazione è composta da diverse componenti chiave, tra cui:

- **Front-end:** Questa è la parte dell'applicazione con cui gli utenti interagiscono. È stata progettata per essere accessibile tramite browser web e dispositivi mobili Android, garantendo una flessibilità massima nell'accesso ai dati. È sviluppata in modo che ogni impianto è completamente personalizzabile e che sia possibile aggiungere e modificare ogni impianti molto semplicemente.
- **Raccolta ed Elaborazione dei Dati:** Questa parte è responsabile della raccolta dei dati provenienti dai dispositivi di acquisizione, dell'allineamento dei dati e dell'elaborazione iniziale. Questi dati vengono quindi trasmessi al database non relazionale tramite API REST e pubblicati sul servizio MQTT. Possono poi venire utilizzati sia dal front-end che da altri servizi del back-end.

- **Configurazioni di Sistema:** Questo modulo consente di personalizzare l'applicazione per diverse aziende clienti ed impianti e di definire le modalità di raccolta, allineamento ed elaborazione dei dati in modo semplice ed intuitivo. Il sistema di configurazioni è stato progettato per essere codeless, il che significa che può essere utilizzato anche da persone senza competenze di programmazione.
- **Distribuzione dell'Applicativo:** L'applicativo è distribuito tramite l'uso di immagini Docker³ gestite da un server cloud che utilizza Kubernetes⁴ per la gestione dei container. Questo approccio consente una distribuzione scalabile e una gestione efficace delle risorse del server.

Sviluppo del Front-end

Il cuore del progetto è il front-end, che offre agli utenti la possibilità di accedere ai dati aziendali in modo intuitivo ed efficiente. Il front-end è stato sviluppato utilizzando Flutter⁵ in modo che il codice fosse utilizzabile sia per l'utilizzo dell'applicazione su Android che su web. Alcune delle funzionalità chiave del front-end includono:

- **Autenticazione dell'Utente:** Per garantire la sicurezza dei dati, è stato implementato un sistema di autenticazione degli utenti. Gli utenti devono effettuare l'accesso con credenziali valide per accedere all'applicazione. Per questa applicazione è stato utilizzato il servizio di autenticazione fornito da Google tramite Firebase⁶ dato che l'azienda già si affidava a questo servizio.
- **Visualizzazione dei Dati Grezzi ed Elaborati:** Gli utenti possono visualizzare sia i dati grezzi, così come sono memorizzati nel database, sia i dati che sono stati elaborati attraverso altri servizi di analisi presenti sul cloud Robson. Con il termine "dati elaborati" si intende dati che sono stati sottoposti a calcoli matematici che possono variare da semplici somme a più complesse funzioni che possono includere derivate, medie, integrali, etc.. Questo permette agli utenti di avere una visione completa delle informazioni.

Implementazione ed Utilizzo delle Configurazioni di Sistema

Una delle sfide principali del progetto è stata l'implementazione del sistema di configurazioni per l'utilizzo nella parte di front-end. Le configurazioni permettono di adattare il front-end alle

³ "Docker - Wikipedia." Ultimo accesso ottobre 3, 2023. <https://it.wikipedia.org/wiki/Docker>.

⁴ "Kubernetes - Wikipedia." Ultimo accesso ottobre 3, 2023. <https://it.wikipedia.org/wiki/Kubernetes>.

⁵ "Flutter (software) - Wikipedia." Ultimo accesso ottobre 3, 2023.

[https://it.wikipedia.org/wiki/Flutter_\(software\)](https://it.wikipedia.org/wiki/Flutter_(software)).

⁶ "Firebase - Wikipedia." Ultimo accesso ottobre 3, 2023. <https://it.wikipedia.org/wiki/Firebase>.

esigenze specifiche di ciascuna azienda cliente e di definire come i dati vengono raccolti, allineati ed elaborati.

Il sistema di manipolazione delle configurazioni è stato progettato con un'interfaccia intuitiva che consente agli utenti, anche senza esperienza di programmazione, di personalizzare l'applicazione.

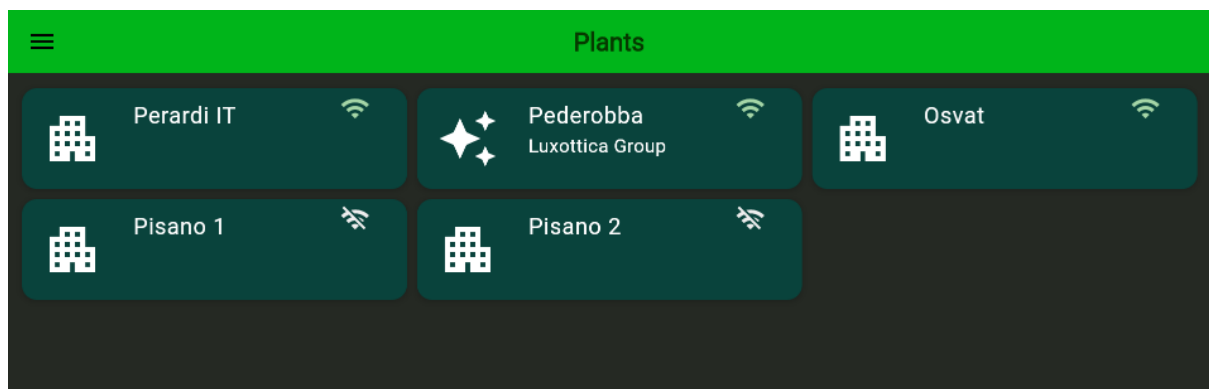
Le configurazioni sono fondamentali anche per la gestione dell'accesso degli utenti alle informazioni, la visualizzazione dei dati e la gestione della struttura dinamica dell'applicazione. Ciò significa che è possibile adattare l'esperienza utente senza dover rilasciare nuove versioni dell'applicazione.

Nel seguito di questa tesi, esploreremo come questa applicazione è stata utilizzata in alcune delle aziende clienti, i benefici ottenuti e le tecnologie specifiche utilizzate nello sviluppo.

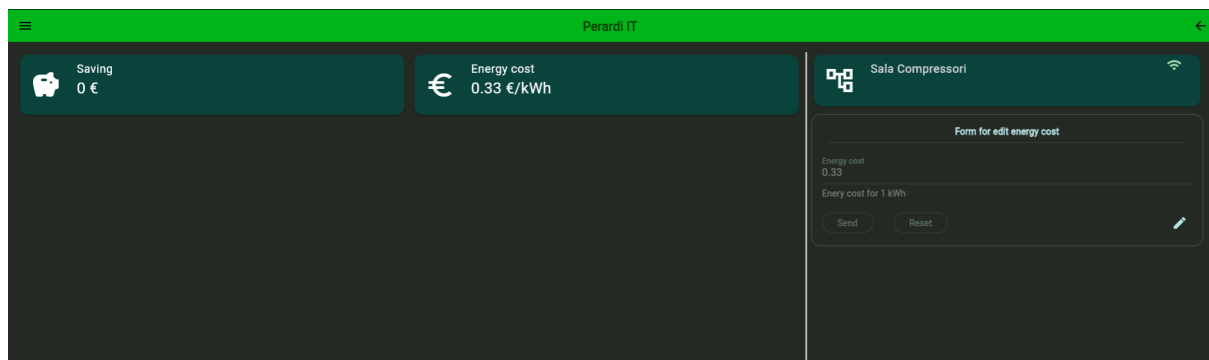
Struttura dell'applicazione

La parte di front-end si basa su un sistema di navigazione composto da:

- una pagina di login
- una pagina di scelta dell'impianto di interesse



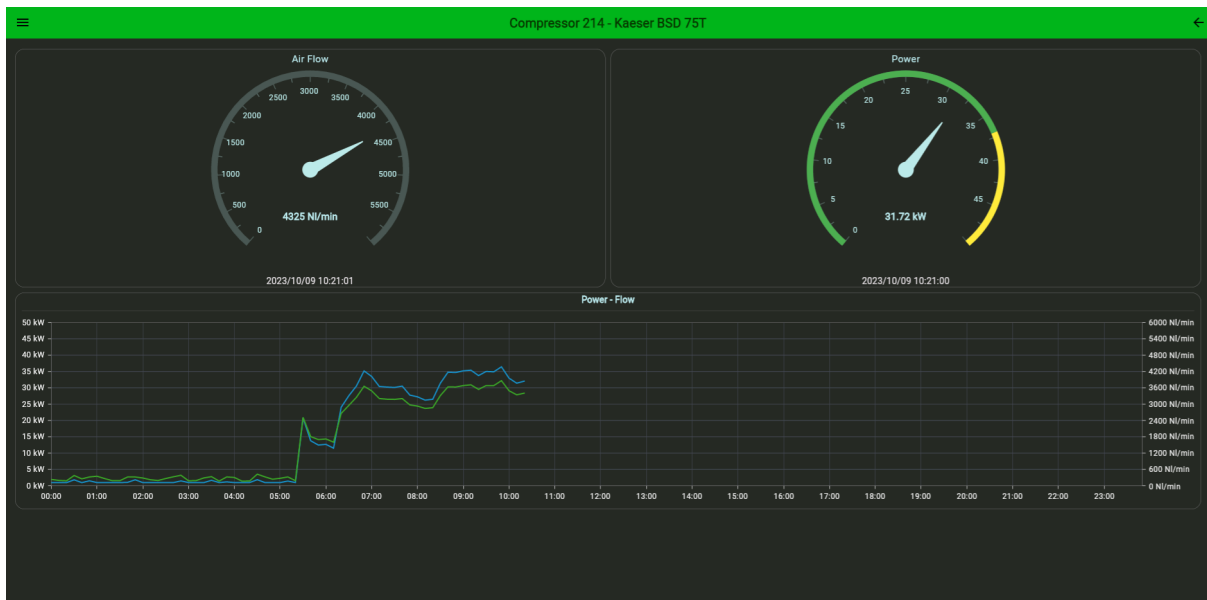
- una struttura dinamica di dashboard relative all'impianto scelto, ad esempio:
 - dashboard principale dell'impianto "Perardi IT"



- da questa si può accedere all' dashboard della sala compressori



- e successivamente all dashboard di un compressore specifico

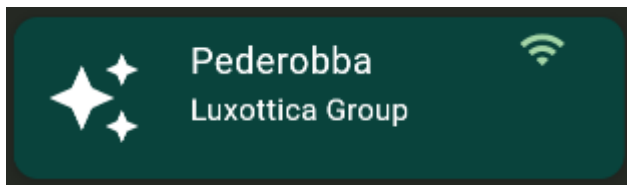


Avendo scelto come framework per il front-end Flutter ogni schermata è composta da “widgets”. I widget sono i componenti base utilizzati in questo framework per costruire qualsiasi interfaccia grafica.

In questa applicazione sono stati implementati numerosi widget personalizzati, ognuno composto di almeno tre parti:

- **Grafica:** il widget vero e proprio; è una classe personalizzata che estende la classe Widget di Flutter. Qui viene implementato il sottoalbero di widget che va a creare la parte grafica.
- **Cubit:** questa è la parte che si occupa della gestione di qualunque tipo di dato inerente al widget creato, si compone di due parti:

- **Cubit:** è una classe che estende dalla libreria Cubit, il suo scopo è quello di gestire i dati inerenti al widget che si sta creando e di notificare alla parte grafica, ed altre eventuali componenti interessate, i cambiamenti. Questo sistema di notifiche è gestito dalla libreria tramite l'assunzione di uno stato, ovvero il cubit in ogni istante ha uno stato corrente e quando questo viene modificato va a notificare tutte le parti dell'applicazione che si sono messe in ascolto sull'istanza del cubit.
- **Stati:** è un sistema di classi e sottoclassi che rappresentano gli stati che il Cubit può assumere e definiscono la struttura del contenuto di ciascuno stato.
- **Settings:** per ogni widget custom viene creata una classe di settings che si occupa di deserializzare il contenuto del file JSON di configurazione del widget e di creare un'istanza della classe stessa contenente quelle informazioni.



Questo è un esempio di widget creato per questa applicazione, è un widget che in questo caso viene utilizzato per rappresentare un impianto ed interagire con esso porta alla schermata principale dell'impianto in questione.

In modo più generico, il suo scopo è quello di consentire lo spostamento da una pagina ad un'altra, per questo è stato chiamato "link_widget".

La parte grafica è composta da una Card che si può cliccare e che contiene a sinistra un'icona, in centro un titolo ed un sottotitolo, a destra un'altra icona rappresentante uno stato.

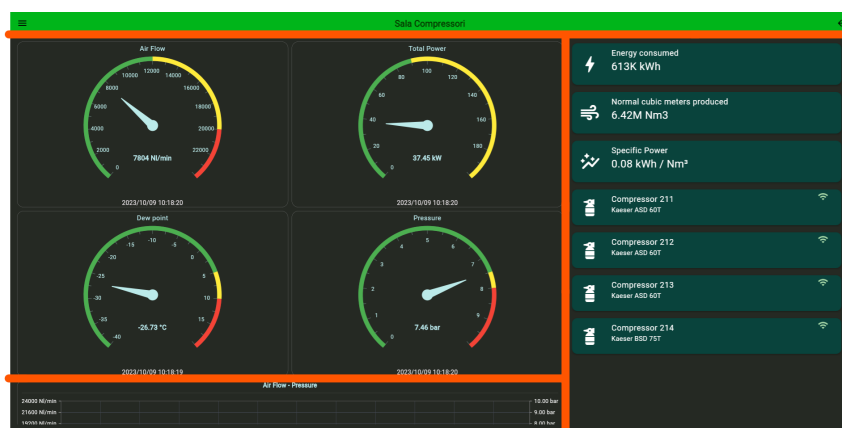
La componente di settings contiene le informazioni su quale icona usare a sinistra, quali sono il titolo ed il sottotitolo e in questo caso a quale topic MQTT sottoscrivere per controllare lo stato dell'impianto. Tiene inoltre traccia di quali colori utilizzare, le dimensioni delle scritte e gli stili applicati. Altre tra le informazioni che memorizza sono la pagina a cui porta ed informazioni riguardanti l'impianto, per esempio il suo ID e nome (il titolo della card non deve per forza coincidere con il nome dell'impianto).

Infine è presente il cubit relativo a questo widget; una delle cose di cui si occupa è la sottoscrizione al topic che contiene le informazioni relative allo stato dell'impianto, e periodicamente controlla se le informazioni dello stato sono aggiornate e quale stato dell'impianto rappresentano. In caso ci siano degli aggiornamenti nei dati abbastanza rilevanti, crea un'istanza dello stato opportuno con le informazioni necessarie (es: il nuovo stato assunto dall'impianto) e la pubblica.

Per decidere quale stato rappresentare, il payload del topic MQTT contiene tra le sue informazioni:

- health rate: è un valore tra 0 e 1 che indica una percentuale di quante delle variabili che dovrebbero venire campionate nella parte sottostante dell'impianto stanno effettivamente venendo campionate.
- time: il timestamp dell'ultimo dato pubblicato dall'edge sottostante.
- ttl: il tempo massimo entro cui l'edge sottostante può pubblicare dei valori senza essere considerato "offline"

In base a come sono impostati questi valori, viene rappresentato uno stato diverso nella scheda dell'impianto.



In questo esempio si può vedere un'altro dei widget più importanti, si chiama "dashboard_widget" e come si può capire dal nome serve a rappresentare una dashboard o pagina.

Anche qui possiamo analizzare le tre componenti principali che ciascun widget deve avere. La parte grafica è composta da quattro sezioni: la barra di navigazione e tre zone in cui sono presenti varie tipologie di altri widget. Queste tre sezioni vengono denominate "top", "right" e "center"; in ciascuna di esse si può specificare quali altri widget ne fanno parte, in questo esempio sono presenti quattro gauge nella zona top, tre MQTT_widget e quattro link_widget nella zona right ed infine si può vedere parte di un Chart_widget nella zona center.

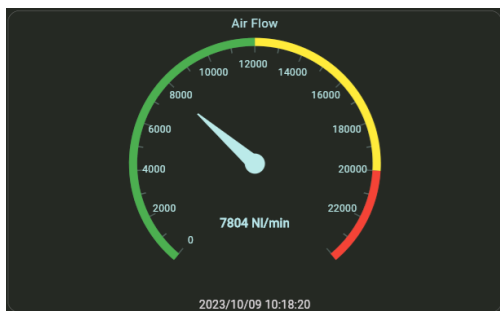
Per la parte di settings sono presenti le indicazioni relative a quali altri widget vanno rappresentati in ciascuna area top/right/center e di che tipologia sono; è specificato anche qual'è il titolo della pagina e con quali impostazioni va visualizzato.

Qua alcune delle cose di cui si deve occupare il cubit sono la creazione di ogni widget nella zona designata, la gestione delle connessioni MQTT per i widget che la richiedono e la schedulazione periodica del refresh dei widget che devono ottenere dati dal database.

Per tutti i widget che contengono una qualunque tipologia di grafico o di indicatore, sia circolare che lineare, si è deciso di utilizzare una libreria pubblica di Flutter sviluppata e rilasciata da Syncfusion. Questa libreria rende disponibili una serie di widget molto versatili, utili per la rappresentazione dei dati. Di seguito vedremo alcuni dei widget che ne fanno uso.

Oltre ai due widget precedentemente descritti, che svolgono un ruolo essenziale nell'applicazione in quanto permettono la creazione delle pagine dell'applicazione e la navigazione tra di esse, ne sono presenti molti altri tra cui:

- **Instant Radial Gauge**

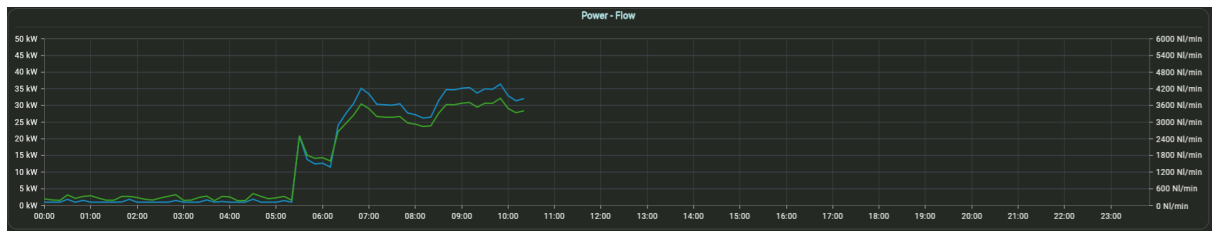


Il suo scopo principale è quello di rappresentare il dato campionato più di recente da un certo sensore; più in generale può essere usato per mostrare una qualunque coppia dato-timestamp. Il widget va a sottoscrivere ad un topic MQTT il cui payload deve contenere almeno il

timestamp ed il valore da assumere; nella maggioranza dei casi questa coppia rappresenta l'ultimo valore campionato da un sensore e viene pubblicata dagli Edge presenti negli impianti. Il widget è però indipendente dall'origine del dato, questo può quindi essere pubblicato da qualunque componente del sistema ed essere anche il risultato di funzioni più complesse.

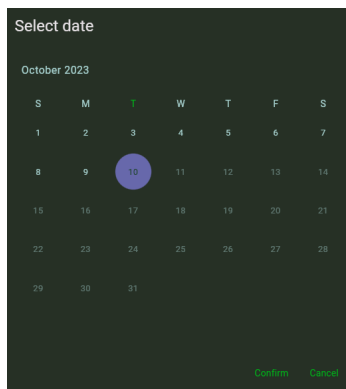
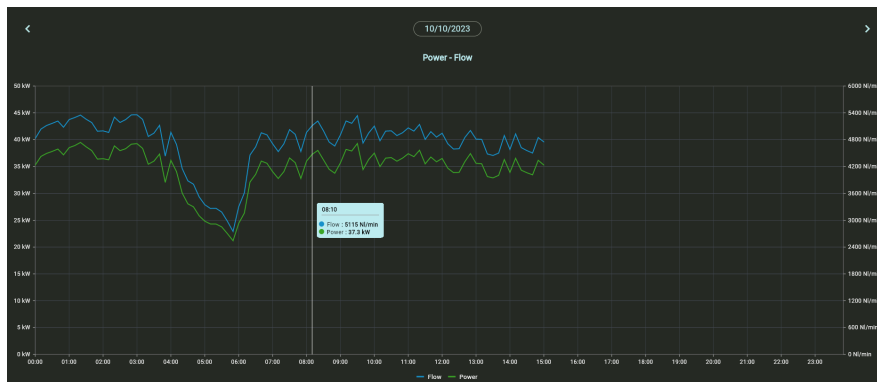
- **Grafica:** l'elemento principale è un widget sviluppato da Syncfusion, ovvero l'indicatore circolare; sono inoltre presenti un titolo ed un timestamp che indicano cosa rappresenta il valore mostrato e quando è stato pubblicato quel valore
- **Settings:** oltre a tutte le impostazioni di dimensione, font e colore per le parti testuali, vengono memorizzati anche i valori di inizio e fine della scala numerica così come i range al suo interno, il colore che ciascun range deve assumere, il suo spessore, l'unità di misura a cui il dato fa riferimento ed anche la precisione con cui il dato deve essere visualizzato. Viene inoltre memorizzato il topic MQTT su cui viene pubblicato il valore da rappresentare.
- **Cubit:** è responsabile della sottoscrizione al topic MQTT e di rimanere in ascolto di nuove pubblicazioni su tale topic. Qualora ricevesse nuove informazioni pubblica un nuovo stato contenente il valore del nuovo dato ed il timestamp relativo, informazioni che sono contenute nel payload del topic.

- **Chart**



Come un comune grafico 2D, ha come scopo quello di rappresentare delle sequenze di coppie di valori che andremo a chiamare tracce.

In questo esempio il grafico è di tipo temporale, ovvero sfrutta l'asse delle ascisse per rappresentare una data ed ora.



È possibile interagire con il grafico per aprirne una versione a schermo intero in cui è possibile usare lo zoom, visualizzare i valori esatti delle tracce e cambiare le impostazioni che indicano il lasso di tempo e la densità dei dati rappresentati.

In questa versione è possibile rappresentare fino a due tipi di tracce, ognuna legata ad uno dei due assi verticali. In una versione ancora non rilasciata è possibile rappresentare un qualunque numero di tipologie di tracce ed è possibile scegliere, tramite un menù apposito, fino a due di esse per mostrare i rispettivi assi delle ordinate.

- **Grafica:** oltre al titolo, l'unica altra componente è un widget rappresentante il grafico in sé che viene fornito dalla libreria di Syncfusion. Nella nuova

versione è anche presente uno slider al di sotto del grafico per permettere di effettuare uno zoom sull'asse delle ascisse con più precisione.

Nella visualizzazione a schermo intero sono anche presenti due frecce per spostare la finestra temporale di interesse avanti e indietro nel calendario, una legenda delle tracce rappresentate e un bottone per aprire un popup che permette la scelta della data in modo più accurato.

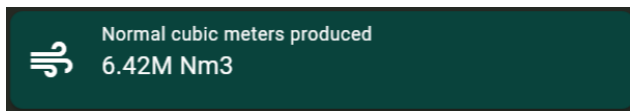
- **Settings:** come prima sono presenti le impostazioni di tutte le componenti testuali ma le impostazioni più importanti riguardano le tracce e la possibilità di cambio del periodo da rappresentare. In questo ambito si imposta quanto è grande il lasso di tempo da prendere in considerazioni e di quanto questa finestra temporale si muove avanti e indietro quando l'utente interagisce con le apposite frecce.

Per quanto riguarda le tracce va definito cosa vanno a rappresentare, ovvero cosa rappresenta il dato, che colore assumono, i limiti della scala visualizzata dell'asse delle ordinate, la precisione della scala e l'unità di misura. Oltre a queste informazioni bisogna anche indicare come ottenere questi dati, vengono quindi specificate varie informazioni sulla query da svolgere, sul metodo e tipologia di aggregazione dei dati (per esempio si può scegliere che ogni dato rappresentato è una media dei precedenti 10 minuti) ed eventualmente come vengono trattati i dati; si può richiedere che una traccia rappresenti il risultato di una funzione di un certo numero di altre tracce. Un ultimo valore molto importante è la frequenza con cui bisogna aggiornare il grafico.

- **Cubit:** in questo caso il cubit svolge un compito più oneroso in quanto deve periodicamente richiedere tramite API che il cloud Robson gli fornisca i dati di ciascuna traccia; successivamente deve elaborarli in modo da combinare insieme tutte le tracce ed emettere uno stato che contenga tutte le informazioni da rappresentare. Alcune tracce possono avere dati mancanti e quindi bisogna interpretare questi "buchi" nel modo adeguato, modificando le informazioni delle tracce in modo che vengano visualizzate correttamente nel grafico.

Deve anche rispondere agli input di cambio data e di movimento della finestra temporale per mantenere le informazioni visualizzate inerenti al periodo scelto. Nella nuova versione in sviluppo deve anche tenere conto delle richieste dell'utente di quali tracce visualizzare e quali invece no.

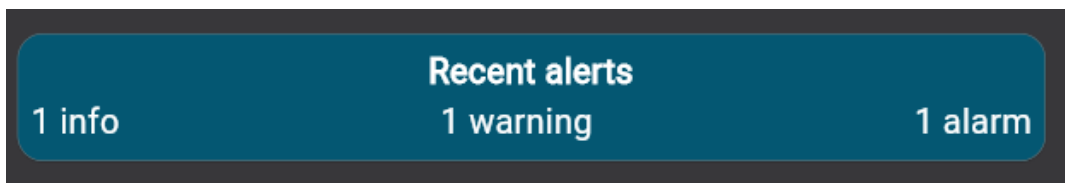
- **MQTT widget**



Questo widget ha lo scopo di rappresentare il valore più recente di una certa serie. Questo valore può provenire direttamente da un sensore o essere il risultato di una qualunque funzione; questo è possibile grazie al fatto che il widget ottiene il valore tramite un topic MQTT ed è quindi indipendente dall'origine del dato.

- **Grafica:** il widget è composto da una card contenente un titolo, un'icona ed un valore con la rispettiva unità di misura.
- **Settings:** nei settings di questo widget, oltre alle solite informazioni relative al titolo, vengono memorizzati il codice dell'icona, la precisione che il valore deve assumere e l'unità di misura a cui il valore fa riferimento. È inoltre necessario tenere traccia di quale topic MQTT contiene il valore aggiornato.
- **Cubit:** il funzionamento del widget è molto semplice: deve semplicemente sottoscrivere al topic MQTT, leggere il valore pubblicato nel payload ed emettere un nuovo stato contenente il valore appena letto arrotondato in modo da soddisfare la precisione richiesta.

- **Alert widget**



Lo scopo di questo widget è quello di mostrare una serie di alert e di permettere a certi utenti di resettarli/cancellarli. Gli alert sono suddivisi in tre tipologie: Info, Warning ed Alarm.

- **Grafica:** il widget è composto di due parti principali, una card di preview che mostra quanti alert sono attualmente presenti ed una seconda parte che consiste in un pop-up in cui sono visibili i dettagli di ciascun alert e dove è possibile resettarli.
- **Settings:** le informazioni principali rappresentate nei settings sono tre topic MQTT sotto cui vengono



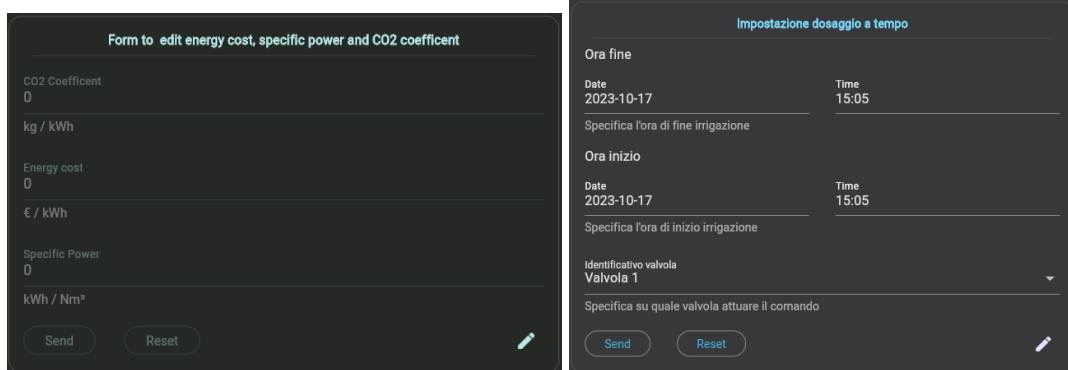
pubblicati gli alert, ciascuno dei tre topic rappresenta una tipologia: Info, Warning, Alarm.

Come per gli altri widget vengono memorizzate anche le informazioni di stile e dimensione delle parti testuali; inoltre si tiene traccia anche del colore che ciascuna tipologia di alert deve assumere.

- **Cubit:** Il cubit svolge un compito abbastanza semplice, deve sottoscrivere ai tre topic ed ogni volta che viene pubblicato un nuovo payload deve fare due cose:
 - se nel payload il campo “is_void” è settato a vero, deve emettere un nuovo stato in cui alla lista di alert attivi aggiunge quello appena ricevuto
 - se il campo “is_valid” del payload è impostato a falso deve invece rimuovere quell’alert dalla lista di alert attivi, se presente, ed emettere un nuovo stato adeguato.

Un’altra funzionalità svolta da questo cubit è quella di reset di un alert attivo, quando l’utente decide di resettare un certo alert, il cubit emette uno stato per notificare la rimozione ed effettua una pubblicazione sul rispettivo topic contenente il payload originale ma con il flag “is_valid” a falso. Vengono inoltre aggiunte delle informazioni supplementari quali il timestamp del reset e l’identificativo dell’utente che ha richiesto il reset.

● Dashboard MQTT



La finalità di questo widget è quella di permettere all’utente di inviare comandi e richieste personalizzate al sistema di back-end.

Per esempio nella prima immagine riportata l’utente di Perardi può impostare un nuovo valore per il costo dell’energia elettrica e per dei coefficienti utilizzati in alcuni calcoli. Nella seconda immagine l’utente di LabGreen può impostare gli orari di inizio e fine irrigazione per varie parti dell’impianto.

- **Grafica:** la parte grafica è composta da un form che comprende un titolo, due bottoni per inviare le richieste ed terzo bottone per attivare e disattivare la

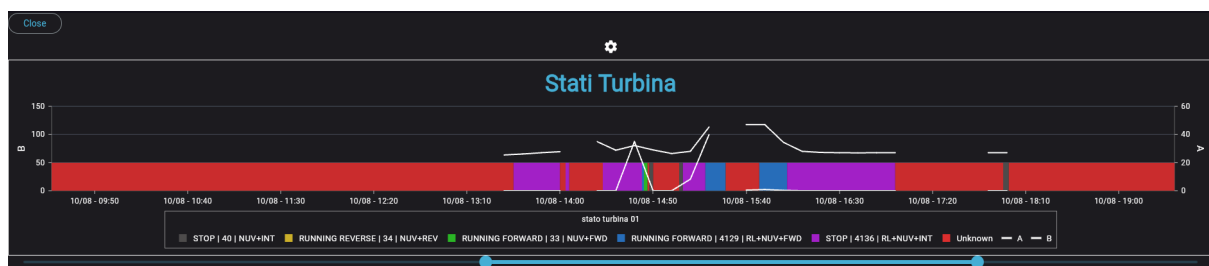
possibilità di interagire con il contenuto del form, questo in modo da non modificare per sbaglio i valori presenti.

La parte principale del form visualizza una serie di righe, ognuna composta da un titolo, un valore ed una descrizione. Possono essere presenti molti tipi di valori: numerici, testuali, vero/falso oppure a scelta da un dropdown.

- **Settings:** oltre alle informazioni del titolo sono presenti i topic MQTT da cui leggere gli ultimi valori pubblicati e su cui inviare i nuovi valori delle richieste. Per ogni riga del corpo del form sono specificati il titolo, la descrizione, il tipo di dato ed il nome che quel dato assume nel payload del topic MQTT; per i tipi di dato a dropdown è inoltre presente l'elenco dei valori tra cui l'utente può scegliere.
Per ogni riga è inoltre possibile specificare se deve essere visibile oppure no, questo per permettere facilmente che il sistema invii dati che non è necessario che sia l'utente a dover inserire manualmente (ad esempio il timestamp di invio della richiesta o l'ID dell'impianto)
- **Cubit:** all'inizializzazione del cubit, questo deve recuperare le ultime informazioni pubblicate dal form tramite l'apposito topic MQTT ed emettere uno stato contenente quelle utilizzate dalla parte grafica. Quando l'utente interagisce con il pulsante a forma di penna deve emettere uno stato per abilitare e disabilitare le interazioni con il widget.
Oltre a ciò, è responsabile di rispondere all'input dell'utente sul bottone di invio e di inviare tutti i dati contenuti nel form e nei suoi campi nascosti sull'apposito topic MQTT.

Sviluppo futuro

Una versione più avanzata del Chart widget è quella rappresentata in questa immagine. Questa versione non è ancora stata rilasciata in quanto è ancora in via di sviluppo.



Questa nuova versione ha come principale funzionalità aggiuntiva quella di utilizzare una traccia come un indicatore di stato. Nella configurazione bisogna scegliere quale traccia

viene utilizzata per rappresentare gli stati, il colore e nome di ciascuno stato ed i range di valori da cui ciascuno stato è rappresentato.

In questa immagine viene utilizzata una traccia contenente un valore di stato letto dell'inverter di una turbina; si è scelto di rappresentare gli stati più comuni che in questo caso sono: spento, acceso in avanti, acceso indietro e sconosciuto.

Sul grafico è possibile visualizzare un qualunque numero di tracce e ciascuna di esse può essere disabilitata per facilitare la lettura del grafico. È anche possibile scegliere due delle tracce presenti in modo che vengano visualizzate le loro scale di riferimento sui due assi verticali presenti a bordo grafico.

Un'ultima aggiunta è la possibilità di applicare uno zoom in modo più preciso tramite un doppio slider presente sotto al grafico.

5 Utilizzo e Applicazioni

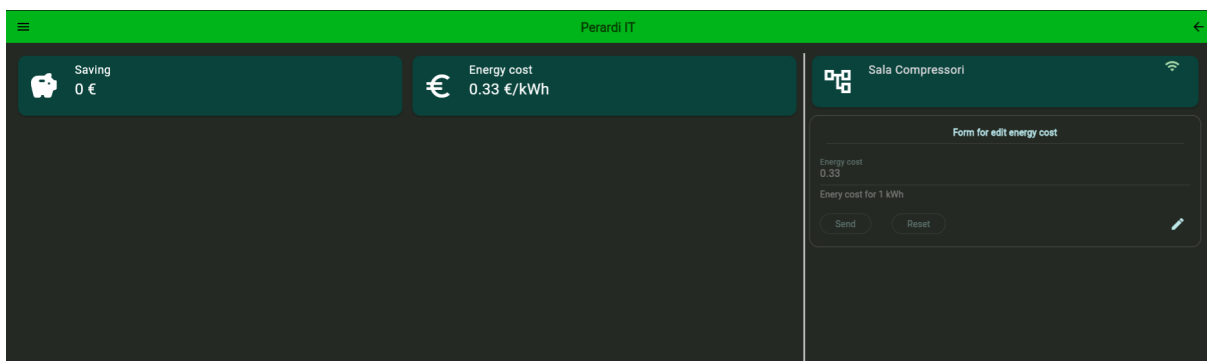
In questa sezione verrà analizzato come è stata utilizzata l'applicazione in tre ambiti diversi.

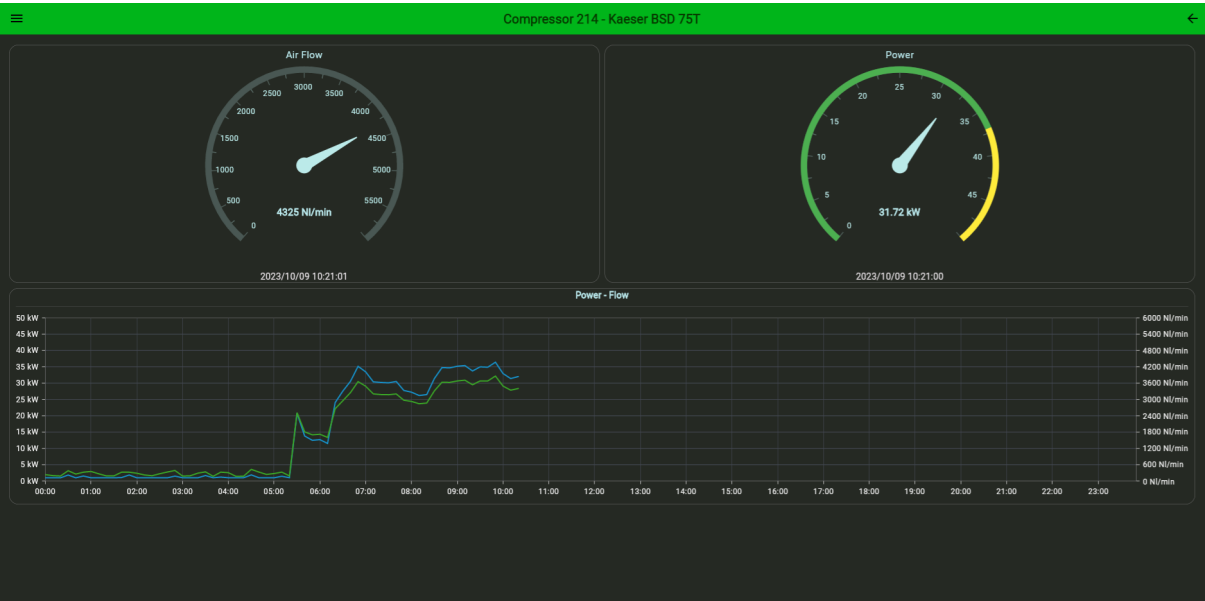
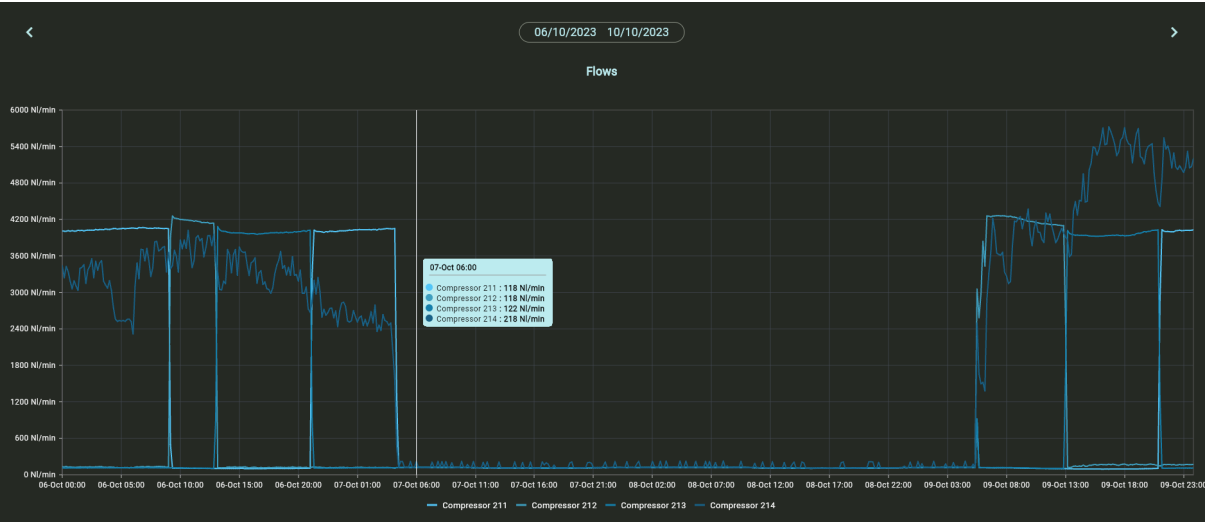
Perardi

Quest'azienda utilizza dell'aria compressa durante dei processi di lavorazione meccanica tramite torni e frese per la realizzazione di componenti per i motori. Al momento stanno utilizzando l'applicazione sviluppata per monitorare il sistema dell'aria compressa.

All'interno dell'impianto è presente una sala compressori responsabile della produzione ed immagazzinamento dell'aria compressa; al suo interno sono presenti quattro nuovi compressori e vengono monitorati tutti i loro consumi energetici e tutti i volumi d'aria compressa che producono, per ogni compressore viene inoltre misurata la pressione in uscita, la temperatura ed umidità dell'aria e lo stato della macchina.

Dei quattro compressori, tre si alternano lavorando a pieno regime mentre il quarto è sempre attivo e modula il suo lavoro in modo da calibrare la pressione dell'impianto.





Tramite l'applicazione l'azienda è in grado di monitorare lo stato del sistema dell'aria compressa e di vedere i risparmi, sia energetici che economici, che si sono verificati dall'installazione dei nuovi compressori.

Il sistema inoltre notifica l'applicazione in caso i compressori lavorino in condizioni non sicure; ad esempio se il voltaggio in ingresso o la potenza assorbita non sono corretti, se la pressione dell'impianto è troppo alta o se la temperatura dei macchinari si sta alzando eccessivamente. Questo permette all'utente di accorgersene in tempo e di intervenire.

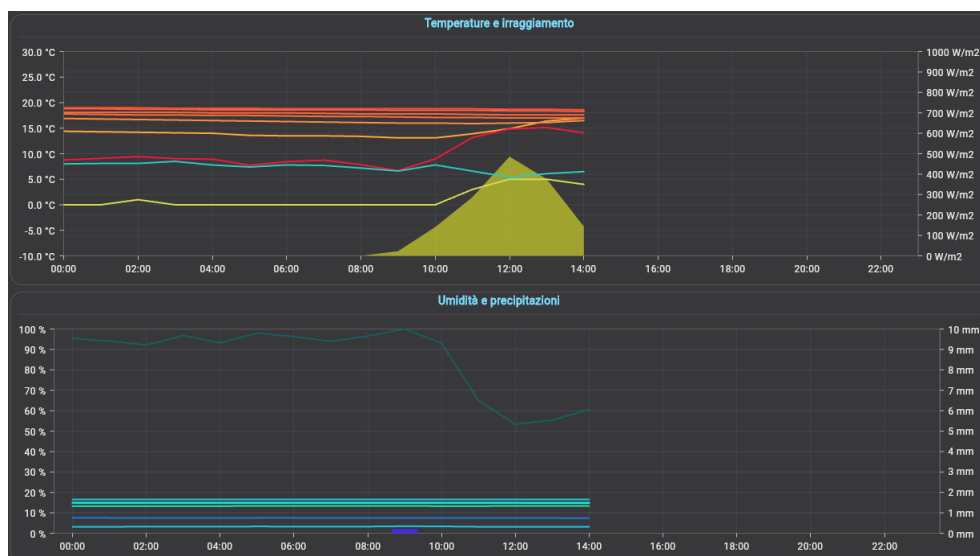
Un'altro utilizzo che verrà fatto di questi allarmi è da parte del sistema di back-end che si potrà occupare di ascoltare il verificarsi di questi eventi e di agire di conseguenza autonomamente.

L'applicazione permette a Perardi anche di verificare in tempo reale la spesa energetica e di accorgersi tempestivamente di un eccessivo costo o di un macchinario inefficiente che consuma troppa energia rispetto a quanto dovrebbe. Questo consente all'azienda di attivarsi per migliorare la fornitura energetica, magari anche spingere la creazione di un'impianto di produzione proprio; o in caso di un macchinario inefficiente, di procedere con della manutenzione o di potenziare l'apparecchiatura.

Lab Green

Questo istituto ha un impianto di irrigazione che è stato automatizzato ed utilizza l'applicazione per monitorare le varie parti del sistema di irrigazione e per attuare l'automatismo del sistema.

Alcuni dei valori monitorati sono i consumi idrici, la temperatura, l'umidità, le precipitazioni, il vento presente ed il grado di evapotraspirazione di ogni sezione del sistema così come quelli generali del sistema nel suo intero.



Lab Green utilizza l'applicazione anche per programmare l'irrigazione di ogni singola zona. È anche capace di attivare e disattivare remotamente l'irrigazione di ciascuna zona tramite l'interfaccia grafica.

The image displays two side-by-side screenshots of a mobile application interface for irrigation control. Both screens have a dark background with light-colored text and input fields.

Left Screenshot: Impostazione dosaggio a tempo

- Ora fine**
Date: 2023-10-17
Time: 15:42
Specifica l'ora di fine irrigazione
- Ora inizio**
Date: 2023-10-17
Time: 15:42
Specifica l'ora di inizio irrigazione
- Identificativo valvola**
Valvola 1
Specifica su quale valvola attuare il comando
- Buttons: Send, Reset

Right Screenshot: Impostazione dosaggio a volume

- Ora inizio**
Date: 2023-10-17
Time: 15:42
Specifica l'ora di inizio irrigazione
- Identificativo valvola**
Valvola 1
Specifica su quale valvola attuare il comando
- Volume da erogare**
0
- Volume in mc da erogare**
- Buttons: Send, Reset

L'utilizzo dell'applicazione ha permesso a questo istituto di migliorare notevolmente l'efficienza dell'impianto di irrigazione.

Ha anche permesso di monitorare con più precisione i consumi così come ha reso possibile visualizzare tutti i valori ambientali in un unico punto in modo da rendere possibile una loro analisi. L'analisi dei dati ambientali ha permesso di calibrare in modo migliore l'irrigazione e di rendere più efficace il sistema, consentendo anche un risparmio idrico ed una riduzione dello stress delle piante.

The image shows a screenshot of the 'Impostazioni comando manuale' (Manual command settings) screen. It features a dark background with light-colored text and toggle switches for four valves.

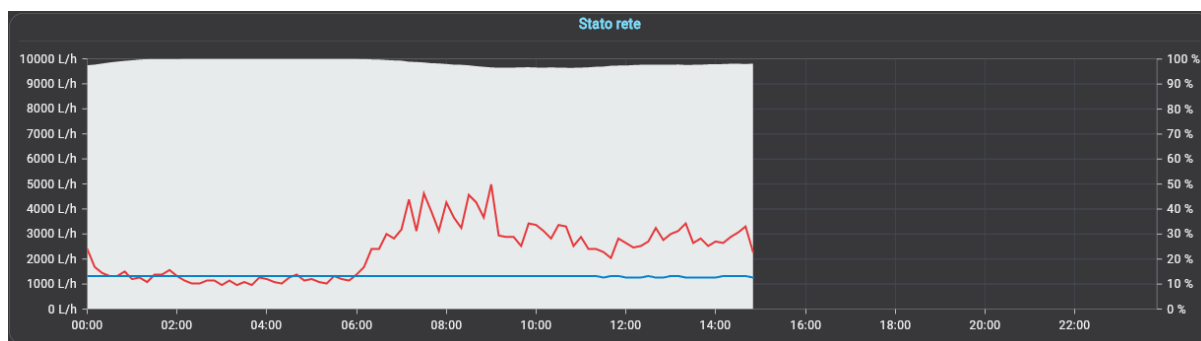
- Valvola 1** (on/off valvola)
- Valvola 2** (on/off valvola)
- Valvola 3** (on/off valvola)
- Valvola 4** (on/off valvola)
- Buttons: Send, Reset

Uno sviluppo futuro dell'applicazione potrebbe permettere a questo impianto di leggere i valori ambientali e di programmare automaticamente l'irrigazione di ciascuna zona calcolando i valori ideali per essa. Questo permetterebbe un completo automatismo ed un'ulteriore risparmio idrico ed un possibile miglioramento nello stato delle piante.

Acquedotto di Bienza

La società dell'acqua potabile di Bienza è responsabile del sistema idrico del paese e porta acqua potabile a più di 350 persone quotidianamente.

Per facilitare le operazioni di manutenzione utilizza l'applicazione per tenere traccia di tutte le entrate ed uscite del bacino idrico.



Già solo con questo grafico contenente il livello del bacino, la somma di tutte le entrate e quella di tutte le uscite, i membri della società sono stati più volte in grado di notare anomalie nel sistema e di intervenire per individuare la fonte del problema.

Il metodo principalmente utilizzato per questo scopo è quello di isolare sezioni dell'acquedotto dal sistema principale e di osservare se il problema persiste. L'utilizzo dell'applicazione ha permesso di individuare molte perdite molto più velocemente di quanto si potesse fare negli anni precedenti; ha permesso anche di individuare trend nei livelli delle sorgenti per iniziare un'analisi a lungo termine della disponibilità idrica della zona in modo da salvaguardare il diritto dei cittadini di avere a disposizione acqua potabile.



Nell'applicazione sono presenti molti altri grafici che permettono l'individuazione di problematiche nell'impianto diverse dalla semplice perdita di una tubatura.

Uno sviluppo futuro di questo impianto sarà la sostituzione delle valvole presenti in tutto l'acquedotto che attualmente sono puramente manuali ma che se fossero digitali ne permetterebbero una chiusura ed apertura remotizzata e tempestiva. Permetterebbero anche l'aggiunta di sensoristica per controllare i livelli di pressione in vari punti dell'acquedotto così come anche dei flussi dell'acqua ed i livelli batteriologici.

6 Tecnologie e Metodologie

Metodologia di Sviluppo

Durante lo sviluppo dell'applicazione si è cercato di adottare una metodologia derivante da Agile chiamate Feature-Driven Development (FDD) che concentra la pianificazione e implementazione sulle funzionalità che vengono richieste.

Alcune delle caratteristiche di come è stata utilizzata questa metodologia includono:

- **Pianificazione Basata su Feature:** le funzionalità dell'applicazione sono state pianificate in base alle esigenze e ai requisiti del cliente, ciascuna trattata come una "feature" distinta.

Ogni qualvolta si individua una nuova feature, per richiesta di un cliente o per necessità interne, il team di sviluppo si raduna per discutere velocemente alcuni punti cruciali:

- cosa serve che la feature svolga
 - quali impatti potrebbe avere sul resto dell'applicazione
 - con quanta priorità bisogna trattare la feature
 - come strutturarne l'implementazione
- **Gestione delle Issue con GitLab:** abbiamo utilizzato GitLab per tenere traccia e gestire issue, problemi e richieste specifiche relative allo sviluppo delle applicazioni. Questo strumento ci ha permesso di organizzare le attività in modo efficace e di assegnare a ciascuna di esse il personale adeguato. Quando viene creata una issue sulla piattaforma gli viene assegnato un nome rappresentativo, un tag (Feature/Bug/etc..), una descrizione ed una persona responsabile per l'implementazione. Qualora l'issue diventasse abbastanza prioritaria allora la persona responsabile si attiva per risolverla.
 - **Unit Test:** Per garantire la qualità e funzionalità del software, sono stati implementati degli unit test specifici per le singole funzionalità. Questi test hanno contribuito a rilevare e correggere gli errori durante lo sviluppo ed ogni volta che venissero apportate importanti modifiche al codice.

Tecnologie Utilizzate

Durante lo sviluppo dell'applicazione abbiamo adottato una serie di tecnologie chiave per garantire l'efficienza, la scalabilità e la sicurezza del sistema. Di seguito sono riportate alcune delle tecnologie principali utilizzate:

Flutter/Dart

Per lo sviluppo del front-end abbiamo utilizzato il framework Flutter che si basa sul linguaggio Dart. Questo ci ha consentito di creare un'interfaccia utente cross-platform reattiva e di alta qualità, accessibile sia tramite browser web che su dispositivi Android.

La scelta di utilizzare questo framework è stata abbastanza obbligata in quanto le attività dell'azienda non si focalizzano sul front-end e si è individuato flutter in quanto permette di scrivere una sola versione del codice e di poterla compilare per tutte le piattaforme di interesse, in questo caso Android e browser web ma in futuro è facilmente espandibile per IOS, Windows e Linux.

La scelta di questo framework è anche dovuta al fatto che è uno dei più famosi e diffusi per lo sviluppo mobile. Questo garantisce un supporto a lungo termine e la presenza di un gran numero di librerie per poter implementare la maggior parte delle funzionalità senza impiegare troppo tempo.

La struttura a widget di flutter è inoltre molto intuitiva e ritengo sia l'ideale per un primo approccio allo sviluppo mobile o crossplatform.

Golang

Sul lato server cloud dove si trovano i servizi di back-end, è stato scelto il linguaggio di programmazione Go (Golang) per gestire quanto concerne le richieste API e tutte le funzionalità dei servizi di back-end. Golang è noto per la sua velocità ed efficienza, rendendolo una scelta ideale per le operazioni di gestione dei dati e la comunicazione con il front-end.

Nonostante utilizzi una sintassi ed una struttura a mio parere un po' poco intuitiva, è un linguaggio noto per le sue alte performance e questo è stato il primo motivo che ha spinto alla sua scelta.

Golang è un linguaggio pensato per essere veloce e per funzionare su macchine multi-core; è in grado di distribuire facilmente il carico di lavoro e la creazione delle corutine è molto veloce e leggera rispetto ad altri linguaggi popolari quali Java, C, C++ e Python.

S.O.L.I.D.

Nello sviluppo dell'applicazione si è cercato il più possibile di attenersi ai principi S.O.L.I.D.:

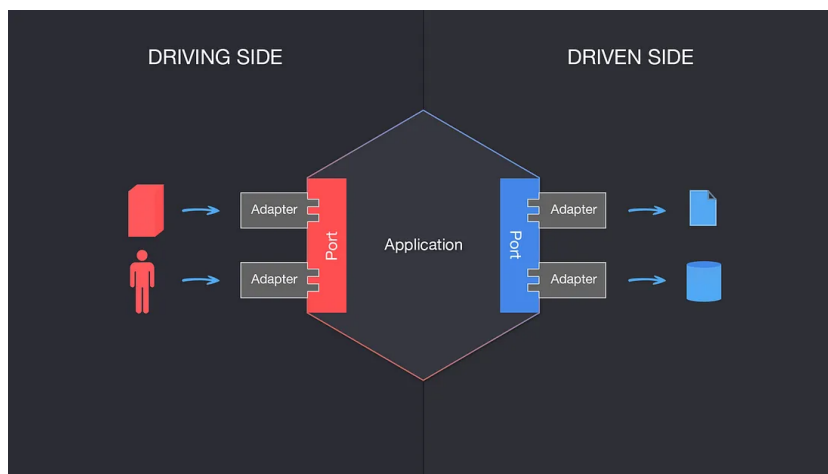
- **Single responsibility:** ogni classe dovrebbe riguardare una sola funzionalità o categoria di esse.

- **Open/closed:** le funzionalità di ciascuna classe dovrebbero essere estendibili senza dover modificare la classe in sé.
- **Liskov substitution:** gli oggetti possono essere sostituiti con qualunque loro sottoclasse senza influenzare il funzionamento del programma.
- **Interface segregation:** le classi non dovrebbero essere obbligate ad implementare interfacce di cui non fanno un completo utilizzo.
- **Dependency inversion:** i riferimenti non dovrebbero riguardare implementazioni specifiche ma definizioni astratte.

Architettura Esagonale

Questa applicazione, sia per quanto riguarda il front-end che il back-end, utilizza un'architettura cosiddetta "esagonale".

Un concetto cardine di questa architettura è che le varie parti di logica ed elaborazione devono essere indipendenti dalle fonti di input/output. Per esempio alla parte di elaborazione dati dell'applicazione non serve conoscere da dove effettivamente provengono i dati e non le serve sapere dove i risultati prodotti andranno a finire.



Questa architettura si basa su tre componenti:

- **Application:** è la componente che in questo caso compie l'effettiva elaborazione dei dati.
- **Port:** in quanto il layer Application è indipendente dalle fonti di input ed output, vengono create delle interfacce di comunicazione che vengono esposte per consentire la trasmissione di informazioni indipendentemente dall'effettiva sorgente o destinazione di questi dati.
- **Adapter:** questo è l'unico punto che conosce l'effettiva origine delle informazioni, per esempio l'utente o un file, il suo compito è quello di interfacciarsi con le porte

dell'architettura e di adattare i dati tra la porta ed il richiedente/destinatario di tali dati e di effettivamente comunicare con il richiedente/destinatario.

Database Non Relazionale

Per la storicizzazione dei dati abbiamo utilizzato un database non relazionale, in particolare MongoDB. Questa scelta è stata motivata dalla flessibilità di un database non relazionale nell'archiviazione di dati semi-strutturati e non uniformi.

In particolare non si presenta la necessità di utilizzare il concetto di entità e relazioni in quanto i dati storicizzati non si relazionano con nient'altro. Con un DB non relazionale le configurazioni sono molto più flessibili non dovendo modificare la struttura del DB per permettere una modifica alle strutture delle configurazioni.

MQTT

Il protocollo MQTT è stato utilizzato per la comunicazione di alcuni tipi di informazioni tra i dispositivi di raccolta dati, il cloud ed il front-end. Questo protocollo leggero ed efficiente è ideale per la trasmissione di informazioni in tempo reale o per mantenere un dato sempre aggiornato.

Il servizio è hostato su un cloud in modo da garantire una maggiore disponibilità.

Docker e Kubernetes

Per la distribuzione dell'applicazione, abbiamo creato immagini Docker che vengono gestite da un cluster Kubernetes. Questa infrastruttura containerizzata permette di gestire le risorse in modo dinamico e di garantire l'alta disponibilità dell'applicazione.

Docker è uno dei più famosi software per l'esecuzione di processi in ambienti isolati e distribuibili. Di questo software è stata utilizzata solo una parte, quella della creazione delle cosiddette "immagini". Queste immagini contengono le informazioni del codice da eseguire e dell'ambiente in cui deve essere eseguito.

Invece di utilizzare i container di Docker, che sono la parte che "interpreta" le immagini, che istanzia l'ambiente e fa girare effettivamente il codice in un ambiente isolato, si è deciso di usare Kubernetes.

Kubernetes è un sistema che si occupa di gestire numerosi container contemporaneamente, è in grado di distribuire le risorse della macchina in modo dinamico tra i container presenti e di seguire delle regole di priorità in questa distribuzione. È in grado di bilanciare le connessioni e risorse tra più container dello stesso tipo, relativi alla stessa immagine, e di

creare nuove copie o eliminarne alcune in caso il lavoro di quelle presenti sia eccessivo o insufficiente.

GitLab

Per tutta l'applicazione è stato utilizzato GitLab come sistema di controllo delle versioni consentendo al team di collaborare efficacemente, tenere traccia delle modifiche al codice e gestire le issue correlate.

Questo sistema era già altamente utilizzato dall'azienda e perciò è stato una scelta abbastanza obbligata. Come già riportato questo è un tool molto utile principalmente per tenere traccia delle modifiche al codice dell'applicazione; permette una storicizzazione elaborata che segue una timeline che può essere diramata in diversi "branch" per effettuare lavori simultanei e poi ogni branch può essere ricongiunto per avere un'unica versione del codice contenente tutte le modifiche.

Questo tool permette anche di utilizzare la metodologia di sviluppo feature-driven in modo molto semplice ed intuitivo e di implementare delle pipeline per automatizzare parti di lavoro ridondante e schematico.

Esempio di Tecnologia e Metodologia in Azione

Per illustrare l'uso di tecnologie e metodologie, consideriamo un esempio concreto:

Scenario: Un'azienda cliente ha richiesto l'implementazione di una nuova funzionalità che consenta agli utenti di personalizzare le notifiche in base alle soglie di temperatura nelle linee di produzione.

Tecnologia Utilizzata: Abbiamo implementato questa funzionalità utilizzando Golang per i servizi di back-end e Flutter per il front-end. Sono stati sviluppati unit test specifici per questa feature per garantire che le notifiche fossero configurabili correttamente.

Metodologia Feature and Issue Driven: La pianificazione e lo sviluppo di questa funzionalità sono stati basati su una specifica feature e sui problemi legati alle notifiche. Le issue sono state aperte, assegnate ed in generale gestite tramite GitLab, questo ci ha permesso di organizzare il lavoro e assicurare la tracciabilità delle modifiche.

Questo esempio illustra come la metodologia "Feature and Issue Driven" abbia permesso di rispondere alle richieste del cliente in modo mirato, concentrandosi su funzionalità specifiche e garantendo la qualità del software tramite unit test dedicati. In questo contesto, la scelta delle tecnologie e l'approccio di sviluppo si sono dimostrati adeguati per soddisfare le esigenze del progetto.

7 Conclusioni

Descrizione

Questa fase comprende una valutazione complessiva del progetto, dei risultati ottenuti e delle lezioni apprese. Le conclusioni riflettono il successo del progetto e identificano le prossime tappe, tra cui eventuali sviluppi futuri e miglioramenti.

Risultati del Progetto

- Durante il progetto, è stata sviluppata un'applicazione altamente personalizzabile che ha permesso a diverse aziende clienti di monitorare e gestire le proprie attività di produzione ed i consumi in modo efficace.
- L'applicazione ha dimostrato di essere scalabile e flessibile, adattandosi alle esigenze specifiche di ciascun cliente.
- La struttura a configurazioni dell'applicazione ha dimostrato di essere efficiente e ha consentito modifiche alle configurazioni in tempo reale senza richiedere il rilascio di nuove versioni dell'applicazione.

Lezioni Apprese

- L'importanza della personalizzazione; la capacità di personalizzare l'applicazione per ciascuna azienda cliente è stata fondamentale per il successo del progetto. Ha dimostrato che una struttura flessibile e generica è cruciale per soddisfare le nuove esigenze in modo più semplice e veloce .
- Il monitoraggio costante e la manutenzione proattiva sono essenziali per garantire prestazioni affidabili e una risposta rapida ai problemi.
- La collaborazione tra le varie parti del team di sviluppo, amministratori di sistema e utenti finali è stata cruciale per il successo del progetto. La comunicazione aperta, la condivisione delle informazioni e la pianificazione delle soluzioni tramite discussioni di gruppo sono state chiavi per un facile sviluppo e per l'adattamento dell'applicazione alle esigenze degli utenti e dei sistemi già esistenti.

Prossime Tappe

- Il progetto è ancora in via di sviluppo. L'applicazione sarà soggetta a sviluppi futuri per migliorare ulteriormente le funzionalità, la sicurezza e le prestazioni.

- Il feedback degli utenti continuerà a essere fondamentale per guidare gli aggiornamenti e le migliorie dell'applicazione. Le esigenze cambiano nel tempo e l'applicazione dovrà rimanere allineata con esse.
- Con l'espansione delle aziende clienti e l'aggiunta di nuovi utenti, sarà importante garantire che l'applicazione rimanga scalabile, sia per la parte di front-end che per quella di back-end. È anche importante che il sistema di hosting sia in grado di gestire un aumento del carico di lavoro.
- Con qualche miglioria di sicurezza e di permessi si potrebbe rendere accessibile alle aziende clienti l'applicazione che permette l'editing delle configurazioni in modo da rendere più semplice l'iter che parte dalla richiesta di modifica e si conclude con l'attuazione della modifica desiderata.

In conclusione, il progetto ha dimostrato l'efficacia dell'approccio di sviluppo basato sulle features, evidenziando l'importanza della personalizzazione e dell'adattamento continuo nonché dell'importanza dell'utilizzo di un'architettura software funzionale e del rispetto dei principi S.O.L.I.D.

8 Bibliografia

Per lo sviluppo di questo progetto è stato indispensabile solo una fonte di informazioni, il sito “pub.dev” dove sono presenti tutte le librerie pubbliche di Flutter ed i link alla loro repository GitHub contenenti la documentazione.

Un secondo strumento utilizzato come supporto è stata l'applicazione di Syncfusion per desktop che permette di vedere un'anteprima di molti dei widget grafici forniti dalla loro libreria in modo semplice e molto rapido.