UNIVERSITÀ DEGLI STUDI DI UDINE

DIPARTIMENTO DI SCIENZE MATEMATICHE, INFORMATICHE E FISICHE

Corso di Laurea in IoT, Big Data e Web

Contributo dei Sistemi MES alla Trasformazione Digitale dell'Industria Manifatturiera

Relatore:
Dr. Ermes Moras

Correlatore:
Prof. Dario Della Monica

Laureando:
Andrea Martin

Anno Accademico 2022/2023

Indice

Astract **7**

1 Introduzione **10**
1.1 Obiettivo . . . 10
1.2 Struttura della tesi . . . 10
1.3 L'azienda . . . 10

2 Stato dell'arte **13**
2.1 Industria 4.0 . . . 13
2.1.1 Definizione di Industria 4.0 . . . 13
2.2 Pilastri tecnologici dell'Industria 4.0 . . . 15
2.2.1 Big Data e analisi dell'AI . . . 15
2.2.2 Integrazione orizzontale e verticale . . . 15
2.2.3 Cloud computing . . . 16
2.2.4 Realtà aumentata (AR) . . . 16
2.2.5 Industrial Internet of Things (IIoT) . . . 16
2.2.6 Produzione additiva/stampa 3D . . . 17
2.2.7 Robot autonomi . . . 17
2.2.8 Simulazione/digital twins . . . 17
2.2.9 Cybersecurity . . . 17
2.3 Innovazione e Progresso . . . 18
2.3.1 Transizione 4.0 . . . 18
2.3.2 Certificazione 4.0 . . . 19
2.3.3 Nuova Sabatini . . . 19
2.4 ERP e SCADA . . . 19
2.4.1 ERP . . . 19
2.4.2 SCADA . . . 20
2.5 PLC . . . 20
2.6 Protocolli di comunicazione . . . 23
2.6.1 EtherNet . . . 23
2.6.2 Profinet . . . 23
2.6.3 Modbus . . . 24
2.6.4 OPC-UA . . . 24
2.6.5 MQTT . . . 24
2.7 Gateway IoT . . . 25
2.8 Docker . . . 25
2.8.1 Container . . . 26

3 Linguaggi, Framework e Strumenti **28**
3.1 .NET 28
3.2 Blazor WebAssembly 28
3.2.1 WebAssembly 28
3.2.2 Struttura di un progetto Blazor WebAssembly 29
3.2.3 Blazor 29
3.2.4 Esecuzione dell'Applicazione nel Browser 30
3.3 Strumenti 30

4 Manifacturing Execution System **33**
4.1 Principali vantaggi 33
4.2 Funzionalità implementate nella nostra soluzione 34
4.2.1 Menu di accesso alle funzionalità 34
4.2.2 Avanzamento della produzione 35
4.2.3 Pianificazione ordini 36
4.2.4 Segnalazione e gestione anomalie 37
4.2.5 Gestione manutenzioni 38
4.2.6 Gestione documentazioni 39
4.2.7 Comunicazione interna 40
4.2.8 Gestione notifiche 41
4.2.9 Contatti rapidi 42
4.2.10 Manuali 42
4.2.11 Planner 43
4.2.12 Import ed Export dei dati 44
4.2.13 Gestione PLC con onEdge 45
4.2.14 Configurazione della piattaforma 46

5 Connessione con i PLC **48**
5.1 Come funziona onEdge? 48
5.1.1 Inserimento dati nel PLC 48
5.1.2 Controllo stato del macchinario 49
5.1.3 Storicizzazione dei dati 49

6 Architettura del sistema implementato **51**
6.1 MES Platform 4.0 51
6.2 onEdge 51

7 Sviluppo **54**
7.1 Software MES 54
7.1.1 Sincronizzazione tra dispositivi 54
7.1.2 Caricamento documenti su server 55

7.2 Connessione con i PLC 57
7.2.1 Main 57
7.2.2 Configurazione 60
7.2.3 MODBUS TCP 61
7.2.4 Utilizzo di file 63
7.3 Distribuzione con Docker 65
7.4 Interfaccia utente 66

8 Test e Validazione **69**
8.1 L'importanza delle persone 69
8.2 Utilizzo dei Log 69

9 Risultati e Conclusioni **71**

A Grafana **73**
A.1 Distribuzione con Docker 74

B Portainer **77**
B.1 Distribuzione con Docker 78

Abstract

Il progresso tecnologico e l'era digitale hanno totalmente stravolto (in veramente poco tempo) il panorama industriale. Oggi definiamo questo nuovo panorama con il termine di Industria 4.0; esso è legato fortemente al progresso e all'applicazione di nuove tecnologie, quali l'Industrial Internet of Things (specializzazione del IoT in ambito Industriale), l'Intelligenza Artificiale, la robotica avanzata e l'analisi dei dati. Ad oggi, le rivoluzioni industriali del mondo occidentale sono state tre, quella del 1784 con la nascita della macchina a vapore (Figura 1 - "Prima"), a seguire quella del 1870 con il via alla produzione di massa (Figura 1 - "Seconda") e per ultima quella del 1970 con la nascita dell'informatica (Figura 1 - "Terza"). La data d'inizio della quarta rivoluzione non è ancora stata scritta ma siamo certi che è in corso e tutti noi ne stiamo prendendo parte; per il momento non è stato determinato nemmeno l'atto fondante, però possiamo affermare con certezza che l'Industria 4.0 rappresenta il carburante migliore per fare avanzare la trasformazione delle tradizionali fabbriche in ambienti altamente connessi e intelligenti, in grado di adattarsi dinamicamente alle mutevoli esigenze del mercato (Figura 1 - "Quarta"). Questa rivoluzione può incutere timore dato che, come ogni rivoluzione industriale passata, la messa in opera di macchinari sempre più efficenti porterà indirettamente (secondo molti studi) ad un saldo netto negativo di oltre 5 milioni di posti di lavoro; bisogna però considerare il drastico calo degli infortuni sul lavoro, l'aumento della qualità di vita delle persone e tutte le attività che compenseranno parzialmente queste perdite, come l'ambito finanziario, il management, l'informatica e l'ingegneria. Di conseguenza, come possiamo leggere su molti libri come "The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies" di Erik Brynjolfsson e Andrew McAfee oppure "The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future" di Martin Ford, cambieranno le competenze e le abilità ricercate nei lavoratori; tutto ciò richiederà un riadattamento radicale di tutto il sistema a partire dalle scuole. In questo nuovo contesto, le imprese rivestiranno un ruolo fondamentale perchè dovranno cogliere appieno tutte le nuove opportunità che si verranno a creare, sfruttando le nuovissime tecnologie per massimizzare l'efficienza operativa, ridurre i costi e restare competitivi. Per aiutare le imprese ad ottenere tutti i goal sopracitati, in questo percorso di trasformazione, è fondamentale l'implementazione di soluzioni avanzate che permettano la gestione dei processi produttivi, come i sistemi *Manufacturing Execution System* (MES). I software MES sono la soluzione più efficiente alle sfide complesse della produzione, offrendo in un unico applicativo la supervisione e il controllo dei processi. Questi sistemi, solitamente, permettono il monitoraggio in tempo reale delle attività di produzione, consentendo un'analisi dettagliata dei dati operativi sia istantanea che in seconda sede (su una quantità di dati maggiore) mediante l'utilizzo di software secondari e facilitando il processo decisionale. L'utilizzo di una di queste piattaforme non solo migliora l'efficienza complessiva dell'impresa, ma contribuisce anche a ottimizzare la gestione delle proprie risorse, a garantire una qualità scelta del prodotto, ridurre costi

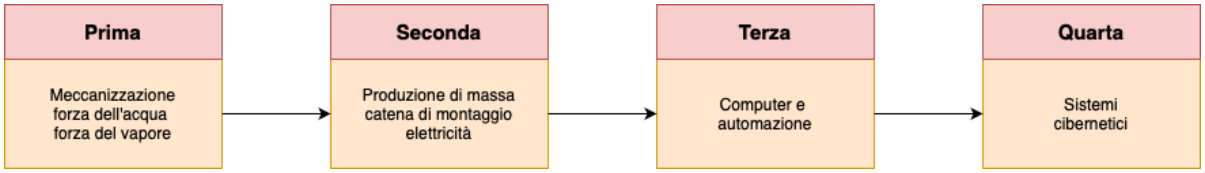

Figura 1: Le quattro rivoluzioni industriali.

e tempi di produzione, aumentare il benessere aziendale e ridurre al minimo gli infortuni sul lavoro. La presente tesi si propone di esplorare aproffonditamente il ruolo cruciale del software MES all'interno di una realtà industriale, ponendo uno sguardo attento sulla realizzazione di esso e di tutti i vari software che lo circondano e gli permettono di interagire con il software gestionale dell'impresa e di scambiare dati con i gateway IoT collegati ai PLC (*Programmable Logic Controller*) dei macchinari 4.0. Prima di proseguire con la lettura di questo documento, l'introduzione vuole lasciare il lettore con una piccola riflessione, presa dal libro "Da zero a uno" scritto dall'imprenditore, investitore e autore statunitense Peter Thiel (noto per il suo successo con PayPal), che verte a far comprendere che l'evoluzione del mondo come lo conosciamo oggi non può proseguire sfruttando la teoria dell'Evoluzione di Darwin (in cui l'evoluzione della specie avviene come risultato di caso e necessità) ma deve essere scatenata da altri principi (come può essere l'innovazione radicale).

1 Introduzione

Negli ultimi anni abbiamo sentito sempre di più parlare (sia in positivo che in negativo) di Industria 4.0, automazione, impiego di software avanzati in campo industriale e dell'importanza che hanno i dati che produciamo ogni secondo della nostra vita.

1.1 Obiettivo

L'obiettivo di questa tesi è portare il lettore a conoscenza della concretizzazione di tutto questo avanzamento tecnologico, illustrando la realizzazione e messa in opera di un software MES (e di tutti i software di contorno) in campo industriale. Nell'esporre questo caso, la tesi si sofferma molto sui concetti relativi al beneficio che può portare l'adozione di un apparato software di questo tipo all'interno di una realtà industriale qualsiasi.

1.2 Struttura della tesi

La tesi è organizzata in una prima parte di introduzione alla materia e illustrazione dei principali concetti. Nella seconda parte viene proposta al lettore un'introduzione ai linguaggi, framework e strumenti adottati per la realizzazione del software (capitoli 2 e 3); sempre nella seconda parte si può apprezzare un rapido riassunto dell'analisi svolta per la scelta dei componenti introdotti in precedenza (capitolo 3) . Solo dopo aver contestualizzato il tutto e fornito al lettore gli strumenti adeguati per comprendere la maggior parte dei concetti esposti in questa tesi, con il capitolo 4 e 5 viene presentato e spiegato minuziosasemente che cos'è un software MES, quali sono gli altri attori che entrano in gioco per l'automazione di un impianto produttivo e come si interfacciano con il software gestionale dell'azienda. Con i capitoli a seguire, il lettore entrerà nella fase di realizzazione (capitolo 6), nella quale si discuterà della macroarchitettura del progetto, della realizzazione del software MES e delle varie interfacce di connessione e scambio dati con i vari dispositivi fisici che compongono l'impianto (capitolo 7). Dopo questa fase prettamente tecnica, sono presenti 2 capitoli di conclusione che trattano rispettivamente Test e Validazione (capitolo 8), le conclusioni e un rapido riassunto di tutto quello che è stato illustrato all'interno di questo documento (capitolo 9).

1.3 L'azienda

L'azienda presso la quale ho potuto seguire il progetto illustrato in questo documento e presso la quale sono attualmente occupato a tempo indeterminato è denominata "Info Team srl", con sede legale e operativa in via D. Guerrazzi, 1 33074 Fontanafredda (PN); ci occupiamo da diversi anni dell'informatizzazione di piccole e medie imprese di tutta Italia, distribuendo software personalizzato sviluppato su richiesta diretta del cliente in base al settore di operatività. Principalmente ci occupiamo della vendita di

un software ERP (*Enterprise Resource Planning*) che viene personalizzato in base alla realtà operativa e aziendale alla quale dovrà servire; oltre alla personalizzazione, ci impegniamo ogni giorno a sviluppare software personalizzato per semplificare la maggior parte delle attività che coinvolgono gli operatori all'interno di ogni contesto aziendale e lavorativo, concentrandoci sulle peculiarità e le necessità di ogni settore e categoria di appartenenza. Attraverso il progetto che ho potuto seguire, abbiamo integrato nei nostri servizi anche una suite software che permette di comunicare con i diversi macchinari/-processi, potendo così ottenere la tracciabilità dell'intera fase produttiva, logistica e di vendita/fatturazione.

2 Stato dell'arte

Questo capitolo del documento si impegna a fornire tutte le nozioni teoriche di base che permettono al lettore la piena comprensione degli argomenti esposti all'interno del documento. Queste informazioni sono state reperite da manuali (online e fisici), forum e siti web dedicati.

2.1 Industria 4.0

Termine utilizzato per la prima volta alla Fiera di Hannover nel 2011 in Germania, in seguito nell'ottobre 2012 viene presentata al Governo Federale Tedesco una serie di raccomandazioni e indicazioni per la sua implementazione.

L'industria 4.0 è da alcuni anni al centro della trasformazione economica in Italia e nel mondo; mira alla risoluzione di alcune problematiche legate alla sicurezza in ambito lavorativo, all'efficienza produttiva e alla qualità di vita (pur portando con sè notevoli svantaggi come il cambio delle competenze e abilità ricercate nelle persone che si avvicinano al mondo del lavoro). Questo termine riconduce direttamente a un processo che sta portando alla produzione industriale del tutto automatizzata e interconnessa; inoltre, le nuove tecnologie digitali avranno un impatto profondo nell'ambito dell'utilizzo dei dati, nell'analytics (per ricavare valore dai dati raccolti), nell'interazione tra uomo e macchina e nel passaggio dal digitale al "reale" (che comprende stampa 3D, robotica, interazioni machine-to-machine e nuove tecnologie per immagazzinare e utilizzare l'energia in modo mirato, razionalizzando i costi e ottimizzando le prestazioni).

Questa "rivoluzione" sta reinventando il modo in cui le aziende progettano, fabbricano e distribuiscono i loro prodotti, cercando di essere sempre più performanti ed avere il minor impatto sull'ambiente. Tecnologie come l'Industrial Internet of Things (IIoT) e l'intelligenza artificiale sono ora profondamente intrecciate nel processo di produzione per ottenere gli obiettivi citati prima e sviluppare collaborazioni simbiotiche e gratificanti tra persone e tecnologia. Quando alla velocità e precisione di un macchinario moderno si aggiunge la creatività, il talento e le capacità che solo le persone possono raggiungere, si ottengono risultati stupefacenti.

2.1.1 Definizione di Industria 4.0

Il termine Industria 4.0 può essere definito come l'introduzione e applicazione delle nuove tecnologie "intelligenti" nel processo produttivo. Comprende un insieme di tecnologie molto vasto, tra cui reti IoT industriali, Intelligenze Artificiali, grandissime quantità di dati (definite dal termine Big Data), robotica e automazione avanzata (Figura 2). L'Industria 4.0, come precedentemente accennato, consente una produzione intelligente e la creazione di fabbriche sempre più consapevoli di ciò che stanno realizzando ed il modo in cui lo stanno facendo (mediante tecniche statistiche). L'obiettivo è migliorare

la produttività, l'efficienza e la flessibilità, consentendo processi decisionali più rapidi e basati su grandi moli di dati reali, i quali consentono un'analisi più accurata rispetto a quella basata su pochi dati o simulazioni.

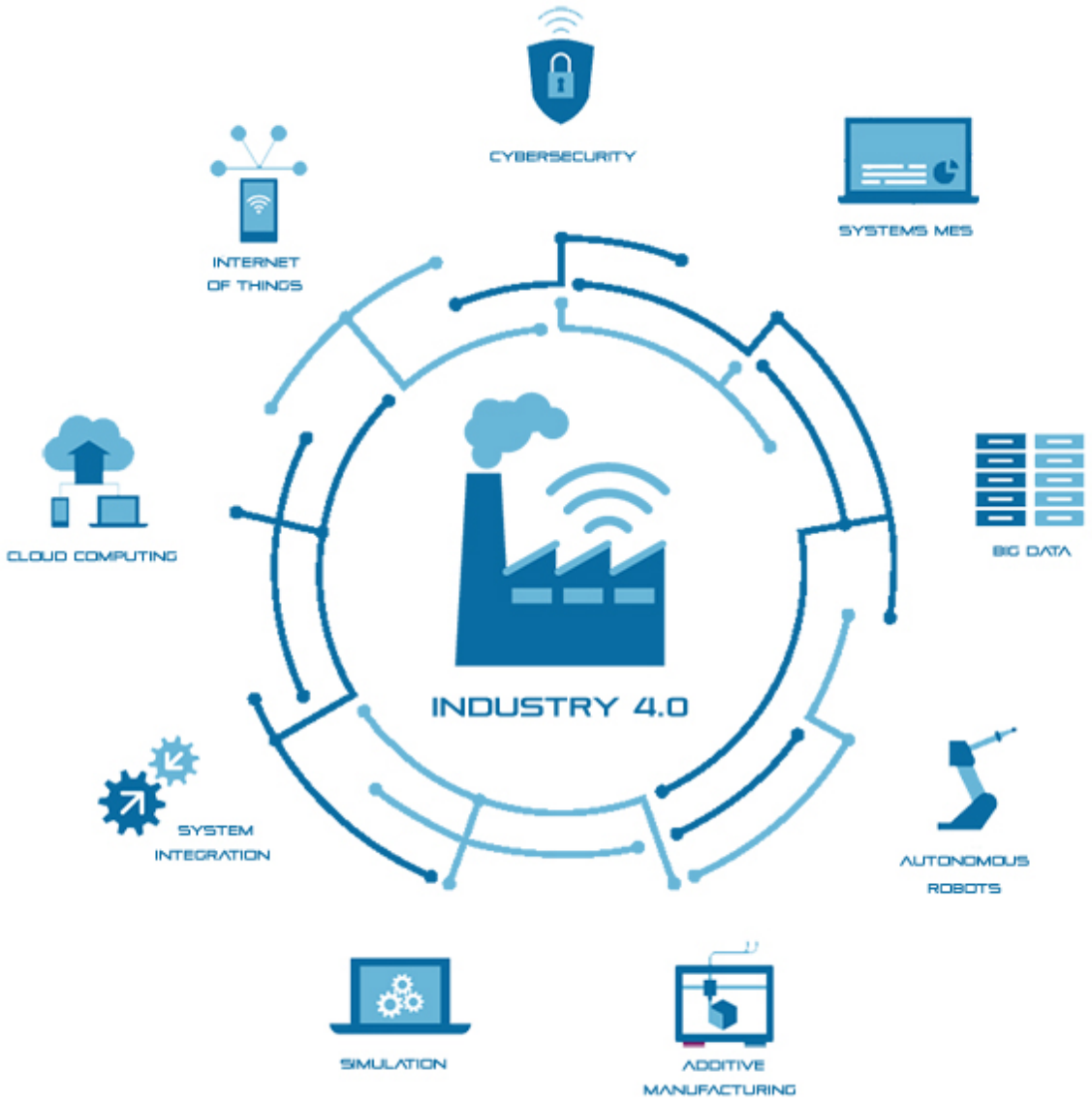

Figura 2: I pilastri tecnologici dell'Industria 4.0

2.2 Pilastri tecnologici dell'Industria 4.0

L'Industria 4.0 si basa su nove pilastri tecnologici; questi creano una connessione stabile tra mondo analogico e mondo digitale rendendo possibile la realizzazione di sistemi intelligenti e autonomi.

2.2.1 Big Data e analisi dell'AI

Con il termine Big Data ci riferiamo a tutti i dati che rispecchiano il concetto delle tre *V*, ovvero dati caratterizzati da una maggiore *Varietà* (caratterizzata da fonti molto eterogenee e mancanza di struttura), che arrivano in *Volumi* sempre più grandi (troppo grandi per essere memorizzati interamente prima di essere elaborati) e con maggiore *Velocità*. Cercando di semplificare il concetto, i big data sono set di dati molto complessi e voluminosi che il software di elaborazione dati tradizionale non è in grado di gestire; questi enormi volumi di dati possono essere utilizzati per gestire problematiche interne all'azienda che, in assenza di queste informazioni aggiuntive, avrebbero avuto soluzioni meno efficienti / soddisfacenti.

Negli ultimi anni, con l'avvento delle Intelligenze Artificiali, per far fronte alla quantità di dati elevata (ingestibile da software tradizionali) si comincia ad adottare questi nuovi software intelligenti per analizzare e sintetizzare le informazioni; inoltre, le intelligenze artificiali vengono solitamente introdotte a supporto dell'analisi dati in tempo reale. In un panorama di Industria 4.0, tutti questi dati vengono raccolti da un'ampia gamma di fonti, che possono includere risorse, apparecchiature e dispositivi abilitati all'IoT. Le fonti di dati si estendono anche al di fuori del mondo digitale (che si viene a creare interconnettendo i macchinari), in altre aree dell'azienda e del mondo. Queste fonti possono includere tutto, dalle recensioni dei clienti, alle tendenze di mercato, alle app meteo e di traffico; generalizzando, possiamo pensare che ogni azione che entra in contatto direttamente o indirettamente con l'azienda produce dati di maggiore o minore importanza.

2.2.2 Integrazione orizzontale e verticale

Nel contesto appena descritto, l'integrazione verticale e orizzontale assumono un ruolo sempre più importante. L'integrazione orizzontale consiste nell'espansione dell'attività di impresa a prodotti, servizi, tecnologie produttive, politiche di mercato, processi e fasi di lavorazione che sono differenti ma complementari al reparto produttivo dell'azienda. In questo modo l'azienda ha la possibilità di ampliare la propria base clienti e ridurre la concorrenza. L'integrazione verticale invece, riguarda l'internalizzazione di tutte le fasi di un processo produttivo necessario per la produzione di un prodotto finito anche attraverso l'acquisizione di aziende con lo scopo di ridurre i costi di produzione e rispondere più rapidamente alle nuove opportunità di mercato; consente inoltre di collegare tutti i livelli logici all'interno della fabbrica, dalla logistica interna fino ai servizi post-vendita.

L'industria 4.0 ha ulteriormente ampliato l'importanza dell'integrazione verticale e orizzontale rendendole fondamentali per la costruzione di smart factory sempre più avanzate. Per soddisfare questi requisiti, l'integrazione orizzontale prevede reti connesse per ottenere elevati livelli di automazione, flessibilità, agilità e capacità lavorativa nei processi di produzione lungo tutta la catena. Semplificando i due concetti, la produzione viene strettamente integrata con processi aziendali come R&D, assicurazione della qualità, vendite e marketing e altri reparti, riducendo le grosse quantità di dati e conoscenze e semplificando le operazioni.

2.2.3 Cloud computing

Con questo termine indichiamo un modello di servizi informatici, tra cui server, storage, reti, software e analisi intelligente dei dati tramite Internet. Questo modello consente alle aziende e agli individui di accedere alle risorse senza doverle gestire o mantenere l'hardware. La tecnologia cloud di oggi costituisce la base per la maggior parte delle tecnologie avanzate, dall'AI all'apprendimento automatico fino all'integrazione con l'IoT. Inoltre, i dati generati dai sistemi implementati nelle aziende che adottano le tecnologie Industria 4.0 risiedono nel cloud e lo utilizzano per comunicare e coordinarsi in tempo reale.

2.2.4 Realtà aumentata (AR)

La realtà aumentata è un sistema informatico che permette di sovrapporre il contenuto digitale a un ambiente reale. Con un sistema di questo tipo, i dipendenti utilizzano occhiali intelligenti o dispositivi mobili (dotati di fotocamera) per visualizzare dati in tempo reale, parti digitalizzate, istruzioni di riparazione o montaggio, contenuti formativi e altre informazioni di interesse, il tutto guardando l'ambiente che li circonda in azienda. Attualmente l'AR trova importanti implicazioni per la manutenzione, il servizio e la garanzia di qualità, così come la formazione e la sicurezza dei tecnici.

2.2.5 Industrial Internet of Things (IIoT)

L'Internet of Things (IoT), in particolare l'Industrial Internet of Things, è così centrale per l'Industria 4.0 che i due termini sono spesso usati in modo intercambiabile. Nell'Industria 4.0, dispositivi, robot, macchinari, attrezzature e prodotti utilizzano sensori per fornire dati in tempo reale relativi alle loro condizioni o prestazioni. Questa tecnologia consente alle aziende di gestire le catene di produzione in maniera più agevole, progettare e modificare rapidamente i prodotti, prevenire l'inattività delle apparecchiature e tenere traccia dei prodotti.

2.2.6 Produzione additiva/stampa 3D

Pur essendo ancora in fase embrionale, la produzione additiva o la stampa 3D viene utilizzata come strumento di prototipazione rapida, offrendo una gamma molto ampia di applicazioni, dalla personalizzazione di massa alla produzione distribuita. Adottando il sistema della stampa 3D, i componenti e i prodotti possono essere archiviati come file di progettazione in inventari virtuali e stampati su richiesta al momento del bisogno, riducendo drasticamente i costi. La tecnologia della stampa 3D cresce molto rapidamente nel tempo, includendo sempre più i filamenti di base come metalli, polimeri ad alte prestazioni, ceramiche e anche biomateriali.

2.2.7 Robot autonomi

Con l'avvento dell'Industria 4.0 e la ricerca dell'automazione sta emergendo una nuova generazione di robot autonomi programmati per svolgere compiti con un minimo intervento umano; questi variano notevolmente per dimensioni e funzionalità, dai droni di scansione dell'inventario (come quelli presentati a fine 2023 dalla grande azienda OpenAI) ai robot mobili autonomi per le operazioni di *pick and place*(propriamente *prelievo e deposito*, utilizzati nei magazzini). Questi robot avanzati sono dotati di software di AI all'avanguardia, sensori e algoritmi di computer vision; questi robot sono in grado di svolgere compit i difficili e delicati e possono riconoscere, analizzare e agire sulle informazioni che ricevono dall'ambiente circostante.

2.2.8 Simulazione/digital twins

Il *digital twin* è la simulazione virtuale di una macchina, un prodotto, un processo o un sistema reale, creata sfruttando i dati dei sensori IoT applicati all'artefatto. Questa tecnologia è fondamentale nell'Industry 4.0 consentendo alle aziende di conoscere meglio, analizzare e migliorare le prestazioni e la manutenzione dei sistemi e prodotti di interesse. Un semplice esempio può essere esposto analizzando la figura dell'operatore di impianto che può utilizzare un digital twin per individuare uno specifico componente malfunzionante, prevedere potenziali problemi e migliorare i tempi di operatività.

2.2.9 Cybersecurity

Innovando gli impianti industriali si naviga verso una maggiore connettività; è proprio in queste circostanze che un'efficace sicurezza informatica è fondamentale. La Cybersecurity si occupa di creare e mantenere un ambiente digitale sicuro atto a proteggere gli utenti e i dati sensibili con l'obiettivo principale di garantire la riservatezza, l'integrità e la disponibilità dei dati, prevenendo i rischi legati a minacce informatiche come virus, malware, attacchi ransomware, phishing e altri tipi di violazioni della sicurezza online.

Questo concetto può essere sviluppato all'interno dell'ambiente aziendale implementando un'architettura Zero Trust e tecnologie come il machine learning e la blockchain; con queste implementazioni, le aziende possono automatizzare la rilevazione, prevenire ed eventualmente rispondere in modo efficiente alle minacce, riducendo al minimo il rischio di violazioni dei dati nelle loro reti.

2.3 Innovazione e Progresso

L'Industria 4.0 oltre a rappresentare un notevole aggiornamento tecnologico abbatte le bariere aziendali collegando team e operazioni in tutta l'infrastruttura di produzione. L'efficienza operativa è aumentata grazie a una migliore allocazione delle risorse, alla riduzione dei tempi di inattività e a una migliore produttività. Questo incremento di efficienza si estende anche alle iniziative di sostenibilità in cui l'analisi e le automazioni intelligenti possono permettere di ottimizzare e snellire ulteriormente l'utilizzo dell'energia, ridurre gli sprechi, progettare e innovare prodotti più sostenibili durante l'intero ciclo di vita del prodotto.

Da citare è anche il lavoro compiuto dal Governo Italiano che, emettendo piani economici a sostegno dell'innovazione, ha reso possibile un rapido sviluppo per tutte le aziende che vi hanno aderito. Inoltre, si stima che grazie all'utilizzo delle nuove tecnologie le imprese italiane potrebbero ottenere un aumento della produttività tra il 30% e il 50%; attualmente possiamo affermare che, negli ultimi anni, le imprese che hanno ottenuto migliori risultati sono proprio quelle che hanno innovato il proprio sistema produttivo seguendo le linee guida dell'Industria 4.0; per questo motivo tutti gli sforzi della politica industriale di oggi protendono verso l'idea di agevolare il più possibile l'innovazione.

2.3.1 Transizione 4.0

Il nuovo Piano Nazionale Transizione 4.0, evoluzione del programma Industria 4.0, intende potenziare la ricerca di base e applicata, favorire il trasferimento tecnologico, promuovere la trasformazione digitale dei processi produttivi e l'investimento in beni immateriali, attraverso crediti di imposta per beni strumentali materiali e immateriali 4.0, ricerca e sviluppo, innovazione tecnologica, design, ideazione estetica e formazione 4.0. Con le disposizioni inserite nella Legge di Bilancio 2020, le agevolazioni per le aziende che investono in innovazione sono cambiate; non si parla più di iperammortamento e superammortamento, ora vengono introdotti i *crediti di imposta* (credito che un'azienda vanta nei confronti dello Stato o di altri enti pubblici), tra cui il Credito di imposta per gli investimenti in beni strumentali, quello per la ricerca e lo sviluppo e quello per la formazione 4.0.

2.3.2 Certificazione 4.0

La Certificazione 4.0 viene definita come dichiarazione attestante la conformità dei beni (sia software che hardware) acquistati secondo i requisiti di Industria 4.0 e presenti negli allegati A e B del Piano Nazionale Transizione 4.0. Può essere richiesta in diverse modalità da tutte le aziende che desiderano beneficiare del credito di imposta 4.0 (citato nel paragrafo precedente). L'attestato di conformità ad Industria 4.0 può essere effettuato da un legale rappresentante (solo se il valore dei beni è inferiore ai 300.000 euro), un perito o un ingegnere iscritto all'Albo (quest'ultimo, tramite perizia giurata, attesta la conformità del bene ai requisiti di industria 4.0) oppure un ente di certificazione abilitato.

2.3.3 Nuova Sabatini

La misura Beni strumentali "Nuova Sabatini" è l'agevolazione messa a disposizione dal Ministero delle Imprese e dei prodotti costruiti in Italia con l'obiettivo di facilitare l'accesso al credito delle imprese. L'agevolazione sostiene gli investimenti per acquistare o acquisire in leasing macchinari, attrezzature, impianti, beni strumentali ad uso produttivo e hardware, nonché software e tecnologie digitali (come una piattaforma MES).

La legge Sabatini rappresenta in assoluto l'agevolazione più longeva dello Stato Italiano ed è rivolta alle imprese che intendono usufruire di un beneficio fiscale per rinnovare il proprio parco macchinari o per acquistare attrezzature funzionali all'attività d'impresa. Inoltre, la normativa specifica la tipologia di beni che possono essere agevolati, definendo le caratteristiche che devono avere tali beni. Più nel dettaglio, i beni (hardware o software) devono essere nuovi, non devono essere in nessun caso terreni o fabbricati e devono essere direttamente riconducibili all'azienda che presenta la domanda a valere sulla nuova Sabatini.

2.4 ERP e SCADA

Altre due componenti fondamentali del sistema informatico di un azienda sono i sistemi ERP e quelli SCADA; i primi si occupano della gestione amministrativa e contabile dell'azienda, mentre i secondi governano i macchinari. Il MES mette in comunicazione queste due realtà, astraendo i dati utili dallo SCADA, come per esempio il numero di pezzi prodotti, il tempo di fermo e operativo dei macchinari ecc. per poi trasmetterli all'ERP affinchè possa elaborarli per ottimizzare la produzione.

2.4.1 ERP

L'*Enterprise Resource Planning* (ERP o software gestionale) è una tipologia di software che le organizzazioni utilizzano per gestire le attività quotidiane di business, come contabilità, project management, gestione del rischio e ciascuna fase della catena produttiva; a questo software possono essere aggregati altri programmi che implementano ulteriori

funzionalità, come l'enterprise performance management, un software che aiuta a pianificare, quantificare, prevedere e comunicare i risultati finanziari di un'organizzazione. Inoltre, grazie alla raccolta di dati transizionali condivisi provenienti da diverse fonti dell'organizzazione, i sistemi ERP eliminano la duplicazione dei dati e ne garantiscono l'integrità tramite un'unica fonte di informazioni. Oggigiorno, per le aziende, l'ERP è indispensabile quanto l'energia che serve a far funzionare tutti i sistemi.

I sistemi ERP si basano su un'unica struttura di dati definita (schema) che condivide, in genere, un database comune, garantendo che le informazioni utilizzate in tutta l'azienda siano normalizzate e basate su definizioni standard. Per riassumere possiamo dire che l' ERP è il veicolo per l'integrazione di persone, processi e tecnologie in un'azienda moderna.

2.4.2 SCADA

Lo *SCADA* (o Supervisory Control And Data Acquisition) non è una tecnologia specifica, ma un tipo di applicazione che riceve i dati derivanti dal funzionamento di un sistema per poterlo controllare e ottimizzare. L'applicazione può essere relativa a un processo di distillazione petrolchimica, a un sistema di filtraggio dell'acqua, al compressore di un condotto o a qualsiasi altro elemento. I sistemi SCADA vengono utilizzati come interfaccia nella maggior parte delle messe in opera, sia verso operatori che verso altri sistemi, nell'ambito dei sistemi di controllo dei processi industriali e sono vincolati dalla presenza di uno o più sensori o attuatori, uno o più microcontrollori (che possono essere PLC o microcomputer), un sistema di telecomunicazione tra i microcontrollori e il supervisore e uno o più computer supervisori che periodicamente raccolgono i dati dai microcontrollori, li elaborano per estrarne informazioni utili, memorizzano su disco i dati o le informazioni riassuntive ed eventualmente fanno scattare un allarme.

2.5 PLC

Un Controllore Logico Programmabile, o *PLC*, è un computer con speciali supporti e protezioni (solitamente collocato a bordo macchina) utilizzato per automatizzare un processo specifico, una funzione della macchina o persino un'intera linea di produzione; essi sono una soluzione di controllo flessibile e robusta, adattabile a quasi tutte le applicazioni. Nella Figura 3 è possibile osservarne un esempio di struttura-tipo. Il PLC riceve dati direttamente dai sensori o dai dispositivi di input collegati al macchinario, elabora i dati e attiva le uscite in base a parametri pre-programmati.

A seconda degli ingressi e delle uscite, un PLC può monitorare e registrare dati relativi al ciclo di produzione, come la produttività della macchina o la sua temperatura in fase di attività, avviare e arrestare automaticamente i processi, generare allarmi in caso di malfunzionamenti, e molto altro. Ci sono alcune caratteristiche chiave che distinguono i PLC da PC industriali, microcontrollori e altre soluzioni di controllo industriale, tra cui

la presenza dei moduli di ingresso e uscita che collegano il PLC al resto della macchina, la presenza di una gamma di porte e protocolli di comunicazione, che gli garantiscono di poter comunicare con una varietà di sistemi, la presenza di un HMI (Interfaccia Uomo-Macchina) per consentire agli utenti di rivedere e inserire informazioni nel PLC in tempo reale.

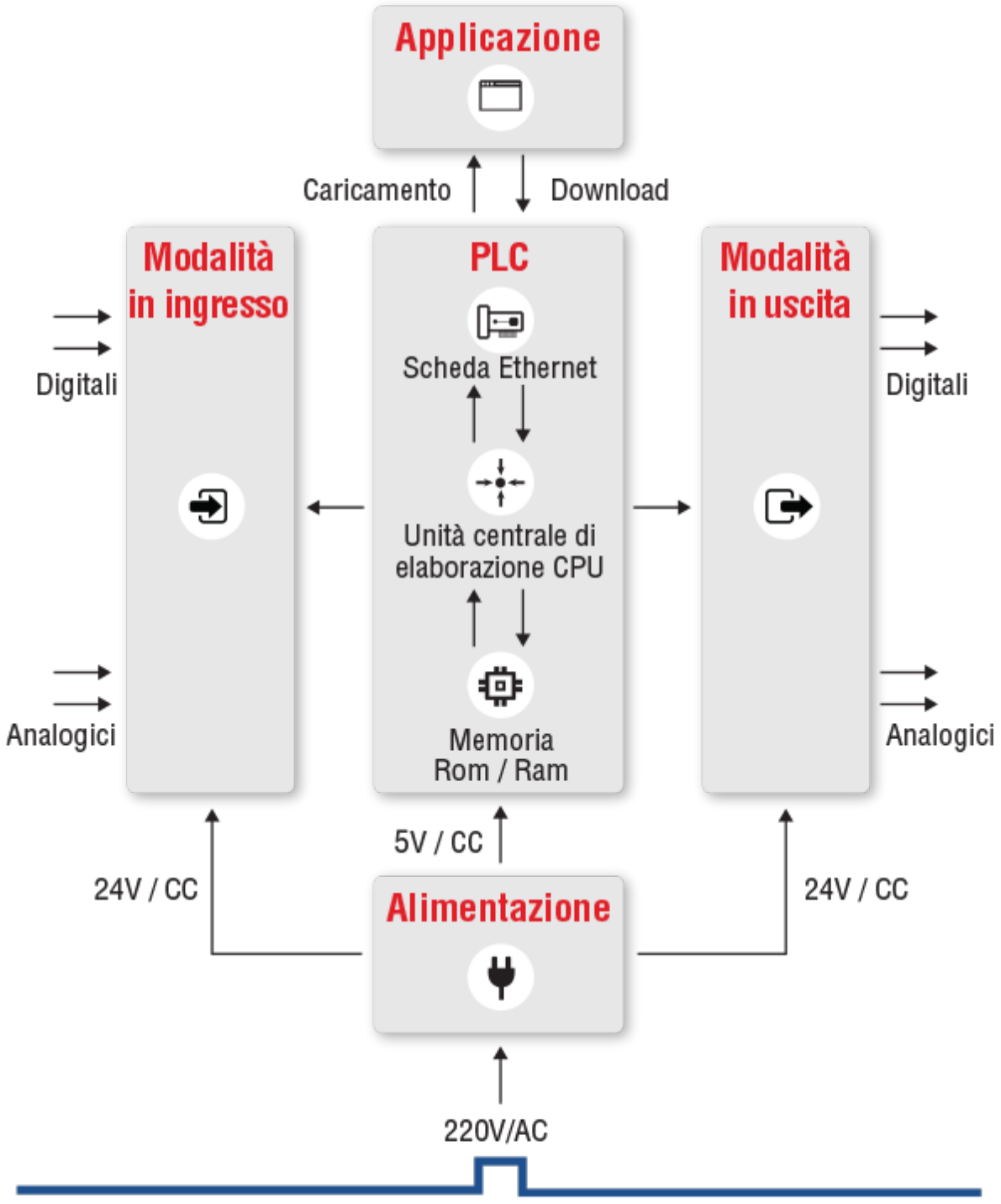

Figura 3: Logica dell'implementazione di un PLC.

2.6 Protocolli di comunicazione

Alla base dell'interconnessione tra un macchinario e un software, c'è necessariamente un protocollo di comunicazione. Quest'ultimo è un linguaggio con il quale una macchina, una scheda elettronica, un software o più in generale un dispositivo comunica con il mondo esterno; più in breve possiamo pensare ad un protocollo come ad una lingua vera e propria. Parlare lo stesso linguaggio (utilizzare lo stesso protocollo) è essenziale per permettere a macchinari, dispositivi e software di comunicare tra loro; quindi, avere un unico protocollo industriale adatto a tutte le esigenze sarebbe ambizioso, ma riuscire ad avere un set limitato di protocolli sarebbe già un ottimo obiettivo. Nella maggior parte delle aziende il parco macchine è estremamente variegato comprendendo macchinari di età, fornitori e tecnologie differenti. Tale eterogeneità richiede spesso l'utilizzo di protocolli di comunicazione diversi rendendo quindi l'operazione di interconnessione complessa e non immediata. La realizzazione del software che scambia informazioni con i macchinari in produzione è di fatto molto articolata dato che quest'ultimo deve fungere da interprete e conoscere tutte le lingue utilizzate dai macchinari interconnessi; in alcuni casi non molto isolati, il macchinario utilizza un protocollo non standard e il software deve poter integrare anche questo "dialetto". Con questo paragrafo proviamo a fare un po' di chiarezza nella piramide dei protocolli di comunicazione industriali utilizzati per l'interconnessione e lo scambio dati.

2.6.1 EtherNet

EtherNet / IP è un protocollo real-time (ben consolidato nel mercato americano) con prestazioni discrete che si adatta perfettamente al campo del monitoraggio; supporta il trasferimento di dati I/O basilari, il monitoraggio del macchinario e il rilevamento di eventi che segnalano un cambio di stato.

2.6.2 Profinet

Profinet è un protocollo industriale real-time, basato su tecnologia Ethernet standard, che definisce una rete che ha l'obiettivo di scambiare dati, allarmi e diagnostica con PLC e altri dispositivi industriali; presenta prestazioni molto elevate rendendolo adatto al controllo istantaneo dello stato della macchina. Profinet IO utilizza tre diversi canali di comunicazione per lo scambio di dati con i PLC, che sono: il canale standard TCP / IP (utilizzato per parametrizzazione, configurazione e operazioni di lettura/scrittura acicliche), il canale real-time (utilizzato per il trasferimento di dati ciclici standard o gli allarmi) e il canale Isochronous Real Time o IRT (canale ad altissima velocità, utilizzato per le applicazioni Motion Control.) La scelta tra il protocollo Ethernet (cfr 2.6.1) o il protocollo Profinet dipende da diversi fattori, tra i quali nominiamo l'area geografica, i sistemi esistenti, i costi e la velocità.

2.6.3 Modbus

Il protocollo Modbus, noto per la sua semplicità e flessibilità, supporta diverse varianti di trasmissione dati, inclusi Modbus RTU (Remote Terminal Unit), che utilizza una trasmissione seriale in formato binario, e Modbus TCP, che utilizza Ethernet per la trasmissione dei dati. Il protocollo è basato su un modello di comunicazione master-slave, in cui un dispositivo (master) controlla e richiede dati da uno o più dispositivi remoti (slave). Nel protocollo i dati sono organizzati in registri (la più piccola entità su cui è possibile effettuare un'operazione di scrittura e lettura) il cui accesso avviene per indirizzo; questi registri sono organizzati in una serie di tabelle ben precise, tra i quali risultano rilevanti Input register (per operazione di sola lettura su registri numerici), Holding register (per operazione di lettura e scrittura su registri numerici), Discrete input (per operazioni di sola lettura su registri booleani) e Coils (per operazione di lettura e scrittura su registri booleani).

Nonostante ci siano protocolli alternativi che presentano prestazioni migliori, il protocollo ModBus è largamente utilizzato perché è open source, ha un basso costo di sviluppo e richiede hardware limitato.

2.6.4 OPC-UA

È un protocollo industriale open-source sviluppato dalla OPC Foundation e multipiattaforma che soppianta l'originale OPC (il quale poteva essere applicato soltanto su dispositivi Windows e poteva essere facilmente oggetto di attacchi informatici), mantenendo tutte le funzionalità del suo predecessore. La forza di questo protocollo è la sua flessibilità, supportando (oltre ai PLC dell'ambito industriale) Linux, iOS, Windows ed anche Android.

2.6.5 MQTT

MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica basato su standard che supporta la messaggistica dispositivi-cloud e cloud-dispositivo; esso è diventato uno standard per la trasmissione dei dati IoT perché offre i seguenti vantaggi: leggerezza, efficienza, scalabilità, affidabilità e sicurezza. I sensori intelligenti, i dispositivi indossabili e altri dispositivi di Internet delle cose (IoT) che utilizzano MQTT per la trasmissione dei dati abbattono i costi di sviluppo, in quanto è facile implementare una comunicazione efficiente.

Questo protocollo implementa il modello "pubblica/sottoscrivi" definendo un *client* (dispositivo che agisce come editore quando *invia* messaggi e agisce come ricevitore quando *riceve* messaggi) e broker (sistema backend che coordina i messaggi tra i diversi client). Inoltre, il broker è responsabile di altri compiti, tra cui autorizzare e autenticare i client MQTT, passare i messaggi ad altri sistemi per ulteriori analisi e gestire i messaggi persi e le sessioni dei client La comunicazione avviene mediante alcuni step ben precisi:

i client iniziano la connessione inviando un messaggio CONNETTI al broker MQTT, questo conferma che è stata stabilita una connessione rispondendo con un messaggio CONNACK (*connection acknowledgement*, fornisce l'esito della avvenuta connessione o meno), il client MQTT e il broker richiedono uno stack TCP/IP (suite di protocolli Internet utilizzata per comunicare facendo in modo che i client non si colleghino soltanto con il broker).

2.7 Gateway IoT

I gateway IoT sono dispositivi hardware che consentono di connettere dispositivi IoT (Internet of Things) alla rete; vengono principalemente utilizzati per la raccolta e l'analisi dei dati generati dai macchinari, nonché per la loro gestione e controllo. L'utilizzo dei questi dispositivi inoltre permette di convertire i dati da diversi protocolli IoT in formati standardizzati, come MQTT, e inviare direttamente questi dati a piattaforme cloud o di analisi.

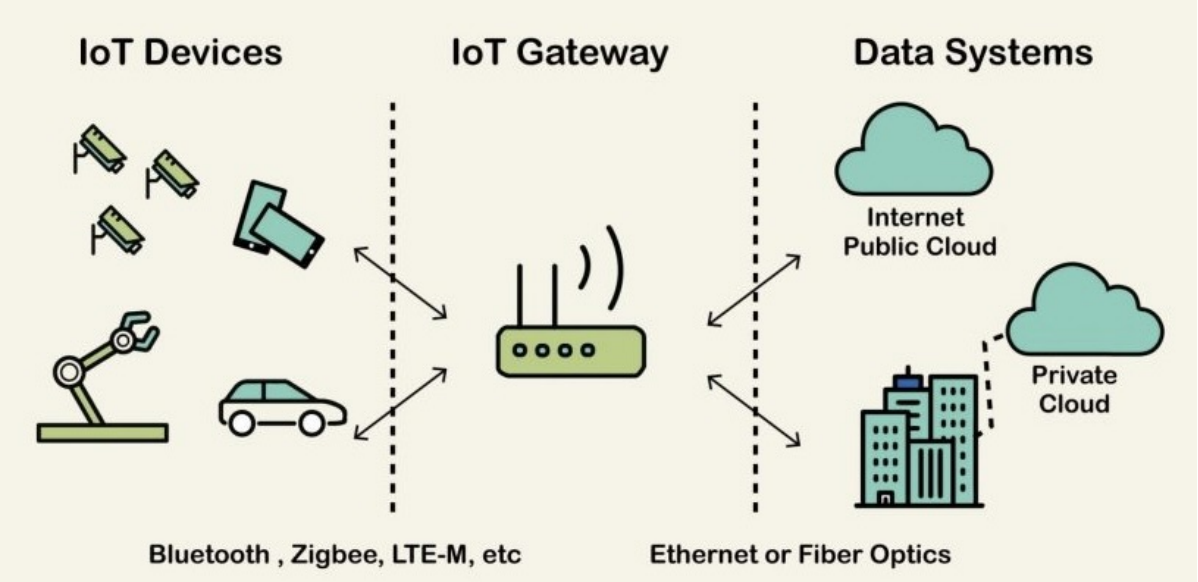

Figura 4: Funzionamento gateway IoT.

2.8 Docker

Docker è un software libero progettato per eseguire processi informatici in ambienti isolabili, minimali e facilmente distribuibili chiamati *container*, con l'obiettivo di semplificare i processi di deployment di applicazioni software; utilizza le funzionalità di isolamento delle risorse del kernel Linux come ad esempio cgroup e namespace per consentire a container indipendenti di coesistere sulla stessa istanza di Linux, evitando tutte le attività

di installazione e di manutenzione che una macchina virtuale comporta. I namespace del kernel Linux per lo più isolano l'albero dei processi, la rete, gli ID utente e i file system montati, mentre i cgroup forniscono l'isolamento delle risorse, inclusa la CPU, la memoria, i dispositivi di I/O a blocchi e la rete. Queste caratteristiche permettono di inglobare un'applicazione e le sue dipendenze in un container virtuale che può essere eseguito su qualsiasi server Linux.

2.8.1 Container

I *container* sono ambienti di elaborazione confezionati e portatili in cui l'esecuzione è completamente isolata dal sistema host; confezionano tutto il necessario per eseguire un'applicazione, come il codice del programma, le librerie di sistema, le dipendenze e le impostazioni di configurazione. Uno dei maggiori vantaggi di Docker è dovuto dal fatto che raggruppando tutti i componenti necessari all'applicazione per funzionare senza problematiche, le applicazioni software possono essere spostate ed eseguite su qualsiasi piattaforma di elaborazione senza incoerenze o errori. Differentemente dalle macchine virtuali, i contenitori non virtualizzano le risorse hardware e questo è dovuto dal fatto che vengono eseguiti sul Docker Engine (la piattaforma di runtime del contenitore) che astrae le risorse. Inoltre, i contenitori sono leggeri e più veloci delle macchine virtuali dato che condividono il kernel del sistema operativo dell'host, eliminando la necessità di un sistema operativo separato per ogni contenitore.

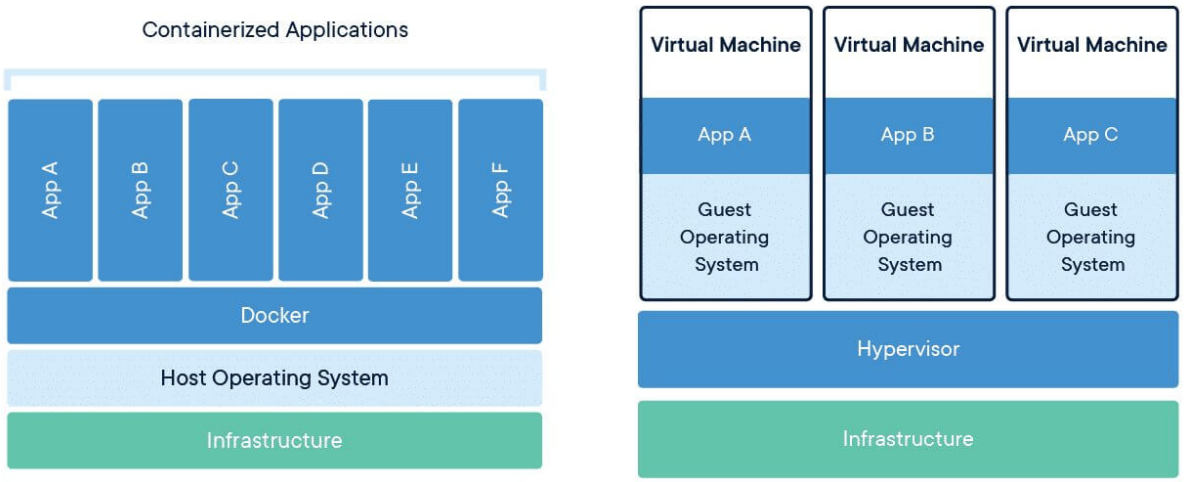

Figura 5: Docker Containers vs Virtual Machines.

3 Linguaggi, Framework e Strumenti

Nel presente capitolo vengono presentati degli strumenti adottati nella realizzazione di questo progetto. Oltre alle loro definizioni è possibile comprendere i motivi per cui sono stati adottati rispetto alle alternative.

3.1 .NET

.NET è un framework open source atto allo sviluppo di applicazioni desktop, Web e mobili con la peculiarità di essere eseguite direttamente su qualsiasi sistema operativo. Questa piattaforma include strumenti, librerie e linguaggi che permettono agli sviluppatori di sviluppare e mantenere software di qualità. I principali linguaggi che funzionano con la piattaforma .NET sono C#, F#, VisualBasic e altri linguaggi personalizzati creati da aziende e sviluppatori. Attualmente le caratteristiche che stanno avvicinando sempre di più gruppi massivi di sviluppatori a questa realtà sono la facilità di sviluppo, le altissime prestazioni e il continuo supporto di una vasta community.

3.2 Blazor WebAssembly

In questa breve sezione verranno illustrati i concetti legati allo standard Blazor WebAssembly che consente di eseguire codice .NET direttamente nel Browser utilizzando WASM.

3.2.1 WebAssembly

WebAssembly (molte volte indicato con l'acronimo WASM) è uno standard aperto che permette di eseguire codice di basso livello direttamente nel browser con l'obiettivo di migliorare drasticamente le prestazioni delle applicazioni web. Fino al 2017 JavaScript ha dominato questo campo ma l'arrivo di WASM ha permesso di eseguire il codice ad una velocità vicina a quella del codice nativo della macchina. WebAssembly, oltre ad essere molto prestante, include molte altre caratteristiche come la portabilità del codice, l'interoperabilità con il codice JavaScript e la particolare sicurezza (dovuta all'esecuzione in un ambiente sandbox sicuro all'interno del browser). Grazie a tutte le caratteristiche appena elencate, WebAssembly è diventata una tecnologia importante per gli sviluppatori, permettendo uno sviluppo rapido di applicazioni sempre più prestanti, sicure e svincolate dall'ambiente di sviluppo ed esecuzione. Inoltre, è importante menzionare che questa tecnologia è sempre più spesso utilizzata in combinazione con linguaggi come C, C++, Rust e C# (come nel caso esposto in questo documento).

3.2.2 Struttura di un progetto Blazor WebAssembly

Nella struttura classica di un'Applicazione Web possiamo immaginare il browser come un'area di lavoro in cui la parte di interfaccia viene sviluppata quasi interamente con HTML (HyperText Markup Language) e CSS (Cascading Style Sheets) e mediante l'utilizzo di JavaScript possiamo interagire con le API messe a disposizione dal Browser (WebSocket, Web Storage e altre); quest'ultimo viene eseguito all'interno della JavaScript Runtime (un ambiente controllato che gestisce l'esecuzione). WebAssembly si innesta direttamente in questo ambiente controllato, guadagnando la possibilità di interoperare con JavaScript (eseguendo un codice legacy oppure utilizzando librerie JavaScript già pronte).

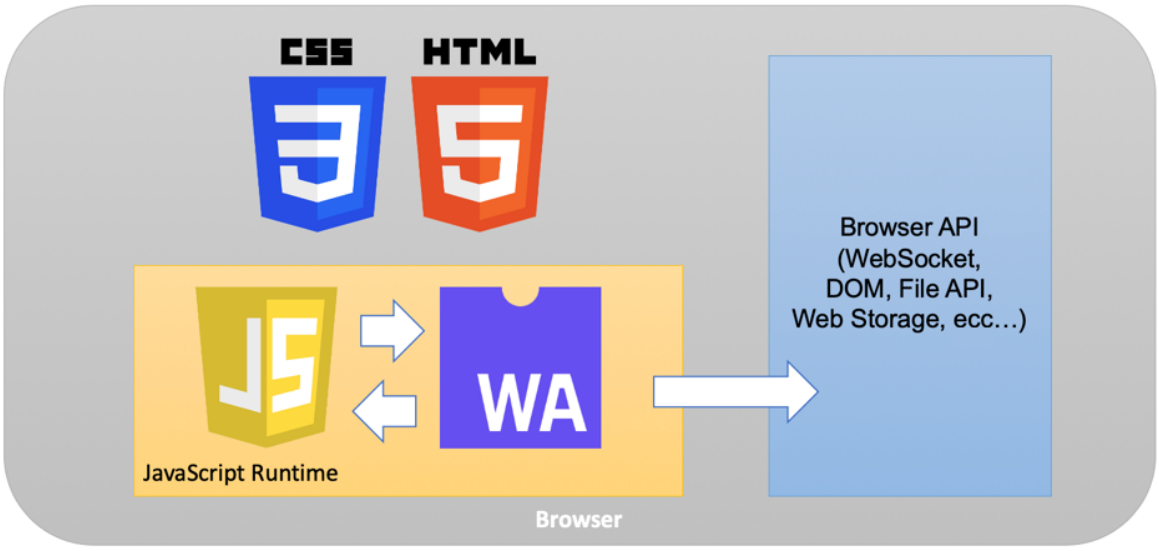

Figura 6: Struttura di un Applicazione Web moderna.

3.2.3 Blazor

Lo standard WASM appena descritto permette di eseguire un codice binario direttamente nel browser ad una velocità simile a quella del codice nativo. Attualmente però non esiste un compilatore che permetta di generare codice WASM a partire da codice C#; per sopperire a questa mancanza viene adottato *Blazor* che permette di eseguire nel browser un codice IL (*Linguaggio Intermedio*). Un aspetto fondamentale di Blazor è che permette allo sviluppatore di realizzare Applicazioni Web sfruttando nozioni pregresse di .NET, senza la necessità di apprendere JavaScript (oppure TypeScript) e di fatto realizzare Server e Client con un unico linguaggio di programmazione.

Figura 7: Esecuzione codice WASM.

3.2.4 Esecuzione dell'Applicazione nel Browser

Altri benefici che si possono ottenere adottando queste tecnologie sono ben visibili in fase di esecuzione; infatti, l'applicazione viene eseguita direttamente nel browser senza la necessità di WebSocket o di un back-end.

Figura 8: Esecuzione nel Browser.

3.3 Strumenti

Per la realizzazioine del progetto trattato da questo documento sono stati utilizzati un numero molto elevato di strumenti informatici; di questi verranno citati solamente i più importanti che hanno segnato un punto cruciale per la chiusura del progetto. Per realizzare l'intero progetto sono stati utilizzati principalmente software Microsoft data l'elevata qualità dei loro prodotti e l'ottima stabilità in campo Enterprise; tra questi possiamo citare l'*IDE* (Integrated Development Environment) di sviluppo Visual Studio 2022 e Microsoft SQL Server Management Studio 18 (utilizzato per la realizzazione del

DB e per i test). Altri strumenti meno comuni sono gli sniffer di rete per individuare in rete i macchinari e alcuni simulatori per emulare il macchinario (e il protocollo adottato) in fase di sviluppo. Per la distribuzione del software sono stati utilizzati ulteriori "programmi" tra cui Docker e IIS (software server Web sviluppato per i sistemi operativi Windows, consente agli utenti di ospitare siti Web e altri contenuti sui computer Windows); questi ultimi sono le opzioni che adottiamo per pubblicare nel server del cliente la piattaforma funzionante.

4 Manifacturing Execution System

Il MES (Manufacturing Execution System), è un sistema software che permette alle aziende di monitorare, tracciare, documentare e controllare il processo di produzione dei beni, dall'inizio con l'arrivo del materiale in magazzino fino al termine della produzione del bene finito. Questo Software si colloca, nell'ambiente aziendale, tra l'ERP (Enterprise Resource Planning) e i sistemi di controllo dei processi; il MES permette di massimizzare l'efficienza mettendo in campo dei dati analitici relativi all'andamento della produzione. Indipendentemente dalle dimensioni di un'azienda manifatturiera, un MES può contribuire ad aumentare produttività e la redditività facendo in modo che le informazioni relative ai processi di produzione siano correttamente indirizzate verso le figure appropriate. Secondo le previsioni esposte in un rapporto di Transparency Market Research , il mercato globale delle piattaforme MES "genererà ricavi pari a 18,06 miliardi di dollari entro la fine del 2025." Questa crescità straordinaria è trainata dalla grande introduzione dell'automazione in ambito industriale e dall'esigenza di conformità normativa.

4.1 Principali vantaggi

Adottando un sistema MES l'azienda può ottenere diversi vantaggi utili a restare competitivi sul mercato. Grazie ai dati relativi al controllo di qualità trasmesse in real-time, l'azienda può arrestare immediatamente la produzione non appena vengono individuati problemi riducendo sprechi, scarti e migliorando del controllo della qualità; incrociando questi dati con risorse di personale, materiali e attrezzature, i sistemi MES generano programmi di produzione validi che alternano produzione e manutenzione in maniera tale da ottenere l'efficienza massima dell'impianto. Grazie alla programmazione della produzione, possono essere definiti con precisione i quantitativi di materiale, semplificando il lavoro dell'ufficio acquisti e di fatto ridurre lo stock a magazzino (semplificando moltissime attività derivate dalla gestione di un magazzino ampio). Facendo transitare tutti i dati di produzione all'interno di una piattaforma software si ha di conseguenza un maggiore tracciamento della vita del prodotto (producendo una mole consistente di dati/informazioni molto utili) e l'eliminazione quasi totale del cartaceo.

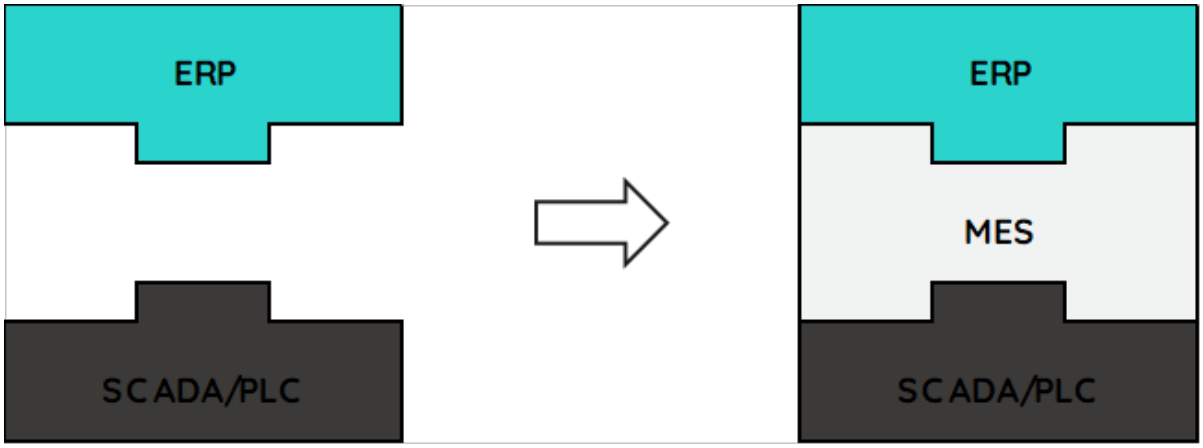

Figura 9: Dove si colloca un MES nell'ambiente aziendale.

4.2 Funzionalità implementate nella nostra soluzione

Il software MES che è stato sviluppato nell'azienda in cui lavoro, oltre ad avere le funzionalità fondamentali atte alla gestione della produzione, integra molte funzionalità utili alla comunicazione intra-aziendale e all'organizzazione. Nelle sottosezioni successive verranno illustrate dettagliatamente tutte le funzionalità disponibili nella piattafroma che abbiamo sviluppato.

4.2.1 Menu di accesso alle funzionalità

Nella piattaforma descritta in questo documento, sono stati progettati e curati nel dettaglio i menù, al fine di mantenere un ordine logico-visivo adeguato e di garantire un accesso intuitivo e rapido alle varie funzionalità. Nella sidebar di sinistra l'utente può trovare tutte le funzionalità legate alla gestione di produzione, attività e documenti. Mentre, nel menu a tendina di destra, sono state collocate tutte le maschere atte alla configurazione della piattafroma o alla risoluzione di eventuali problematiche legate all'utilizzo del software (manualistiche e assistenza da remoto). Un punto fondamentale di questo software è proprio l'elevata possibilità di configurazione che offre al cliente, permettendo all'amministratore o responsabile di limitare le funzionalità accessibli ad ogni account (creando di fatto una gerarchia di utenti che possiedono l'accesso alle sole maschere di interesse).

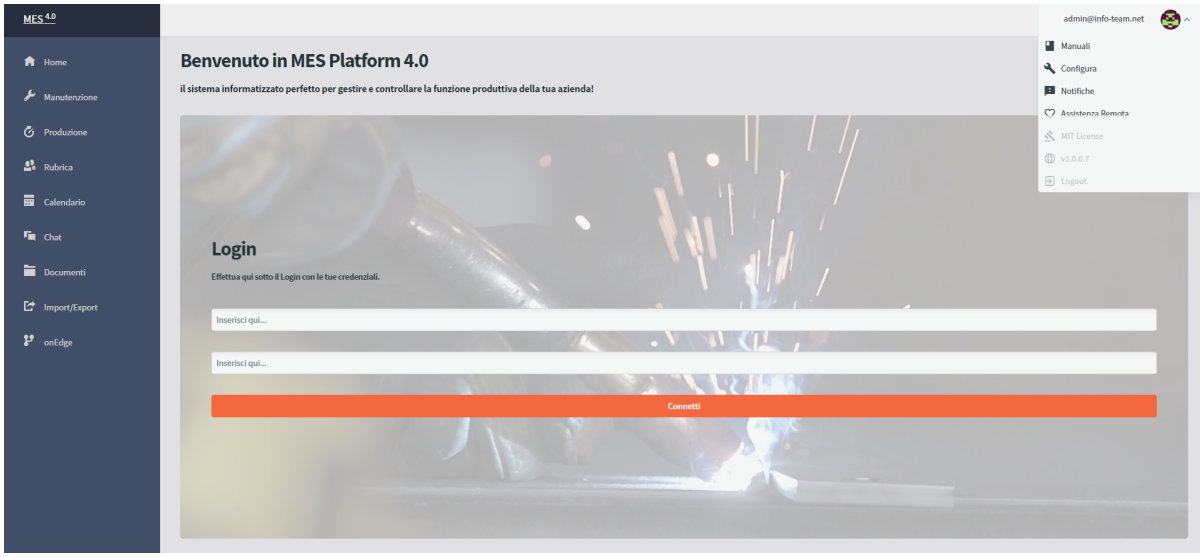

Figura 10: Maschera di Login con visualizzazione dei Menu.

4.2.2 Avanzamento della produzione

La piattaforma permette di avviare la produzione relativa ad un ordine specifico (precedentemente aggiunto alla lista degli ordini disponibili dalla maschera di pianificazione) e terminarla specificando il numero di pezzi validi prodotti e il numero di pezzi da scartare. Questa mascherà viene abilitata per tutti i macchinari che non possiedono un sistema di avanzamento degli ordini già implementato nel Pannello di Controllo a bordo macchina; la funzionalità appena descritta sfrutta il connettore software denominato onEdge (sviluppato dall'azienda per cui lavoro) per scambiare dati, mediante un canale di comunicazione biunivoco, con i PLC.

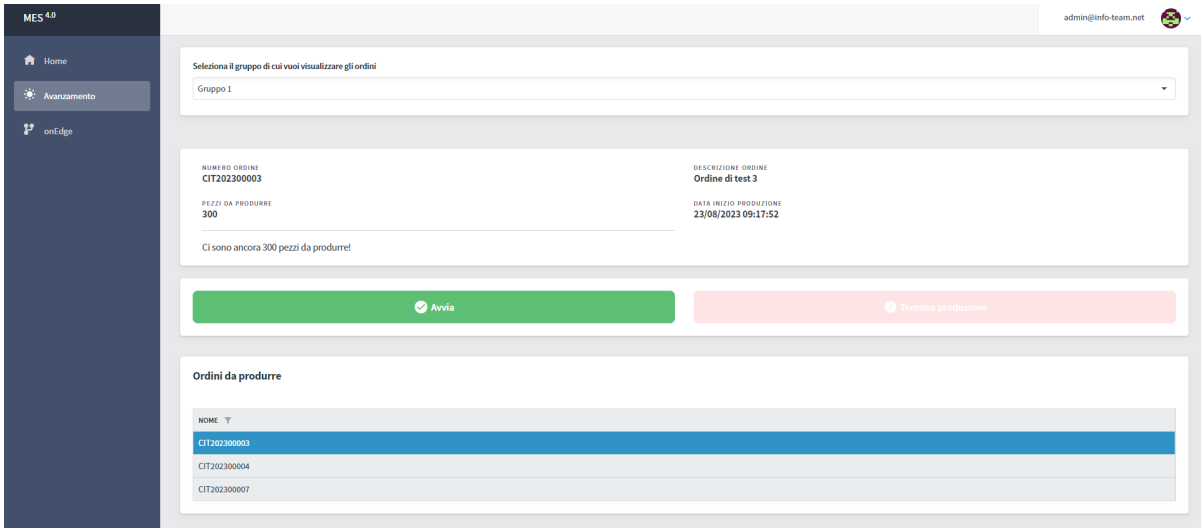

Figura 11: Maschera per gestire l'avanzamento della produzione.

4.2.3 Pianificazione ordini

Nella maschera di pianificazione l'utente ha accesso a tutte le informazioni relative al ciclo produttivo; da qui l'utente può (dopo aver selezionato il gruppo produttivo di interesse) visualizzare e gestire gli ordini da mandare in produzione, prendere visione di eventuali ordini attualmente in fase di produzione e ottenere informazioni relative ad un ordine già prodotto nello storico. Questa mascherà solitamente viene utilizzata per monitorare la produzione oppure fare una rapidissima analisi dell'andamento azziendale (per poi approfondire nelle maschere, presenti nel software ERP, dedicate alle statistiche).

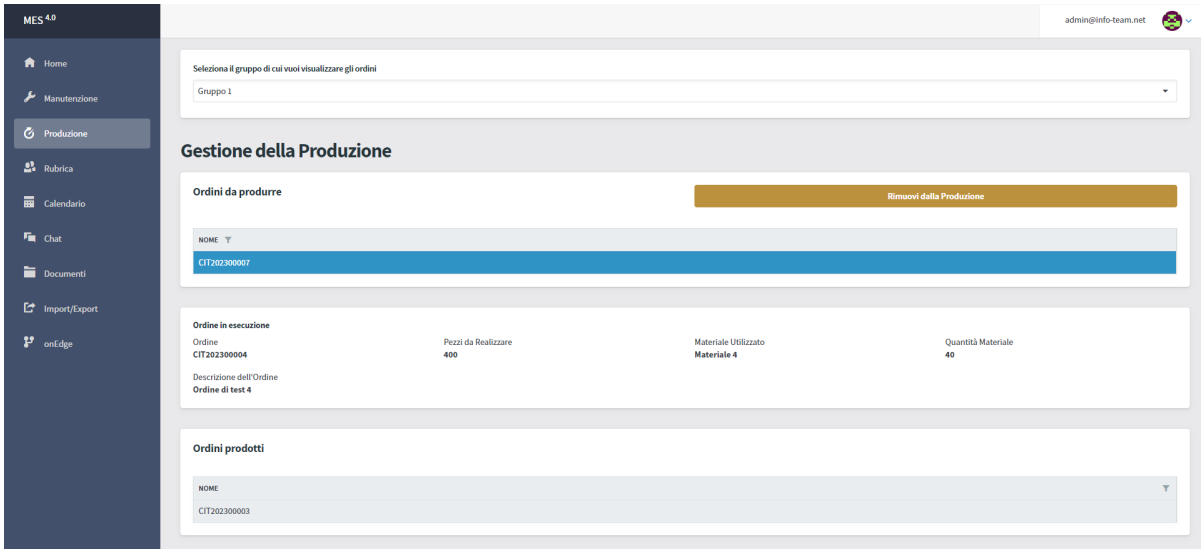

Figura 12: Maschera riassuntiva stato della produzione.

4.2.4 Segnalazione e gestione anomalie

Nella schermata home vengono mostrati gli ordini da confermare e preparare per la produzione e le anomalie rilevate degli impianti. La gestione degli ordini in questa schermata permette di visualizzare gli ordini di cui si deve ancora pianificare la produzione, visualizzare i dettagli di questi ultimi e in caso di aggiungerli alla coda degli ordini da produrre. Per quanto riguarda le anomalie, oltre a storicizzarle, forniamo all'utente una lista di anomalie da gestire; al termine delle attività che mirano alla risoluzione della segnalazione, l'utente può chiudere la notifica mediante il bottone posto sulla destra di ogni anomalia. Un'altra caratteristica fondamentale è il codice colore che abbiamo utilizzato per codificare il bottone di "gestione anomalia" che cambia colore in base al tempo trascorso dalla segnalazione (verde se presente da 1 giorno, giallo se permane per più di 3 giorni e rosso se persiste ulteriormente).

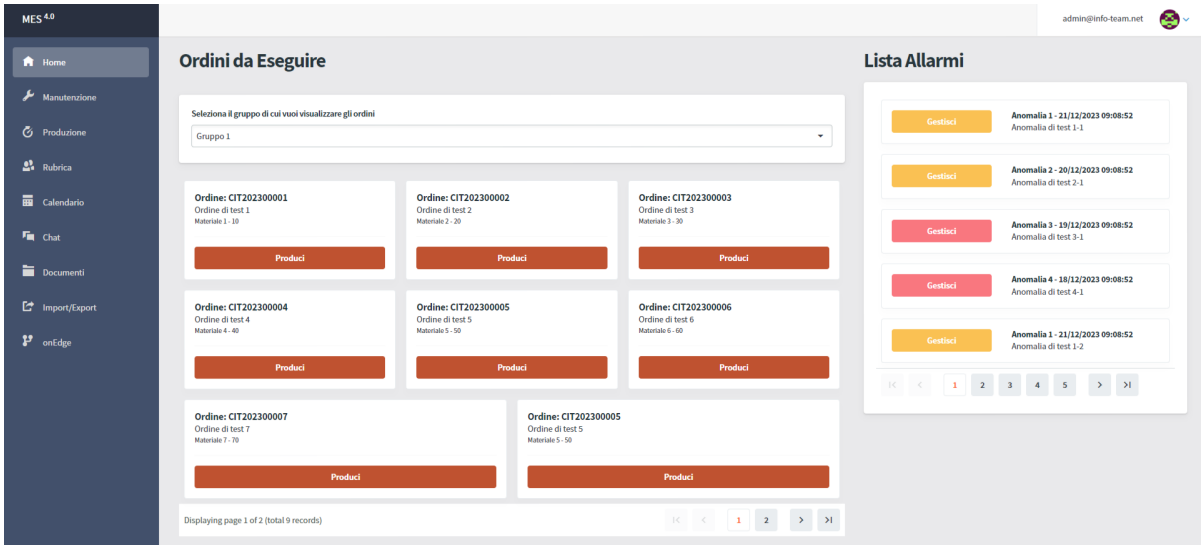

Figura 13: Schermata Home con Ordini da Produrre e Segnalazioni.

4.2.5 Gestione manutenzioni

La maschera per la gestione delle manutenzioni consente di accedere allo storico degli interventi passati, visualizzando in modo dettagliato tutte le informazioni relative ai precedenti interventi di manutenzione. Questa funzionalità permette di tenere traccia delle riparazioni effettuate sui macchinari nel tempo, aiutando gli utenti a pianificare interventi futuri e a mantenere i macchinari in perfetto stato di funzionamento. In sintesi, la maschera per la gestione delle manutenzioni della nostra piattaforma MES è uno strumento essenziale per la gestione efficiente degli interventi di manutenzione, che consente di registrare gli interventi e di monitorare lo storico delle riparazioni effettuate, migliorando l'affidabilità e la produttività dei macchinari.

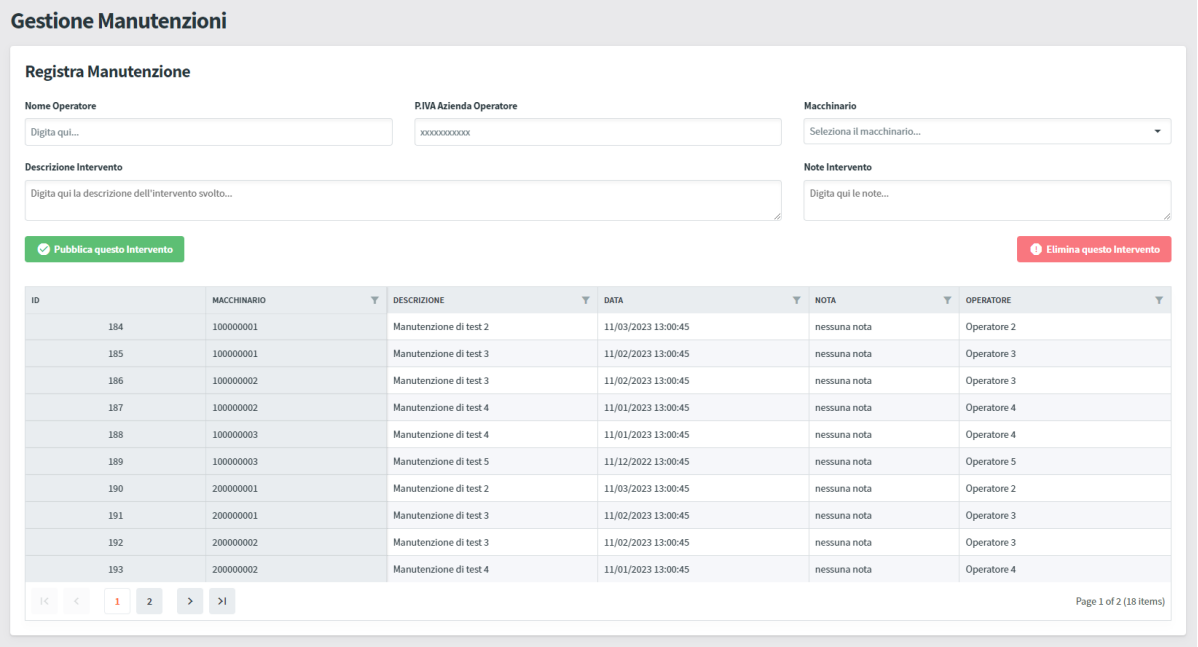

Figura 14: Schermata Manutenzione.

4.2.6 Gestione documentazioni

All'interno della piattaforma MES è stata resa disponibile un'area di salvataggio e consultazione di documenti vari. Questa maschera permette all'utente di caricare sul server documenti di vario tipo, direttamente dal dispositivo con cui sta fruendo della piattaforma; sempre in essa è possibile consultare e scaricare la documentazione caricata. Questa semplice funzionalità permette all'utente di portare con sè manuali, certificazioni o altri documenti utili.

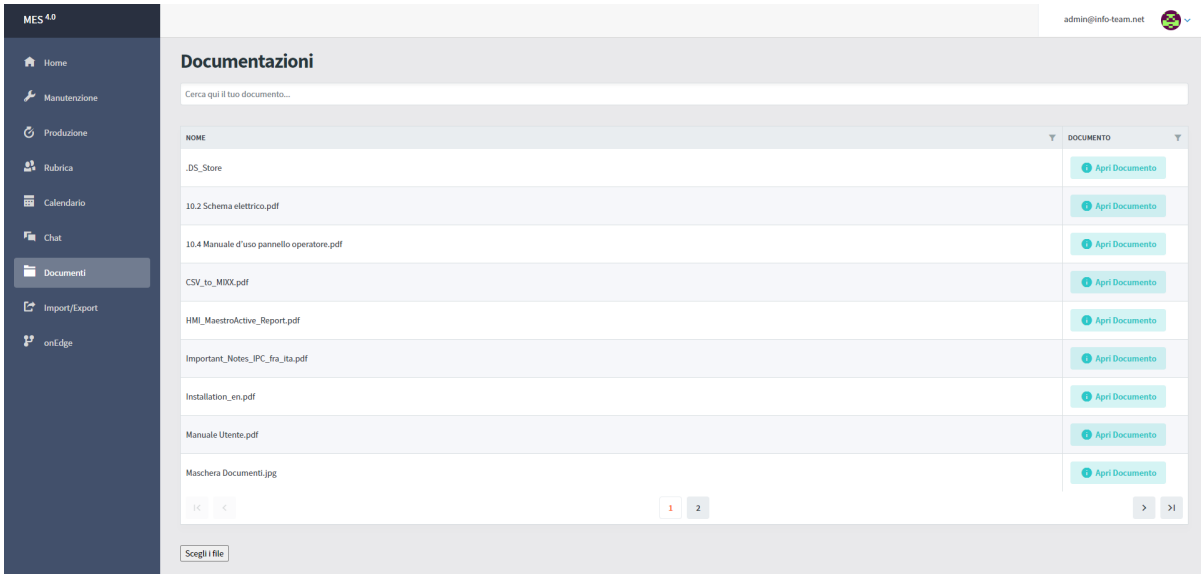

Figura 15: Maschera gestione documenti.

4.2.7 Comunicazione interna

Un argomento molto discusso con le varie realtà aziendali analizzate per la realizzazione del software è quello della comunicazione interna all'azienda (informazioni relative a variazione delle ricette utilizzate in produzione, appuntamenti, promemoria, ecc...). La nostra piattaforma MES include una chat avanzata che utilizza la tecnologia SignalIR per consentire agli utenti di scambiare messaggi con i clienti attivi in tempo reale. Questo sistema garantisce la massima sicurezza nell'invio di informazioni sensibili, come la trasmissione di una ricetta per la produzione. Inoltre, abbiamo previsto la possibilità per gli utenti di salvare i messaggi inoltrati, in modo che siano disponibili anche agli utenti che non erano presenti al momento dell'invio. Questa funzionalità contribuisce a mantenere una comunicazione efficiente e sicura tra tutti i membri del team, riducendo al minimo gli errori e migliorando la produttività.

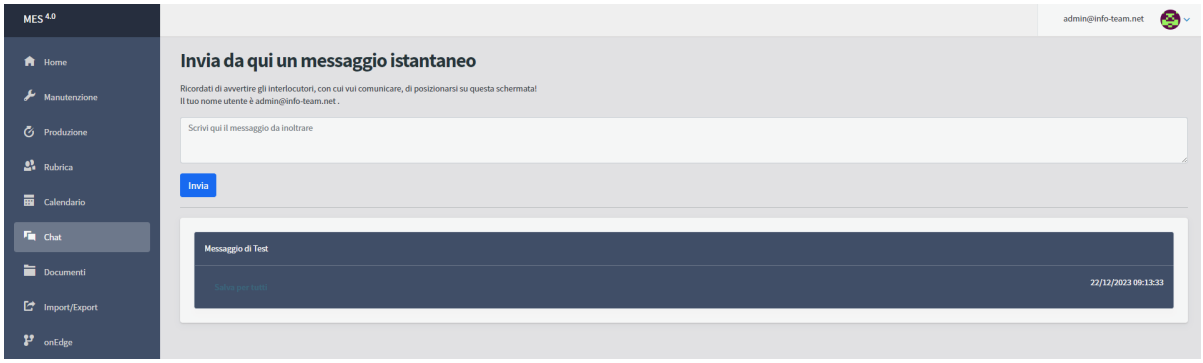

Figura 16: Maschera della chat.

4.2.8 Gestione notifiche

La rilevazione errori, oltre che a registrare dati sull'andamento dei macchinari, serve agli operatori per accorgersi di anomalie ed intervenire in tempi ragionevoli per risolverle e fare ripartire la produzione. Proprio per accellerare il processo di informazione degli addetti alla manutenzione delle macchine, abbiamo deciso di implementare un sistema di notifiche che sfrutta Telegram e premette di ricevere la notifica in una vastissima gamma di dispositivi (tutti quelli che supportano l'applicazione telegram); per gestire i numeri di telefono delle persone a cui inoltrare le notifiche è presente una maschera dedicata.

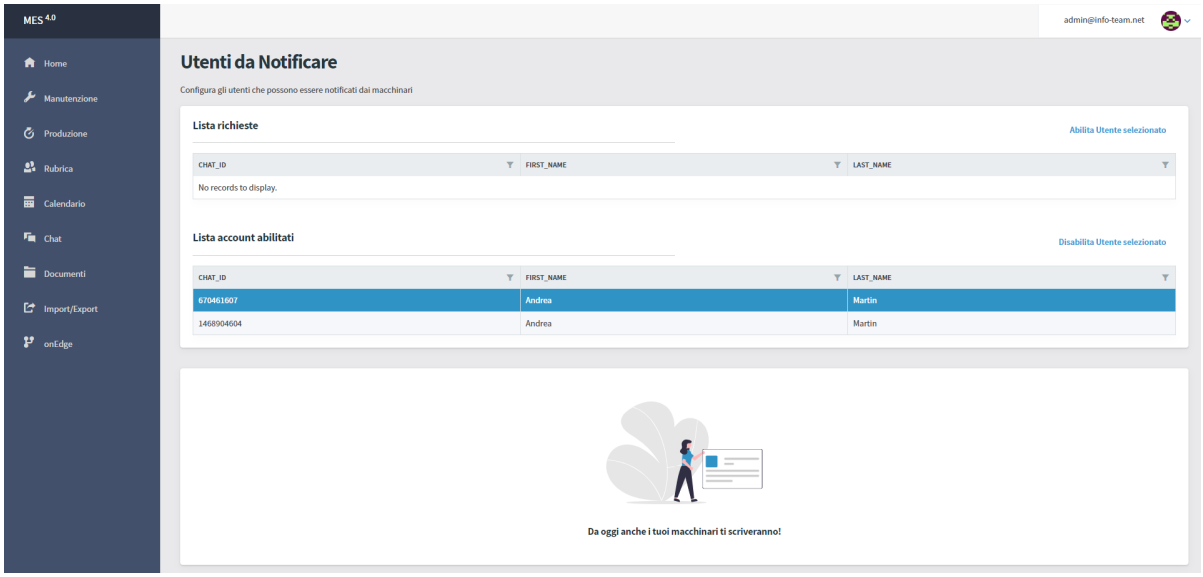

Figura 17: Maschera gestione notifiche.

4.2.9 Contatti rapidi

La piattaforma MES mira a semplificare la comunicazione all'interno dell'azienda lasciando un grande spazio alle persone e automatizzando solamente il necessario per semplificare le operazioni di routine; per questo abbiamo pensato di rendere disponibili i contatti di tutte le figure utili all'interno dell'azienda (manutentori, responsabili, operatori, ecc...) in maniera tale da agevolare la reperibilità e arginare una problematica nel minor tempo possibile.

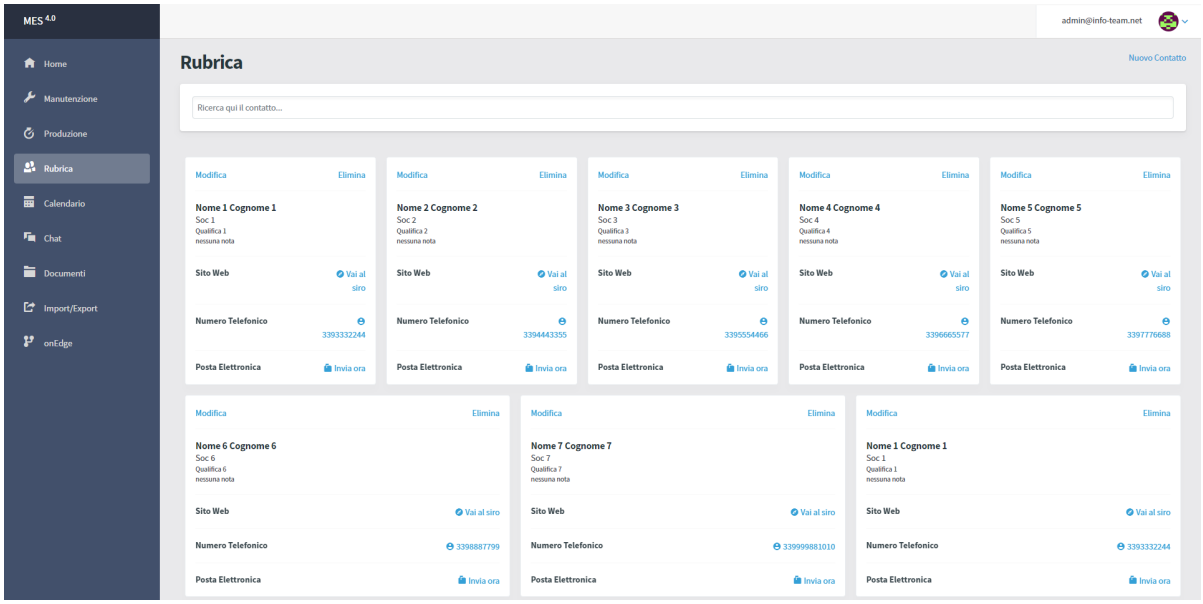

Figura 18: Maschera dei contatti rapidi.

4.2.10 Manuali

Per utilizzare al meglio la piattaforma, pur essendo molto intuitiva, abbiamo pensato di stilare un manuale che illustra le principali funzionalità e come si possono fruire. Oltre al manuale, dopo la vendita e installazione del software, viene fornito un percorso di formazione generale sull'utilizzo delle varie maschere.

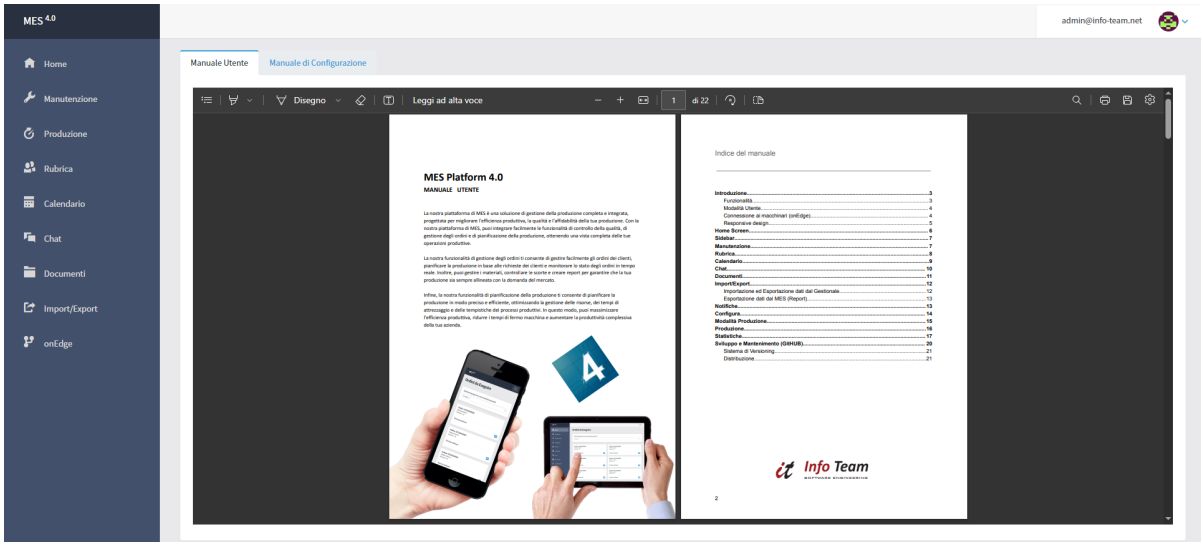

Figura 19: Maschera manualistiche.

4.2.11 Planner

Nella maschera denominata Planner, l'utente ha la possibilità di ottenere con un rapido sguardo l'andamento dell'azienda e comprendere le scadenze delle prossime attività. Questa funzionalità recupera i dati da più maschere e li propone mediante l'ausilio di un calendario; all'interno di esso vengono mostrati (con colorazioni differenti che possono essere configurate nella maschera delle impostazioni) eventi di vario tipo legati a eventi in programma o passati, inizio e fine della produzione di ordine, manutenzioni dei macchinari oppure segnalazioni inserite dagli utenti.

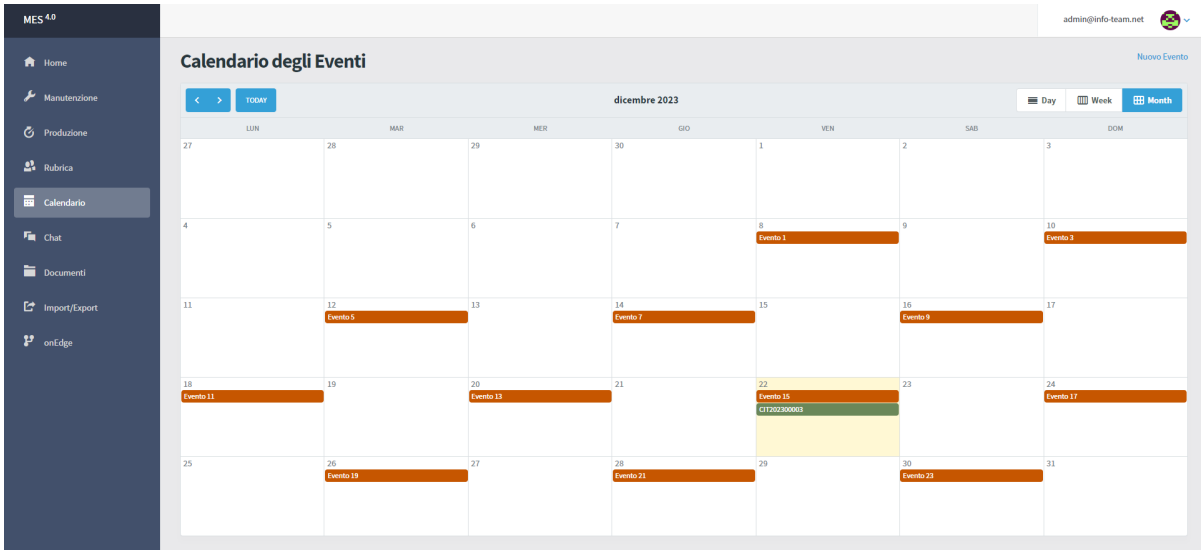

Figura 20: Maschera planner.

4.2.12 Import ed Export dei dati

Per evitare problemi legati allo spostamento da e verso altre piattaforme, abbiamo pensato di introdurre una maschera (in continua evoluzione) che permettesse all'utente di importare ed esportari i dati relativi a Contatti, Produzione, Ordini e Manutenzioni. Attualmente la piattaforma permette di fare un estrazione di tutti i dati in formato csv e di importare Contatti e Ordini sempre nel medesimo formato; inoltre. permettiamo di importare anche gli Ordini dal software UP!ERP (software gestionale sviluppato da un azienda di Udine che personalizziamo) che vendiamo ai nostri clienti.

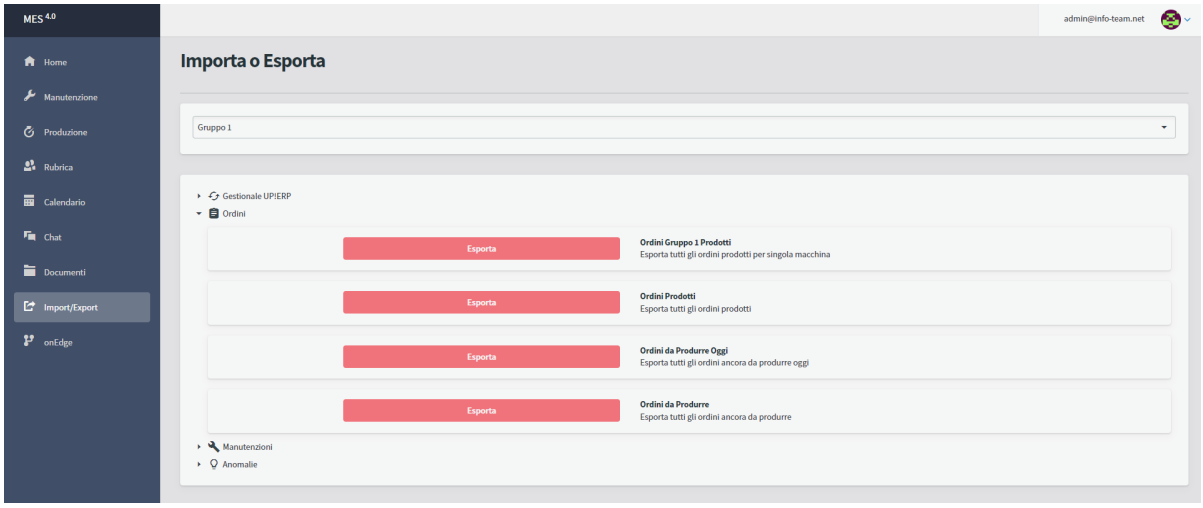

Figura 21: Maschera scambio dati.

4.2.13 Gestione PLC con onEdge

Un'altra maschera fondamentale per la configurazione e messa in funzione del sistema, permette di aggiungere i connettori onEdge (precedentemente citati e illustrati nella sezione 6.2) alla lista macchinari, specificando la configurazione per poter scambiare dati con il PLC del macchinario o impianto. Anche questa maschera è in continua evoluzione dato che cerchiamo di implementare continuamente nuovi macchinari in maniera tale da essere compatibili con la maggior parte degli impianti delle aziende che seguiamo.

Figura 22: Maschera di controllo dei PLC (che comunicano grazie al nostro connettore software onEdge).

4.2.14 Configurazione della piattaforma

La maschera di configurazione consente all'utente di personalizzare la piattaforma in base alle proprie esigenze. Grazie a questa maschera, è possibile configurare la conformazione delle tabelle, attivare o disabilitare gli automatismi della piattaforma e cambiare il colore degli eventi nel calendario. Le varie configurazioni vengono salvate tramite un sistema di cookie direttamente nel dispositivo in cui vengono effettuate. Inoltre, la maschera di configurazione permette di abilitare diverse modalità di visualizzazione, come ad esempio quella con le descrizioni delle maschere o quella dedicata alla visualizzazione delle fasi di produzione (che viene illustrata qualche capitolo più avanti sempre all'interno di questo manuale). La maschera di configurazione, dunque, rappresenta uno strumento fondamentale per personalizzare la piattaforma e adattarla alle esigenze specifiche dell'azienda, garantendo una maggiore efficienza e una gestione più semplice ed efficace del processo produttivo. N.B. non tutti i parametri di configurazione sono stati citati in questa breve introduzione.

5 Connessione con i PLC

La piattaforma MES implementa tutte le interfacce e le logiche per la gestione della produzione senza gestire tutta la parte di comunicazione con i macchinari che tenderebbe a rallentare e appesantire inutilmente il software; tale compito viene delegato ad un altro software che abbiamo progettato e sviluppato a seguito del rilascio delle prime beta della piattaforma. Questo software appena presentato prende il nome di onEdge e si occupa dello scambio di dati con diversi macchinari che utilizzano una vasta gamma di protocolli di comunicazione (questi ultimi vengono approfonditi nella sezione dedicata all'introduzione teorica). Ad oggi è la parte a cui dedichiamo più tempo per rendere compatibili il maggior numero di macchinari; attualmente vengono implementati solamente i macchinari richiesti dai clienti. Molti macchinari utilizzano standard e altri invece utilizzano metodi proprietari a cui dobbiamo prestare maggiore attenzione e maggiore tempo di sviluppo.

5.1 Come funziona onEdge?

Il funzionamento di onEdge dipende moltissimo dalla tipologia di protocollo adottato dal macchinario con cui deve scambiare dati; alcuni protocolli necessitano di essere interpellati ciclicamente mentre altri sono in grado di segnalare al software onEdge ogniqualvolta è presente un aggiornamento relativo allo stato della macchina. In linea di massima possono essere identificati dei macrotask in ogni protocollo, tra i quali: una fase di *inserimento dati nel PLC*, una fase di *controllo stato del macchinario* (con eventuale segnalazione di anomalie), una fase di *storicizzazione dei dati* (con eventuale segnalazione del termine produzione) e una fase di *controllo licenza* (che ci permette di verificare se la licenza dell'utilizzatore è valida o necessita di intervento). Le fasi appena elencate vengono combinate in base alla tipologia del protocollo permettendo la scambio dati ambe le direzioni.

5.1.1 Inserimento dati nel PLC

In questa fase, il connettore software onEdge scarica i dati dalla base di dati, verifica lo stato del PLC e, se non sono presenti ordini in esecuzione, trasferisce i dati del nuovo ordine nella memoria del PLC. Solitamente vengono passati dati come il codice dell'ordine, il numero di pezzi, il part program (solitamente viene indicato il nome oppure il path del programma che è già presente a bordo macchina) e il lotto; possono essere presenti altri campi oltre a quelli elencati (solitamente vengono configurati in base alle necessità del cliente).

5.1.2 Controllo stato del macchinario

Essendo molto importante conoscere lo stato del macchinario (per cercare di prevenire anomalie oppure intervenire per tempo), abbiamo implementato una fase di controllo e validazione dello stato del macchinario; ciclicamente vengono recuperati determinati dati rilevati dai sensori del macchinario e confrontati con alcuni valori configurabili tramite un file presente nella cartella di installazione del software. Un esempio valido potrebbe essere rappresentato dalla rilevazione della temperatura di una fornace e la verifica che questo valore sia sempre compreso tra due cifre definite nel file di configurazione; questo controllo permette di evitare un eccessivo surriscaldamento dell'impianto oppure una temperatura troppo bassa che potrebbe causare una scarsa efficenza e un calo della qualità. Questa parte cruciale di controllo si occupa, nel caso in cui venga rilevato un evento anomalo, di segnalare l'anomalia alle figure selezionate nella maschera dedicata alle notifiche della piattaforma MES.

5.1.3 Storicizzazione dei dati

Nella fase più articolata del connettore software siamo riusciti a codificare un sistema di controllo dello stato dell'ordine in produzione che identifica se l'ordine in esecuzione è terminato, in pausa oppure in avanzamento. Nel caso in cui sia in avanzamento oppure in pausa vengono storicizzati solamente i dati rilevati dalla fase precedente, mentre se viene rilevato il termine di produzione vengono storicizzati anche i dati relativi alla produzione dell'ordine appena prodotto (compresi dati come numero pezzi validi e numero pezzi scartati).

6 Architettura del sistema implementato

Il sistema che abbiamo realizzato e in seguito illustrato in questo documento si compone di due software che assieme permettono di implementare tutte le funzionalità illustrate negli altri capitoli. Abbiamo deciso di suddividere la parte relativa alle logiche e all'interazione con l'utente dalla parte che gestisce la comunicazione con i macchinari.

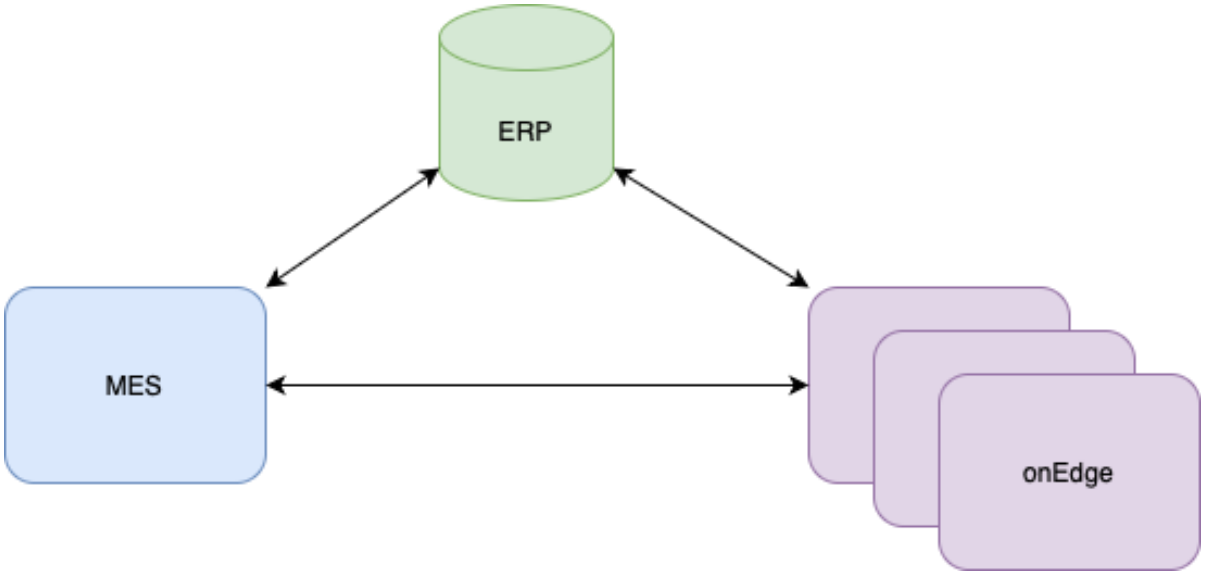

Figura 23: Illustrazione architettura di esempio.

6.1 MES Platform 4.0

La piattaforma MES è sviluppata mediante l'utilizzo dello standard Blazor WASM di casa Microsoft per raggiungere il compromesso ottimale tra prestazioni e stabilità. Esso crea un'interfaccia tra gli operatori e la varietà di dati relativi all'andamento dell'impianto, permettendo l'immediata comprensione e agevolando l'intero flusso di lavoro. Oltre a permettere la semplice gestione del processo produttivo, consente la configurazione della piattaforma stessa e la creazione di profili.

6.2 onEdge

onEdge è una Console Application interamente scritta in C# (.NET Core 7) che permette di scambiare dati con i PLC dei macchinari, verificare lo stato del macchinario e segnalare eventuali anomalie mediante l'utilizzo delle API di Telegram; il tutto configurabile dalla maschera dedicata presente nella piattaforma MES. Attualmente abbiamo implementato alcuni protocolli, tra cui Modbus TCP, OPC-UA, MQTT e scambio dati mediante

l'utilizzo di file (metodo utilizzato da molti macchinari che sono stati customizzati in un secondo momento e non nativamente predisposti).

7 Sviluppo

Nel seguente capitolo verrà esposta tutta la parte legata allo sviluppo delle varie componenti software che realizzano il sistema. Nella prima parte è stata dedicata una quantità di tempo per effettuare una prima analisi di massima (approfondita durante lo sviluppo dei singoli pezzi) per poter avere una prima visione chiara del sistema visto come una blackbox, per poi scomporre il progetto in tanti pezzi da approfondire, ri-analizzare e in seguito realizzare. Abbiamo cercato di suddividere il progetto in piccoli prodotti software, partendo da un idea di sistema a micro-servizi; il sistema che abbiamo realizzato dopo tutta questa fase di analisi e sviluppo permette di interconnettere al core principale (che identifichiamo nel MES) tanti macchinari mediante il connettore (o servizio) onEdge.

7.1 Software MES

Per la realizzazione della piattaforma MES (cfr capitolo 4) abbiamo adottato un framework che ci permettesse di integrare in un'unica soluzione back-end e front-end e cercando di utilizzare un solo linguaggio di programmazione per semplificare la manutenibilità, lo sviluppo e la distribuzione del software. All'interno del sorgente sono stati aggiunti anche i file di configurazione che permettono di distribuire l'applicativo tramite docker (cfr capitolo 2). Di seguito viene illustrata l'implementazione di alcune parti specifiche della piattaforma MES.

7.1.1 Sincronizzazione tra dispositivi

Per evitare che gli operatori lavorino su dati datati, ogni volta in cui un ordine cambia di stato, il dispositivo (che effettua la modifica di stato) "avvisa" tutti i dispositivi che sono collegati, facendo aggiornare i dati. Il codice che viene riportato di seguito, illustra come il client comunica agli altri client in linea di aggiornare i dati e come il esso resta in ascolto per comprendere se altri stanno segnalando un aggiornamento.

Listato 1: Sincronizzazione tra dispositivi.

```
1  //Implementazione SignIR
2  private HubConnection? hubConnection;
3
4  //OnInitialized---------------
5  protected override async Task OnInitializedAsync()
6  {
7      //Implementazione SignalIR
8      hubConnection = new HubConnectionBuilder()
9          .WithUrl(NavigationManager.ToAbsoluteUri("/syncproductionhub
               "))
10         .Build();
```

```
11
12      hubConnection.On<String>("ReceivedMessage",
            HandleReceivedMessage);
13
14      await hubConnection.StartAsync();
15
16      [...]
17  }
18
19  //SignalIR METHOD
20  private async Task Send()
21  {
22      if (hubConnection is not null)
23      {
24          await hubConnection.SendAsync("SendUpdate", "Richiesta di
                Aggiornamento");
25      }
26  }
27
28  public bool IsConnected =>
29      hubConnection?.State == HubConnectionState.Connected;
30
31  public async ValueTask DisposeAsync()
32  {
33      if (hubConnection is not null)
34      {
35          await hubConnection.DisposeAsync();
36      }
37  }
38
39  private async Task HandleReceivedMessage(String message)
40  {
41      //NotificationService.Notify(new NotificationMessage { Severity
            = NotificationSeverity.Error, Summary = "Nuovi", Detail = "
            dati disponibili", Duration = 4000 });
42      await GetOrdini(); // aggiorno gli ordini
43      InvokeAsync(StateHasChanged);
44  }
```

7.1.2 Caricamento documenti su server

Per realizzare un sistema che permettesse di effettuare upload di file direttamente dai dispositivi client verso il server per poterli salvare e fruire in un secondo momento (anche da un dispositivo differente che ha accesso alla piattaforma) abbiamo optato per una gestione del file in maniera tale che il client inoltri al server il file come un byte array, che in seguito alla ricezione viene riconvertito dal server nel formato iniziale e archiviato

all'interno di una cartella appostia. Di seguito viene riportato sia il codice del Client che quello del Server.

Listato 2: Caricamento di un file sul server.

```
/***
* Upload dei file nella cartella del Server
* @param uploadedFile nome del file + byte array
*/
[HttpPost("PostFile")]
public async Task<ActionResult> PostFile(UploadedFile uploadedFile)
{
    var builderCustomConfig = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("Configuration.json", optional: true,
           reloadOnChange: true);

    var configuration = builderCustomConfig.Build();

    String DocumentPath = configuration["DocumentFolderPath"];

    var path = $"C:\\" + DocumentPath + "\\{uploadedFile.FileName}";
    await using var fs = new FileStream(path, FileMode.Create);
    fs.Write(uploadedFile.FileContent, 0, uploadedFile.FileContent.
       Length);
    return new CreatedResult(env.WebRootPath, uploadedFile.
       FileContent);
}

public class UploadedFile
{
    public String FileName { get; set; }
    public byte[] FileContent { get; set; }
}
```

Listato 3: Codice del Client per il caricamento del file.

```
selectedFiles = e.GetMultipleFiles();

foreach(var file in selectedFiles)
{
    try
    {
        using (var ms = new MemoryStream())
        {
            await file.OpenReadStream(50000000).CopyToAsync(ms); //
               MAX 50MB

            var uploadedFile = new UploadedFile();
```

```
12              uploadedFile.FileName = file.Name;
13              uploadedFile.FileContent = ms.ToArray();
14
15              await Http.PostAsJsonAsync<UploadedFile>("api/Document/
                    PostFile", uploadedFile);
16
17          }
18      }
19      catch (Exception ex)
20      {
21          String msg = ex.Message;
22          NotificationService.Notify(new NotificationMessage {
                Severity = NotificationSeverity.Error, Summary = "Errore"
                , Detail = file.Name, Duration = 2000 });
23      }
24
25  }
26  NotificationService.Notify(new NotificationMessage { Severity =
        NotificationSeverity.Success, Summary = "Terminato", Detail = "
        upload dei files", Duration = 2000 });
```

7.2 Connessione con i PLC

Per interconnettere i dati dei PLC al sistema, abbiamo ideato onEdge; questo software è lievemente equiparabile ad uno SCADA ma senza interfaccia grafica. La fase di sviluppo del seguente software si può dividere in diverse fasi ben distinte, tra cui la *realizzazione del main*, l'*interconnessione all'API di Telegram*, l'*importazione delle configurazioni* (specificate nella piattaforma MES), il *controllo licenza*, l'i*mplementazione delle primitive* dei vari protocolli e i rispettivi loop da eseguire per scambiare i dati.

7.2.1 Main

Listato 4: Codice del Main di onEdge.

```
1  //Controllo le credenziali
2  checkAuth();
3
4  //Controllo la licenza e scarico la configurazione ad ogni avvio
5  TTrace_Info info = TTrace.Info();
6
7  //Scarico la configurazione
8  MicrosoftSQL.getConfigurazione(MicrosoftSQL.defaultConnectionString
       ());
9
10 var protocol = ConfigurationManager.AppSettings["protocol"];
```

```
11
12 Costa_ModbusTCP_Dati_Config machine = new
       Costa_ModbusTCP_Dati_Config();
13 Costa_ModbusTCP_Anomalie_Config report = new
       Costa_ModbusTCP_Anomalie_Config();
14
15 switch (Convert.ToInt32(protocol))
16 {
17     case 0:
18         machine = GetClientConfig();
19         report = GetReportConfig();
20         break;
21     default:
22         break;
23 }
24
25 Boolean telegramNotify = Convert.ToBoolean(ConfigurationManager.
       AppSettings["TelegramNotify"]);
26
27 List<TelegramUser> telegramUsers = new List<TelegramUser>();
28
29 if (telegramNotify)
30 {
31     telegramUsers = MicrosoftSQL.GetUtentiDaNotificare(MicrosoftSQL.
           defaultConnectionString());
32 }
33
34 Log.Text("CodLicenza: " + info.CodAgente + " - Scadenza Licenza: " +
       info.DataScadenza);
35 Telegram.multipleNotify(telegramUsers, "   stato appena avviato il
       connettore!\n\n\U00002764 Scadenza Licenza: " + info.DataScadenza
       );
36 int Contatore = 0;
37
38 int Cicli_checkLicence = Convert.ToInt32(ConfigurationManager.
       AppSettings["Cicli_checkLicenza"]);
39
40 while (true)
41 {
42     Console.Clear();
43
44     //Controllo le credenziali
45     checkAuth();
46
47     Console.Clear();
48     switch (Convert.ToInt32(protocol))
49     {
50         case 0:
```

```
51              Log.Logo(machine.ip, machine.GetPort().ToString(),
                    machine.GetConnectionTimeout().ToString(), machine.
                    GetBoudrate().ToString(), ConfigurationManager.
                    AppSettings["GUI_protocol"], (Cicli_checkLicence -
                    Contatore).ToString());
52              break;
53          default:
54              Log.Logo(ConfigurationManager.AppSettings["GUI_protocol"
                    ], (Cicli_checkLicence - Contatore).ToString());
55              break;
56      }
57
58
59      //Controllo la licenza e scarico la configurazione ogni x cicli
            (configurabili momentaneamente dall'App.config)
60      if (Contatore == Cicli_checkLicence)
61      {
62          Contatore = 0;
63          info = TTrace.Info();
64          Log.Confirm("Autenticazione avvenuta con Successo ->
                CodLicenza: " + info.CodLicenza);
65          Telegram.multipleNotify(telegramUsers, "la licenza del
                connettore    stata verificata con SUCCESSO \U00002728\n\
                nScadenza Licenza: " + info.DataScadenza);
66      }
67
68      //Controllo se    PRESENTE una LICENZA
69      if (info.CodLicenza == "")
70      {
71          Log.Error("Error [Program] - Autenticazione Fallita");
72          Telegram.multipleNotify(telegramUsers, "la licenza del
                connettore NON    PRESENTE \U0000274c\n\nScadenza Licenza
                : " + info.DataScadenza);
73          //Controllo la licenza e scarico la configurazione ogni
                volta che la LICENZA NON    PRESENTE
74          info = TTrace.Info();
75
76          //Svuoto i dati
77          Config.cleanData();
78
79          //Attendo 1minuto prima di ripartire (40secondi + 20secondi
                di "fine ciclo")
80          Thread.Sleep(40000);
81      }
82      else if (DateTime.Parse(info.DataScadenza) < DateTime.Now)
83      {
84          Log.Error("Error [Program] - Licenza Scaduta");
85          Telegram.multipleNotify(telegramUsers, "la licenza del
                connettore    SCADUTA \U0000274c\n\nScadenza Licenza: " +
```

```
                info.DataScadenza);
86          //Controllo la licenza e scarico la configurazione ogni
                volta che la LICENZA    SCADUTA
87          info = TTrace.Info();
88
89          //Svuoto i dati
90          Config.cleanData();
91
92          //Attendo 1minuto prima di ripartire (40secondi + 20secondi
                di "fine ciclo")
93          Thread.Sleep(40000);
94      }
95      else
96      {
97          //Gestisco gli ordini e comunico con il macchinario
98          switch (Convert.ToInt32(protocol))
99          {
100             case 0:
101                 ModbusTCP_Client.Application(info, machine, report,
                        telegramUsers);
102                 break;
103             case 1:
104                 Cefla_CSV_Client.Application();
105                 break;
106             case 3:
107                 SCM_m200_Client.Application(telegramUsers);
108                 break;
109             default:
110                 Log.Error("Error [Program] - Non    stato
                        configurato alcun protocollo di comunicazione
                        oppure il protocollo richiesto non    ancora
                        stato implementato");
111                 break;
112         }
113     }
114     Contatore++;
115     Thread.Sleep(20000);
116 }
```

7.2.2 Configurazione

L'importazione delle configurazioni del connettore software avvengono quasi interamente tramite delle *web request* durante la fase di autenticazione; per alcuni casi è necessario importare ulteriori parametri tramite query al DB SQL oppure mediante lettura di file. Un valido esempio di lettura di un file di configurazione è visibile quando il connettore

deve scambiare dati utilizzando il protocollo Modbus TCP; il connettore legge dal file gli indirizzi appositi delle aree di memoria del PLC.

Listato 5: Codice esempio Configurazione aggiuntiva per ModbusTCP.

```
/***
* Scarica la configurazione del client Modbus dal file json (Produzione)
* @return machine che contiene i parametri di configurazione del Client
*/
private static Costa_ModbusTCP_Dati_Config GetClientConfig()
{
    Costa_ModbusTCP_Dati_Config machine = new Costa_ModbusTCP_Dati_Config();
    using (StreamReader r = new StreamReader("..//..//Configurazioni//Costa//Costa_ModbusTCP_Config.json"))
    {
        string json = r.ReadToEnd();
        machine = JsonConvert.DeserializeObject<Costa_ModbusTCP_Dati_Config>(json);
    }

    return machine;
}
```

7.2.3 MODBUS TCP

Per realizzare la comunicazione con i PLC che utilizzano ModbusTCP abbiamo sviluppato delle primitive (esposte come esempio di codice qui sotto) che sono state combinate per leggere e scrivere dal PLC.

Listato 6: Lettura di un Registro tramite ModbusTCP.

```
//Lettura di uno o pi  registri ricercandoli per indirizzo
public static int[] GetRegisters(Costa_ModbusTCP_Dati_Config Machine, int startingAddress, int quantity)
{
    ModbusClient modbusClient = GetClient(Machine);
    int[] results = null;

    try
    {
        if (modbusClient.Available(availableTimeout))
        {
            modbusClient.Connect();
            results = modbusClient.ReadHoldingRegisters(startingAddress, quantity);
```

```
13              Log.Confirm("[ModbusTCP.GetRegisters()] -
                    ReadHoldingRegisters - Letto correttamente!");
14          }
15          else
16          {
17              Log.Warning("[ModbusTCP.GetRegisters()] -  Machine non
                    disponibile");
18          }
19      }
20      catch (ConnectionException cex)
21      {
22          Log.Error("[ModbusTCP.GetRegisters() ConnectionException] -
                " + cex.Message);
23      }
24      catch (SocketException sex)
25      {
26          Log.Error("[ModbusTCP.GetRegisters() SocketException] - " +
                sex.Message);
27      }
28      catch (Exception ex)
29      {
30          Log.Error("[ModbusTCP.GetRegisters() GeneralException] - " +
                ex.Message);
31      }
32      finally
33      {
34          if (modbusClient.Connected)
35          {
36              modbusClient.Disconnect();
37          }
38      }
39      return results;
40  }
```

Listato 7: Scrittura di un Registro tramite ModbusTCP.

```
1  //Scrive uno o pi  registri ricercandoli per indirizzo
2  public static void SetRegisters(Costa_ModbusTCP_Dati_Config Machine,
       int[] register, int startingAddress)
3  {
4      ModbusClient modbusClient = GetClient(Machine);
5
6      try
7      {
8          if (modbusClient.Available(availableTimeout))
9          {
10             modbusClient.Connect();
```

```
11              modbusClient.WriteMultipleRegisters(startingAddress,
                    register);
12              Log.Confirm("[ModbusTCP.SetRegisters()] -
                    WriteMultipleRegisters - Scritto correttamente!");
13          }
14          else
15          {
16              Log.Warning("[ModbusTCP.SetRegisters()] -  Machine non
                    disponibile");
17          }
18      }
19      catch (ConnectionException cex)
20      {
21          Log.Error("[ModbusTCP.SetRegisters() ConnectionException] -
                " + cex.Message);
22      }
23      catch (SocketException sex)
24      {
25          Log.Error("[ModbusTCP.SetRegisters() SocketException] - " +
                sex.Message);
26      }
27      catch (Exception ex)
28      {
29          Log.Error("[ModbusTCP.SetRegisters() GeneralException] - " +
                ex.Message);
30      }
31      finally
32      {
33          if (modbusClient.Connected)
34          {
35              modbusClient.Disconnect();
36          }
37      }
38  }
```

7.2.4 Utilizzo di file

Per la realizzazione delle primitive per lo scambio dati mediante file, abbiamo utilizzato, quando possibile, *parser* già esistenti (come formato .csv che è molto conosciuto). In alcuni casi abbiamo dovuto optare per una soluzione custom come il listato riportato qui sotto che permette la lettura di un file proprietario.

Listato 8: Lettura di un file proprietario per il recupero di dati.

```
1  public static void readReport(SCM_m200_PRO_Report obj)
2  {
3      try
```

```
4       {
5           using (StreamReader reader = new StreamReader(
                ConfigurationManager.AppSettings["SCM_reportPath"] +
                Utils.getTodayPath(".pro")))
6           {
7               String line = reader.ReadLine();
8               int contatoreRighe = 0;
9
10              obj.rowReadCounter = ReadSCMValue();
11
12              String[] splittedRow;
13
14              while (line != null)
15              {
16                  switch (contatoreRighe)
17                  {
18                      case 0:
19                          splittedRow = line.Split(new char[] { ',' })
                                ;
20                          obj.setTempo_Totale(splittedRow[0],
                                splittedRow[1], splittedRow[2]);
21                          obj.setQuantita(splittedRow[3]);
22                          break;
23                      case 1:
24                          splittedRow = line.Split(new char[] { ',' })
                                ;
25                          obj.setNome_Operatore(splittedRow[1]);
26                          break;
27                      default:
28                          if (contatoreRighe >= obj.rowReadCounter)
29                          {
30                              splittedRow = line.Split(new char[] {
                                    ',' });
31                              if (!obj.readRow(splittedRow)) { Log.
                                    Error("Errore [SCM_m200.readReport] -
                                     Errore di lettura di una riga"); };
32                          }
33                          break;
34                  }
35                  line = reader.ReadLine();
36                  contatoreRighe++;
37              }
38              obj.rowReadCounter = contatoreRighe;
39              UpdateSCMState(contatoreRighe);
40              Log.Confirm(obj.rowReadCounter + " righe, relative alla
                    produzione, lette correttamente oggi");
41          }
42      }
43      catch (Exception ex)
```

```
44      {
45          Log.Error("Exception [SCM_m200.readReport(...)] - " + ex.
                Message);
46      }
47  }
```

7.3 Distribuzione con Docker

Una delle principali soluzioni per distribuire in modo stabile e semplice la piattaforma MES è Docker; si creano due semplici file che contengono le righe illustrate qui di seguito.

Listato 9: Docker Compose per pubblicare la Piattaforma MES.

```
1  version: '3.9'
2  services:
3    app:
4      image: mes_image:latest4
5      build: MES_App\Server\
6      container_name: app_container
7      ports:
8        - 55555:80
9      restart: unless-stopped
10     networks:
11       - my_network
12
13   db:
14     build:
15       context: ./DB/
16       dockerfile: Dockerfile
17     container_name: sql_container
18     image: db
19     ports:
20       - "1433:1433"
21     environment:
22       SA_PASSWORD: "Password2023!"
23       ACCEPT_EULA: "Y"
24     volumes:
25       - sql_data:/var/opt/mssql/data
26       - "/C/Documenti/"
27     networks:
28       - my_network
29
30   grafana:
31     image: grafana/grafana:latest
32     container_name: grafana
33     ports:
34       - "3000:3000"
```

```
35      volumes:
36        - grafana_data:/var/lib/grafana
37        - ./datasources.yaml:/etc/grafana/provisioning/datasources/
              datasources.yaml
38      environment:
39        - GF_SECURITY_ALLOW_EMBEDDING=true
40        - GF_AUTH_ANONYMOUS_ENABLED=true
41        - GF_SECURITY_STRICT_TRANSPORT_SECURITY=true
42        - GF_SECURITY_ADMIN_USER=admin
43        - GF_SECURITY_ADMIN_PASSWORD=password
44        - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
45      restart: always
46      networks:
47        - my_network
48
49  volumes:
50    sql_data:
51    grafana_data:
52
53  networks:
54    my_network:
55      driver: bridge
```

Lanciando il file verranno creati alcuni container, tra cui quello della Piattaforma MES, quello relativo al DB e quello di Grafana (cfr capitolo A). Inoltre, è fondamentale ricordare che per la pubblicazione su docker il pacchetto necessita di un ulteriore file di configurazione (di cui potete trovare un esempio di seguito).

Listato 10: Dockerfile per pubblicare la Piattaforma MES.

```
1 # Hosting
2 FROM mcr.microsoft.com/dotnet/aspnet:6.0
3
4 WORKDIR /app
5 COPY bin/Release/net6.0/publish/ .
6
7 EXPOSE 80
8
9 ENTRYPOINT ["dotnet", "MES_App.Server.dll"]
```

7.4 Interfaccia utente

L'interfaccia utente è stata implementata mediante l'utilizzo di un framework di componenti chiamato Radzen e un rapido studio delle principale piattaforme Mes open-source. Per decidere la composizione delle maschere più fruite dagli utenti, abbiamo utilizzato

anche delle brevi interviste con alcuni utenti finali che ci hanno introdotto brevemente il loro lavoro e le varie dinamiche che il sw avrebbe dovuto gestire.

8 Test e Validazione

Sia nella fase conclusiva che durante la realizzazione del software, abbiamo avuto la possibilità di interfacciarci direttamente con gli utilizzatori finali (operatori), in modo tale da poter verificare passo dopo passo che la piattaforma fosse in linea con il workflow aziendale, che creasse valore aggiunto all'azienda e non ostacolasse le operazioni quotidiane degli stessi. Inoltre, sono state adottate tecniche di *logging* per ottenere il maggior numero di informazioni possibili sull'utilizzo della piattaforma e orientare così lo sviluppo in maniera più rapida e precisa possibile. L'unione di questi due feedback ha permesso di diminuire drasticamente i tempi di realizzazione.

8.1 L'importanza delle persone

Tra le informazioni emerse durante la realizzazione di questo progetto, non trascurabile è il fatto che le capacità sviluppate dagli operatori e la memoria storica dell'azienda permettono di realizzare un software qualitativamente migliore e funzionale per l'azienda stessa che lo richiede e che poi lo utilizzerà all'interno dei diversi processi.

8.2 Utilizzo dei Log

Nella piattaforma sono stati implementati diversi sistemi di Log che hanno permesso di realizzare un tracciato dell'utilizzo della stessa, individuare BUG e correggere nel minor tempo possibile le eventuali disfunzioni. Nel caso specifico è stata adottata adottata una classe *ad hoc* che permette di loggare errori o segnalazioni sia in un file locale nel server che in un database non relazionale hostato su una piattaforma cloud. Quest'ultimo permette di convogliare le informazioni relative all'utilizzo della piattaforma in un punto unico e rilasciare aggiornamenti mirati in completa autonomia.

La fase di log su file ha come scopo principale quello di scongiurare il malfunzionamento dello stesso lato cloud; quest'ultimo infatti storicizza dati che in un primo momento vengono visualizzati tramite un'altra piattaforma prodotta e gestita dall'azienda Info Team e a lungo termine permette di ottenere importanti quantità di dati da analizzare e utilizzare nuovamente per lo sviluppo di nuovi progetti.

9 Risultati e Conclusioni

La soluzione implementata permette di automatizzare la produzione dell'azienda che adotta il nostro software e di avere sotto controllo l'andamento generale degli impianti con una valida storicizzazione dei dati.

Inoltre, il sistema interconnette i macchinari compatibili alla rete aziendale, creando dati fondamentali per l'azienda; questi ultimi possono essere visionati e analizzati per incrementare la sua produttività e l'efficienza.

La piattaforma permette di avere il pieno controllo degli ordini e dei macchinari con delle maschere sviluppate appositamente per la pianificazione dell'esecuzione degli ordini, la gestione delle anomalie, la gestione delle manutenzioni, la comunicazione interna dell'azienda, la pianificazione di eventi, la configurazione dei macchinari, il salvataggio di documentazione, la consultazione di documenti precedentemente salvati e la ricezione delle notifiche di allerta.

Abbiamo raccolto dei feedback da parte degli utenti di diverse realtà in cui è stato implementato il MES. A seguito dell'analisi di questi, è possibile affermare che lo scopo iniziale della piattaforma è stato rispettato anche in fase di distribuzione, semplificando in modo non trascurabile le routine giornaliere dei nostri clienti e valorizzando la genialità dell'operatore.

Prevediamo di sviluppare ulteriormente la piattaforma per integrare nuove tecnologie che possano essere efficaci per gli operatori che entreranno in contatto con la nostra piattaforma, aumentare la compatibilità con i macchinari che più comunemente incontriamo nelle realtà con cui abbiamo rapporti quotidiani e correggere eventuali BUG o errori di progettazione.

Le principali implementazioni future saranno mirate principalmente ad aumentare le potenzialità della piattaforma nel campo produzione e analisi dei dati prodotti (facendo uso anche di algoritmi di machine learning).

A Grafana

Per quanto riguarda l'analisi dei dati ottenuti, la piattaforma utilizza un'applicazione web open-source multipiattaforma per interrogare, visualizzare e comprendere questi ultimi; con Grafana è possibile creare, esplorare e condividere tutti i dati attraverso dashboard semplici ed efficienti. Essa raccoglie, correla e visualizza i dati con dashboard strutturate (vedi Figura 24); si configura come la soluzione di visualizzazione e monitoraggio dei dati open source che migliora le prestazioni del sistema e semplifica la risoluzione dei problemi.

Figura 24: Dashboard in Grafana.

Uno dei principali vantaggi di Grafana è quello di unificare varie sorgenti di dati (senza unificare il database) e poter analizzare tutti i dati in un punto unico. Come anticipato in precedenza, i dati possono essere visualizzati su apposite dashboard che gli attribuiscono un significato approfondito; la creazione di questi pannelli è effettuabile tramite query builder interattivi (vedi Figura 25). Le dashboard che vengono create possono essere condivise con tutti i componenti dell'azienda e possono essere visualizzate direttamente nelle maschere del MES mediante un'integrazione apposita.

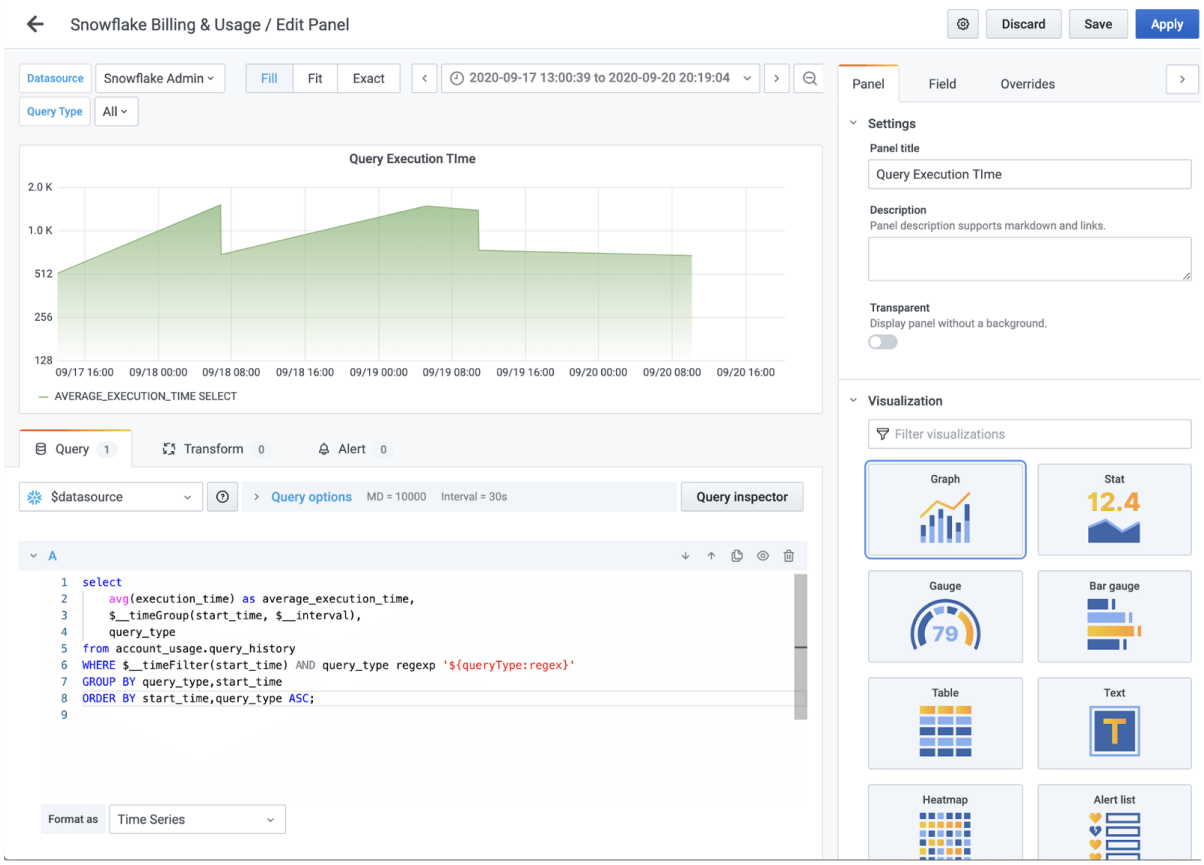

Figura 25: Query builder in Grafana.

Inoltre, con Grafana è possibile creare facilmente regole di avviso multidimensionali che aiutano l'operatore (o chi di competenza) ad identificare eventali anomalie e ottenere informazioni preziose sull'intero sistema.

Questa implementazione è di fondamentale importanza per noi, perchè ci ha permesso di abbassare i tempi di sviluppo ed evitare lo sviluppo di un sistema di dashboard e analisi dei dati.

A.1 Distribuzione con Docker

Una delle principali soluzioni per distribuire in maniera stabile e semplice Grafana è Docker; infatti, nei file di configurazione per la pubblicazione della Piattaforma MES con Docker (cfr capitolo 4.2.14) è prevista l'aggiunta di Grafana.

Listato 11: Docker Compose per pubblicare Grafana.

```
1 grafana:
2     image: grafana/grafana:latest
```

```
3       container_name: grafana
4       ports:
5         - "3000:3000"
6       volumes:
7         - grafana_data:/var/lib/grafana
8         - ./datasources.yaml:/etc/grafana/provisioning/datasources/
              datasources.yaml
9       environment:
10        - GF_SECURITY_ALLOW_EMBEDDING=true
11        - GF_AUTH_ANONYMOUS_ENABLED=true
12        - GF_SECURITY_STRICT_TRANSPORT_SECURITY=true
13        - GF_SECURITY_ADMIN_USER=admin
14        - GF_SECURITY_ADMIN_PASSWORD=password
15        - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
16      restart: always
17      networks:
18        - my_network
```

Lanciando il file di pubblicazione, sarà lanciato sulla macchina server sia il software MES che Grafana rendendoli entrambi raggiungibili in rete a due porte differenti; inoltre è possibile pubblicare solamente Grafana su Docker con le righe di testo mostrate qui sopra.

B Portainer

Per semplificare molte attività legate alla manutenzione, abbiamo deciso di adottare *Portainer* come applicativo per la gestione dei container dei software che pubblichiamo nei macchinari dei clienti. Portainer è un software Open Source e multipiattaforma che, tramite un'interfaccia utente, permette di gestire più facilmente i diversi container e i vari dispositivi IoT (che in ambito industriale vengono denominati *IIoT*); la UI di Portainer non ha vincoli di sistema operativo per la fase di installazione dato che viene installata direttamente su docker (mediante l'inserimento di alcune righe di testo nel docker-compose.yml) e consente di amministrare container, immagini, network e volumi attraverso una dashboard in grado di rendere visibili i dettagli relativi a tutte le entità gestibili.

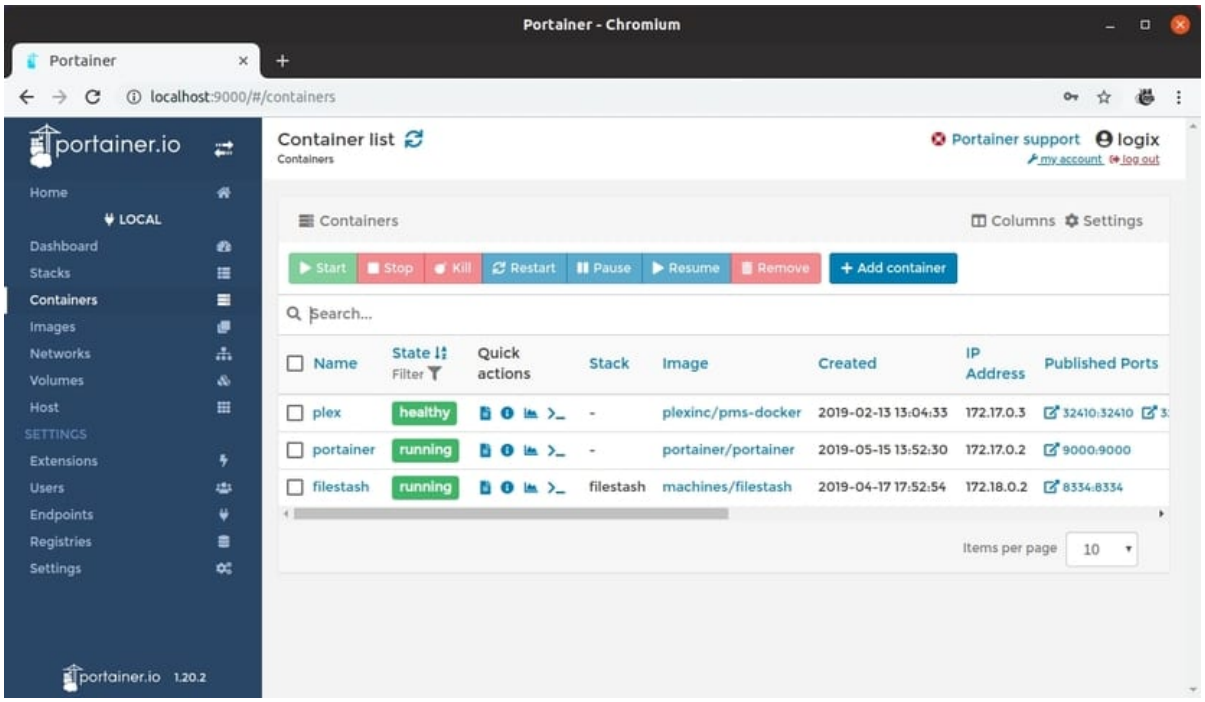

Figura 26: Dashboard di Portainer.

La sezione dedicata ai container permette di reperire con estrema facilità le informazioni riguardanti gli stati, le immagini e le porte impiegate, inoltre permette di gestire i container utilizzando delle azioni standard (avvio, rimozione..).

Nello stesso modo è possibile accedere alle statistiche di funzionamento e ai log delle attività.

L'interfaccia di Portainer consente di visualizzare le statistiche in tempo reale, permettendo di monitorare l'utilizzo delle risorse fornite da CPU e memoria mentre ven-

gono consumate, oppure verificando lo stato delle attività di networking e dei processi funzionanti all'interno di ciascun container.

Portainer non si limita a permettere la sola gestione dell'esistente; tramite la sua UI si possono infatti creare nuovi container o effettuare fasi di deploy semplificati utilizzando un apposito sistema basato sui template; sono disponibili vari modelli pronti all'uso per la messa in produzione di applicazioni e piattaforme diffusamente utilizzate.

B.1 Distribuzione con Docker

Anche Portainer, come Grafana, prevede di essere pubblicato con Docker in maniera semplice e intuitiva, aggiungendo al file di configurazione (docker-compose.yml) le seguenti righe:

Listato 12: Docker Compose per pubblicare Portainer.

```
# Portainer - WebUI for Containers
portainer:
  container_name: portainer
  image: portainer/portainer-ce:latest
  restart: unless-stopped
  networks:
    - default
  command: -H unix:///var/run/docker.sock
  ports:
    - "9000:9000"
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock:ro
    - $DOCKERDIR/appdata/portainer/data:/data
  environment:
    - TZ=$TZ
```

Riferimenti bibliografici

[1] Martin Ford. *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future*. CreateSpace Independent Publishing Platform, 2009. ISBN: 978-1448659814.

[2] Jerry Kaplan. *Humans Need Not Apply: A Guide to Wealth and Work in the Age of Artificial Intelligence*. Yale Univ Pr, 2017. ISBN: 978-0300223576.

[3] Erik Brynjolfsson e Andrew McAfee. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W W Norton Co Inc, 2016. ISBN: 978-0393350647.

[4] Klaus Schwab. *The Fourth Industrial Revolution*. Portfolio Penguin, 2017. ISBN: 978-0241300756.

[5] Peter Thiel e Blake Masters. *Da zero a uno. I segreti delle startup, ovvero come si costruisce il futuro.* Rizzoli, 2015. ISBN: 978-8817080460.

[6] Che cos'è l'Industria 4.0 e perché è importante saperla affrontare. URL: https://www.economyup.it/innovazione/cos-e-l-industria-40-e-perche-e-importante-saperla-affrontare/.

[7] MES: MANIFACTURING EXECUTION SYSTEM, IL MEDIATORE NECESSARIO TRA IMPIANTO E GESTIONALE. URL: https://www.doeeconsulting.it/blog.aspx?w=5.

[8] Qual è la definizione di "PLC"? URL: https://www.unitronics.it/qual-e-la-definizione-di-plc/.

[9] SCADA. URL: https://it.wikipedia.org/wiki/SCADA.

[10] Che cos'è l'ERP? URL: https://www.oracle.com/it/erp/what-is-erp/.

[11] QUALI SONO I PRINCIPALI PROTOCOLLI INDUSTRIALI PER IL MONITORAGGIO DEGLI IMPIANTI? URL: https://www.toolsforsmartminds.com/it/approfondimenti/blog/240-protocolli-industriali-i-principali-per-il-monitoraggio-degli-impianti.

[12] 5 Real-Time, Ethernet-Based Fieldbuses Compared. URL: https://www.manufacturingtomorrow.com/article/2016/05/5-real-time-ethernet-based-fieldbuses-compared/8044/.

[13] AN INTRODUCTION TO PROFINET IO. URL: https://www.rtautomation.com/technologies/profinet-io/.

[14] Il protocollo Modbus. URL: http://www.swappa.it/wiki/uploads/Uni/progetto_LR_mcavenaghi.pdf.

[15] Cos'è MQTT? URL: https://aws.amazon.com/it/what-is/mqtt/.

[16] Che cos'è .Net? URL: https://aws.amazon.com/it/what-is/net/.

[17] Che cos'è Blazor. URL: https://blazordev.it/articoli/che-cosa-e-blazor/.

[18] ASP.NET Core Blazor. URL: https://learn.microsoft.com/it-it/aspnet/core/blazor/?view=aspnetcore-7.0.

[19] Industria 4.0 - Articolo 1, commi da 8 a 13, della legge 11 dicembre 2016, n. 232 - Proroga, con modificazioni, della disciplina del c.d. URL: https://www.mimit.gov.it/images/stories/normativa/Circolare_Agenzia_entrate.pdf.

[20] Beni strumentali - Nuova Sabatini. URL: https://www.mimit.gov.it/it/incentivi/agevolazioni-per-gli-investimenti-delle-pmi-in-beni-strumentali-nuova-sabatini.

[21] Guida alla nuova Sabatini. URL: https://www.safinance.it/guida-alla-nuova-sabatini/.

[22] Transizione 4.0. URL: https://www.mimit.gov.it/it/transizione40.

[23] Piano Transizione 4.0: cos'è e come ottenere fondi per l'Impresa 4.0. URL: https://www.agendadigitale.eu/industry-4-0/piano-transizione-4-0-il-nuovo-credito-di-imposta-tutte-le-aliquote-e-i-beni-compensabili/.

[24] Che cos'è l'Industry 4.0? URL: https://www.sap.com/italy/products/scm/industry-4-0/what-is-industry-4-0.html.

[25] I gateway: cosa sono e perché sono importanti per la comunicazione tra dispositivi, reti e applicazioni. URL: https://moxadistryshop.com/i-gateway-cosa-sono-e-perche-sono-importanti-per-la-comunicazione-tra-dispositivi-reti-e-applicazioni/.

[26] Cos'è la certificazione 4.0. URL: https://www.orimos.dev/industria-4-0/cose-la-certificazione-4-0/.

[27] Blazor WebAssembly. URL: https://blazordev.it/articoli/blazor-webassembly/.

[28] Cos'è il MES (Manufacturing Execution System)? URL: https://www.sap.com/italy/products/scm/execution-mes/what-is-mes.html.

[29] Industria 4.0. URL: https://www.b2a.biz/industria-4-0/.

[30] Cosa sono i Big Data? URL: https://www.oracle.com/it/big-data/what-is-big-data/.

[31] INTEGRAZIONE VERTICALE E ORIZZONTALE: LA QUINTA TECNOLOGIA ABILITANTE DELL'INDUSTRIA 4.0. URL: https://www.focusindustria40.com/integrazione-verticale-e-orizzontale/.

[32] Portainer Docker Compose: FREE MUST-HAVE Container Manager. URL: https://www.smarthomebeginner.com/portainer-docker-compose-guide/.

[33] Cos'è uno SCADA? URL: https://www.wonderware.it/cose-uno-scada/.

[34] IoT Gateway – a Key to connect the World. URL: https://www.daviteq.com/blog/en/iot-gateway-a-key-to-connect-the-world/.

[35] A Complete Overview of Docker Architecture. URL: https://www.cherryservers.com/blog/a-complete-overview-of-docker-architecture.

[36] Docker. URL: https://www.docker.com/.

[37] Grafana. URL: https://it.wikipedia.org/wiki/Grafana.

[38] Grafana Labs. URL: https://grafana.com/.

[39] Portainer per Docker: come installarlo. URL: https://www.ionos.it/digitalguide/server/configurazione/come-installare-portainer-per-docker/.

[40] Portainer, un'interfaccia utente per Docker. URL: https://www.html.it/magazine/portainer-uninterfaccia-utente-per-docker/.

Elenco delle figure

1 Le quattro rivoluzioni industriali. 8
2 I pilastri tecnologici dell'Industria 4.0 14
3 Logica dell'implementazione di un PLC. 22
4 Funzionamento gateway IoT. 25
5 Docker Containers vs Virtual Machines. 26
6 Struttura di un Applicazione Web moderna. 29
7 Esecuzione codice WASM. 30
8 Esecuzione nel Browser. 30
9 Dove si colloca un MES nell'ambiente aziendale. 34
10 Maschera di Login con visualizzazione dei Menu. 35
11 Maschera per gestire l'avanzamento della produzione. 36
12 Maschera riassuntiva stato della produzione. 37
13 Schermata Home con Ordini da Produrre e Segnalazioni. 38
14 Schermata Manutenzione. 39
15 Maschera gestione documenti. 40
16 Maschera della chat. 41
17 Maschera gestione notifiche. 41
18 Maschera dei contatti rapidi. 42
19 Maschera manualistiche. 43
20 Maschera planner. 44
21 Maschera scambio dati. 45
22 Maschera di controllo dei PLC (che comunicano grazie al nostro connettore software onEdge). 45
23 Illustrazione architettura di esempio. 51
24 Dashboard in Grafana. 73
25 Query builder in Grafana. 74
26 Dashboard di Portainer. 77

Elenco dei listati

1 Sincronizzazione tra dispositivi. 54
2 Caricamento di un file sul server. 56
3 Codice del Client per il caricamento del file. 56
4 Codice del Main di onEdge. 57
5 Codice esempio Configurazione aggiuntiva per ModbusTCP. 61
6 Lettura di un Registro tramite ModbusTCP. 61
7 Scrittura di un Registro tramite ModbusTCP. 62
8 Lettura di un file proprietario per il recupero di dati. 63

9 Docker Compose per pubblicare la Piattaforma MES. 65
10 Dockerfile per pubblicare la Piattaforma MES. 66
11 Docker Compose per pubblicare Grafana. 74
12 Docker Compose per pubblicare Portainer. 78

Ringraziamenti

Dedico questo traguardo a tutti voi che, in modi diversi, avete contribuito al mio successo. Grazie per avermi accompagnato in questo viaggio e per aver reso questo giorno così speciale.

Amici e Fidanzata:
Un grazie immenso a voi amici, che avete reso questo percorso una commedia drammatica. Le vostre risate, il vostro sostegno e la vostra pazienza sono state fondamentali. Grazie per avermi ascoltato, incoraggiato e sopportato le mie ansie da prestazione. Le nostre serate insieme, le risate condivise e i momenti di spensieratezza sono stati la valvola di sfogo di cui avevo bisogno per non perdermi tra i pdf e le video lezioni. Vi porto tutti nel mio cuore e vi prometto che continueremo a fare serata anche dopo questo traguardo.

Mamma e papà:
A mamma e papà, dedico questo traguardo con immensa gratitudine. Il vostro sostegno costante e i vostri sacrifici mi hanno permesso di arrivare fin qui. Grazie per aver creduto in me, per avermi insegnato il valore del sacrificio e della perseveranza, e per avermi sempre spinto a dare il massimo. Vi devo tutto e non potrò mai ringraziarvi abbastanza per il vostro immenso amore.

Azienda:
Ringrazio l'azienda per cui lavoro per avermi dato l'opportunità di conciliare il lavoro con lo studio e per avermi supportato durante questo percorso. In particolare, ringrazio Ermes Moras per la sua comprensione e flessibilità. L'esperienza lavorativa ha arricchito il mio bagaglio di conoscenze e mi ha permesso di applicare le teorie apprese in aula a casi concreti.

A tutti coloro che hanno sorriso con me:
Infine, vorrei ringraziare tutte le persone che, nonostante le difficoltà e gli ostacoli, hanno trovato il tempo per sorridere con me. Un sorriso, una parola gentile o un semplice gesto di affetto hanno fatto la differenza in molte giornate difficili. Grazie per avermi ricordato che la vita è bella e che vale la pena sorridere, anche quando le cose si fanno complicate.

Per una persona anonima:
A te, persona speciale che mi hai sempre portato sul palmo della mano, va il mio più profondo e sincero ringraziamento. Il tuo sostegno incrollabile e la tua fiducia in me hanno fatto la differenza in questo percorso. Hai creduto in me anche quando io stesso ne dubitavo, hai saputo incoraggiarmi nei momenti difficili e hai reso i miei errori occasioni di crescita. Grazie per avermi sempre spinto a dare il massimo e per aver visto il meglio in me anche quando io non lo vedevo.