Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Sviluppo di un sistema web on-premise per la configurazione e il calcolo dei costi di pannelli per l'automazione di valvole industriali

Tesi di laurea Triennale

Relatore
Prof. Marco Zanella

Laureando
Enrico Cotti Cottini
Matricola 2077993

Anno Accademico 2024-2025

Watashi: *I got to where I am by believing in my own potential! I'm not sure I'm saying it right... but why does my heart feel so cold? Maybe there's a choice I should have made that would have led to some other possibility?*

Seitarô Higuchi: *You cannot use the word 'possibility' without limitations. Can you become a bunny girl? Can you become a pilot? Perhaps you could. But if you keep focusing your gaze on that which is unrealistic, you never will. The root of all your evil is in always relying on one of your other possibilities to get your wish. You must accept that you are the person here, now, and that you cannot become anyone else other than that person. There is no way that you can lead some worthwhile college life and feel satisfied. I guarantee it, so have confidence! There is no such thing as that rose-colored campus life. Why? Because there is nothing rose-colored in this world. Everything is all a bunch of colors mixed up, you see.*

— The Tatami Galaxy

Dedicato a Elia, mio fratello.

Sommario

Il presente documento illustra il lavoro svolto durante il periodo di stage, della durata complessiva di circa trecento ore, dal laureando Enrico Cotti Cottini presso l'azienda Nuvem s.r.l., in collaborazione con Starline Services S.p.A., società del gruppo Starline specializzata nella progettazione e configurazione di pannelli di automazione per il controllo e l'azionamento di valvole industriali. Il principale obiettivo di questa tesi consiste nell'illustrare il ciclo vitale dello sviluppo di un'applicazione web, realizzata durante il periodo di stage, e dedicata alla configurazione automatizzata dei pannelli della linea Automation. L'applicazione è stata progettata per sostituire l'attuale procedura manuale, basata su fogli Excel complessi e soggetti a errori.

La configurazione è affidata a tecnici specializzati che, grazie alla loro esperienza, utilizzano fogli Excel strutturati per selezionare e combinare i componenti necessari. Questa modalità risulta particolarmente onerosa e dispendiosa in termini di tempo e attenzione, oltre a essere suscettibile a errori e poco scalabile, soprattutto nella fase di redazione delle offerte commerciali.

La configurazione dei pannelli di automazione è un processo ad alta complessità, che richiede l'integrazione di componenti provenienti da diverse aziende del gruppo, come le valvole di Starline S.p.A. e gli attuatori pneumatici di Air Torque S.p.A., nonché da aziende esterne. Ogni soluzione deve essere personalizzata in funzione delle specifiche esigenze operative e ambientali del cliente finale. Il processo di selezione coinvolge numerose variabili tecniche e operative: materiali, dimensioni, valori di coppia, compatibilità tra componenti, limiti meccanici e condizioni di esercizio particolari, quali alte temperature o ambienti sottomarini. Le variabili e le condizioni da considerare sono molteplici e spesso interdipendenti, rendendo il processo ulteriormente complesso e articolato. Inoltre, l'applicazione di formule fisiche e il rispetto di vincoli tecnici, commerciali e ambientali rendono difficile standardizzare il procedimento, aumentando il rischio di errori e rallentando la formulazione delle offerte.

L'applicazione web sviluppata mira ad automatizzare e ottimizzare l'intero processo di configurazione. Il sistema guida l'utente nella selezione dei componenti compatibili e ottimali, calcola dinamicamente il costo complessivo del pannello e genera un output strutturato, pronto per la preventivazione commerciale. Tra i principali requisiti figurano la gestione centralizzata e aggiornata della base dati, l'elaborazione automatica delle combinazioni valide, la valutazione delle prestazioni tramite formule fisiche e l'integrazione con i flussi di lavoro aziendali. L'adozione di tale sistema permette di incrementare l'efficienza, migliorare la qualità delle configurazioni e ridurre tempi ed errori nella formulazione delle offerte, ponendo le basi per una futura implementazione produttiva.

"For Those Who Come After"

— Gustave, Clair Obscur: Expedition 33

Ringraziamenti

Innanzitutto, desidero esprimere la mia più sincera gratitudine al Prof. Marco Zanella, relatore della mia tesi, per la disponibilità, la pazienza e i preziosi consigli che mi ha fornito durante l'intero percorso di stesura.
Un ringraziamento speciale va ad Andrea Paris, mio tutor esterno, e a tutti i ragazzi della Nuvem che mi hanno accolto durante il periodo di stage. Grazie per avermi dato l'opportunità di crescere professionalmente e per avermi insegnato così tanto con competenza e passione.
Un grazie profondo va alla mia mamma e al mio papà, per il sostegno costante e per avermi dato la possibilità di studiare a Padova.
Grazie a tutta la mia famiglia, per l'affetto, la pazienza e la presenza costante anche nei momenti più impegnativi.
Grazie a mio fratello, che mi ha insegnato tantissimo su questo mondo e continua a farlo ogni giorno.
Grazie a tutti i miei amici, quelli di sempre e quelli che ho incontrato lungo il percorso. Anche se il tempo per vedersi diventa sempre meno man mano che si cresce, ogni volta che ci si ritrova è come se il tempo non fosse mai passato.
Grazie a chi mi fa scoprire perle dell'internet che riescono ancora a sorprendermi lol, a chi mi sprona a fare più sport — che ultimamente sto trascurando, lo ammetto — e a prendermi cura del mio corpo. Grazie a chi mi dà consigli sinceri quando tutto sembra troppo difficile, a chi ha sempre piacere di uscire per due birre, ai compagni del DDR per le serate passate in sala giochi, e a chi ha piacere di passare del tempo insieme, indipendentemente da quello che si fa.
Un ringraziamento speciale anche ai ragazzi di Padova che mi hanno aiutato a inserirmi in una città che inizialmente non conoscevo e in cui mi sono ritrovato da solo. Mi avete fatto sentire accolto fin da subito.
Grazie a tutti i miei compagni di corso con cui ho affrontato questa lunga battaglia: auguro a chi deve ancora affrontare il progetto di "SWE" di superarlo senza finire in manicomio (Mai Mola!).
Un pensiero anche ai ragazzi più grandi, che con grande disponibilità hanno condiviso le loro conoscenze e appunti che hanno reso il nostro percorso un po' più semplice, tracciando una strada, o una rotta se preferite la "Grande Era della Pirateria".
Infine, grazie a chi sarebbe stato felicissimo di vedermi arrivare fin qui, ma purtroppo non può esserci. Questo traguardo è anche per voi.

Padova, Luglio 2025 Enrico Cotti Cottini

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo capitolo si presentano i profili di: Nuvem S.r.l., azienda ospitante dello stage, e Starline Services S.p.A., cliente finale del prodotto software. Inoltre, viene illustrato il contesto applicativo, ovvero l'ambito su cui si modella il prodotto software, insieme a una sezione dedicata alla necessità del prodotto stesso e all'idea che ne ha guidato la realizzazione.

1.1 L'azienda Nuvem S.r.l.

1.1.1 Profilo e attività

Nuvem S.r.l. è una società fondata il 6 giugno 2023, con sede legale a Pian Camuno (BS)[1]. L'azienda opera nel settore della consulenza informatica, offrendo servizi volti all'ottimizzazione e all'efficientamento dell'operatività delle imprese.

1.1.2 Offerta tecnologica

L'offerta di Nuvem S.r.l. comprende:

- **Cyber Security**: valutazione e miglioramento della sicurezza delle infrastrutture IT attraverso test e consulenze specifiche.
- **Migrazioni in Cloud**: assistenza nel trasferimento di infrastrutture aziendali su piattaforme cloud, inclusi server applicativi e sistemi di backup.
- **Sviluppo Software**: realizzazione di soluzioni software personalizzate per supportare le attività aziendali, dai siti web alle applicazioni per l'ottimizzazione dei processi produttivi.
- **Formazione**: corsi dedicati all'utilizzo efficace di software aziendali, per garantire una piena padronanza degli strumenti a disposizione.

1.1.3 Mission e valori

La mission di Nuvem S.r.l. è comprendere le necessità dei clienti e trasformarle in soluzioni concrete, senza intermediari, garantendo un contatto diretto tra tecnici e

[1] 10.

clienti per evitare fraintendimenti. L'azienda si impegna a proporre soluzioni adeguate alle esigenze specifiche, con l'obiettivo primario di ottenere la fiducia e la soddisfazione dei clienti.

1.1.4 Team e competenze

Il team di Nuvem S.r.l. è composto da professionisti con competenze complementari nei seguenti ambiti:

- Infrastruttura IT
- Sviluppo Software
- Cloud Computing
- Backup e Sicurezza
- Consulenza e Formazione

1.2 L'azienda Starline Services S.p.A.

1.2.1 Profilo e attività

Starline Services S.p.A., fondata nel 1976 e parte del gruppo Samson, è un'azienda italiana specializzata nella progettazione e produzione di valvola a sfera[g][2] per applicazioni industriali, in particolare nei settori Oil & Gas[g], chimico e di processo. L'intero processo produttivo si svolge nello stabilimento di Costa di Mezzate (BG)[3], consentendo un controllo diretto su tutte le fasi, dall'ingegnerizzazione al collaudo finale. La produzione si basa su materiali provenienti da fornitori europei selezionati e componentistica forgiatura da aziende italiane qualificate.

1.2.2 Offerta tecnologica

L'offerta tecnologica di Starline Services S.p.A. comprende valvole a sfera disponibili in una vasta gamma dimensionale, che va da $\frac{1}{4}$" (circa 6 millimetri) fino a 36" (circa 900 millimetri) di diametro[4]. Questa ampia scelta consente di soddisfare sia le esigenze di piccoli impianti sia quelle di grandi infrastrutture industriali. Per quanto riguarda la resistenza alle pressioni di esercizio, le valvole sono progettate per operare fino a una pressione nominale (PN[g]) di 420, corrispondente alla classe classe 2500[g]secondo gli standard internazionali, oppure fino a 15.000 psi (pound per square inch).

1.2.3 Controllo qualità e certificazioni

Tutti i sistemi vengono assemblati e testati internamente, secondo procedure di qualità sia interne sia su richiesta del cliente. L'azienda ha sviluppato competenze specifiche nella certificazione SIL[g], fornendo unità certificate fino al livello 3, in funzione delle esigenze applicative. La filosofia aziendale prevede la standardizzazione della qualità su tutto il processo produttivo, anche attraverso l'esecuzione di FAT[g].

1.2.4 Digitalizzazione e innovazione dei processi

Starline Services S.p.A. ha avviato un percorso di digitalizzazione, con particolare attenzione all'ottimizzazione delle procedure di configurazione e preventivazione dei sistemi automatizzati. L'azienda ha espresso la necessità di strumenti software in grado di stimare dinamicamente il costo dei prodotti in base alla composizione selezionata e alle specifiche richieste dal cliente, superando i limiti delle attuali procedure manuali.

1.2.5 Sintesi

In sintesi, Starline Services S.p.A. si configura come realtà specializzata nella produzione di valvola a sfera e sistemi automatizzati per l'industria, con una struttura produttiva integrata e un approccio orientato alla qualità, alla sicurezza e all'innovazione.

[2] i termini presenti nel glossario o nella tabella acronimi vengono contrassegnati con [g]alla loro prima occorrenza

[3] 14.

[4] " indica l'unità di misura del pollice, che corrisponde a 25,4 millimetri

1.3 Contesto Applicativo

Prima di procedere con l'idea e l'analisi del sistema software, risulta fondamentale acquisire una conoscenza, almeno a livello funzionale, del contesto applicativo e delle dinamiche operative dell'impianto fisico oggetto di modellazione. Una comprensione solida delle logiche di funzionamento, delle interazioni tra i componenti e delle esigenze operative dell'ambiente industriale costituisce infatti il presupposto indispensabile per una corretta strutturazione delle soluzioni software, in termini sia di coerenza con i requisiti reali sia di efficacia nell'automazione dei processi.

L'impianto fisico oggetto di modellazione si configura come un sistema integrato per l'automazione industriale, la cui architettura si basa sull'interazione sinergica tra tre elementi fondamentali: la valvola a sfera, l'attuatore[g]e il pannello di supervisione, come illustrato in Figura 1.1. Questa tipologia di impianto trova ampia applicazione in contesti industriali ad elevata complessità, quali impianti chimici, raffinerie, centrali energetiche e infrastrutture per il trasporto di fluidi, dove la gestione sicura, efficiente e affidabile dei processi rappresenta un requisito imprescindibile.

In termini generali, il funzionamento dell'impianto si fonda sulla capacità di regolare e monitorare il flusso di fluidi all'interno di una rete di tubazioni, garantendo la possibilità di intervenire tempestivamente sia in condizioni operative ordinarie sia in situazioni di emergenza. La valvola a sfera, installata in punti strategici della rete, svolge la funzione di dispositivo di intercettazione e regolazione, consentendo l'apertura, la chiusura o la modulazione del passaggio del fluido. L'attuatore, selezionato in base alle specifiche esigenze di coppia[g], frequenza di manovra e condizioni ambientali, è responsabile della movimentazione automatica della valvola, assicurando precisione e affidabilità anche in presenza di cicli operativi intensi.

Il pannello di supervisione rappresenta l'interfaccia principale tra il sistema fisico e gli operatori. Esso consente il monitoraggio in tempo reale dello stato dei dispositivi, la ricezione di segnalazioni di allarme e la gestione operativa delle manovre, intervenire in caso di anomalie e attuare procedure di sicurezza automatizzate, riducendo al minimo i tempi di risposta e il rischio di errore umano.

Nel dettaglio, in un impianto di distribuzione di gas naturale, tale architettura consente, ad esempio, di isolare rapidamente una sezione della rete in caso di perdita o di effettuare regolazioni di portata in base alla domanda. In questi casi, la scelta dell'attuatore e delle caratteristiche della valvola a sfera è determinata da parametri quali pressione, temperatura e natura del fluido, al fine di garantire la massima affidabilità operativa. Analogamente, nel settore Oil & Gas, la presenza di fluidi corrosivi o ad alta pressione impone la selezione di materiali e componenti certificati, nonché la considerazione di variabili tecniche aggiuntive, come la compatibilità dei materiali, le dimensioni delle connessioni e le condizioni ambientali estreme (ad esempio, installazioni in zone ATEX[g]o in ambienti marini).

L'architettura descritta si distingue per la capacità di integrare dispositivi di campo e sistemi di supervisione in un'unica infrastruttura coerente, in grado di supportare sia le esigenze di automazione ordinaria sia le procedure di gestione delle emergenze. La corretta progettazione e configurazione di questi sistemi costituisce il presupposto per il raggiungimento di elevati standard di sicurezza, efficienza e conformità normativa nei processi industriali automatizzati.

Figura 1.1: Modello di sistema di controllo: pannello di controllo, attuatore e valvola a sfera.

Il pannello di controllo invia comandi elettrici all'attuatore, che trasforma l'energia elettrica in movimento meccanico, regolando l'apertura e la chiusura della valvola a sfera per il controllo del flusso.

1.3.1 Valvola

Descrizione generale. La valvola a sfera è un dispositivo meccanico progettato per regolare, interrompere o permettere il flusso di un fluido (liquido o gas) all'interno di una tubazione o di un impianto. Dal punto di vista costruttivo, esistono numerose tipologie di valvole, tra cui le valvole a sfera, a farfalla, a globo e a saracinesca. La valvola a sfera, in particolare, è costituita da un corpo principale all'interno del quale è alloggiata una sfera forata (come mostrato nello schema di Figura 1.2) che, ruotando attorno al proprio asse, apre o chiude il passaggio del fluido. Il funzionamento è semplice: quando il foro della sfera è allineato con la tubazione, il fluido può scorrere

liberamente; quando la sfera viene ruotata di 90°, il passaggio viene bloccato.

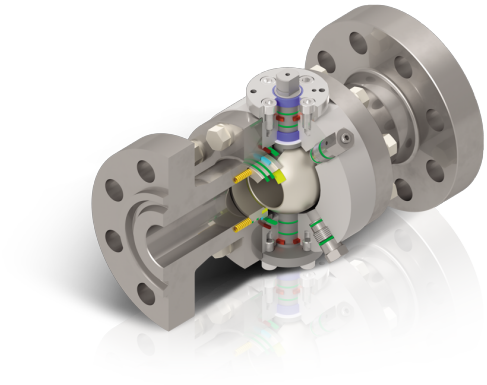

Figura 1.2: Schema di una valvola a sfera: la rotazione della sfera (causata dall'attuatore) permette l'apertura o la chiusura del flusso.

Le valvole sono realizzate con materiali metallici (alluminio, acciaio al carbonio, acciaio inox, leghe speciali) o plastici, scelti in base alla natura del fluido, alla pressione e alla temperatura di esercizio. Gli elementi principali comprendono il corpo, la sfera, le guarnizioni di tenuta, il sistema di azionamento e le connessioni alle tubazioni. Le applicazioni spaziano dal settore idrico a quello chimico, petrolchimico, energetico e alimentare.

Soluzioni Starline Services S.p.A. Nel caso specifico di Starline Services S.p.A., la produzione si concentra sulle valvola a sfera per applicazioni industriali ad alta criticità, come Oil & Gas, chimico e di processo. Le valvole Starline sono disponibili in un ampio range dimensionale, da $\frac{1}{4}$" a 36" di diametro, e sono progettate per resistere a pressioni fino a 420 bar[g](classe 2500) o 15.000 psi[g]. I materiali impiegati provengono da fornitori europei qualificati e la componentistica è realizzata tramite forgiatura in aziende italiane certificate.

Ogni valvola è sottoposta a rigorosi controlli di qualità, tra cui test di tenuta, prove di pressione e collaudi funzionali, e può essere certificata secondo i principali standard internazionali (es. API[g], ISO[g], SIL[g]). Un esempio concreto di applicazione è la gestione di linee di trasporto gas in cui, in caso di emergenza, una valvola a sfera Starline può isolare rapidamente una sezione dell'impianto, garantendo sicurezza e continuità operativa anche in condizioni estreme.

1.3.2 Attuatore

Descrizione generale. L'attuatore (un esempio è mostrato in Figura 1.3) è il componente che permette la movimentazione automatica della valvola, trasformando un segnale di controllo (elettrico, pneumatico o idraulico) in un movimento meccanico di apertura, chiusura o regolazione.

Figura 1.3: Modellino a scopo didattico di un attuatore pneumatico AirTorque.

Gli attuatori pneumatici sfruttano la pressione dell'aria compressa, quelli elettrici utilizzano motori e sistemi di riduzione, mentre quelli idraulici impiegano la pressione di un fluido. La scelta dell'attuatore dipende da variabili come la coppia necessaria per muovere la valvola, la velocità di risposta richiesta, la frequenza di azionamento e le condizioni ambientali (ad esempio presenza di atmosfere ATEX o temperature estreme).

Gli attuatori sono dotati di sistemi di feedback (finecorsa, sensori di posizione) che permettono di conoscere in ogni momento lo stato della valvola. In ambito industriale, la corretta selezione dell'attuatore è fondamentale per garantire affidabilità, sicurezza e durata del sistema.

Soluzioni Starline Services S.p.A. Per quanto riguarda i sistemi forniti da Starline Services S.p.A., l'integrazione tra valvola a sfera e attuatore avviene secondo criteri di compatibilità meccanica e funzionale, rispettando le normative di settore. Gli attuatori pneumatici, spesso forniti da Air Torque S.p.A. (altra società del gruppo), sono progettati per garantire rapidità di risposta e robustezza anche in ambienti ostili, come impianti offshore o raffinerie. Un esempio pratico è rappresentato dall'automazione di valvole installate su linee di processo chimico, dove l'attuatore pneumatico consente di eseguire cicli di apertura e chiusura frequenti, mantenendo elevati standard di sicurezza e affidabilità.

1.3.3 Pannello di controllo

Descrizione generale. Il pannello di controllo (illustrato in Figura 1.4) è l'interfaccia attraverso cui l'operatore può monitorare e gestire il sistema di automazione.

Figura 1.4: Esempio di pannello di controllo industriale Starline con interfaccia per la supervisione e il comando remoto.

Dal punto di vista costruttivo, può essere costituito da una combinazione di dispositivi hardware (pulsanti, selettori, display, indicatori luminosi) e software (interfacce grafiche HMI, sistemi SCADA). Il pannello consente l'invio di comandi (apertura, chiusura, regolazione), la visualizzazione dello stato dei componenti (posizione valvola, allarmi, parametri di processo) e l'acquisizione di dati per la diagnostica e la manutenzione. Nei sistemi più evoluti, il pannello è integrato con reti di comunicazione industriale e piattaforme di supervisione remota.

Soluzioni Starline Services S.p.A. Nel contesto Starline, la configurazione dei pannelli di controllo è un processo altamente personalizzato, che richiede la selezione e l'integrazione di molteplici componenti in funzione delle specifiche esigenze del cliente e delle condizioni operative. Ad esempio, in un impianto Oil & Gas, il pannello può essere dotato di sistemi di sicurezza ridondanti, indicatori di stato certificati per ambienti ATEX e interfacce per il collegamento a sistemi SCADA aziendali. La complessità della configurazione è ulteriormente aumentata dalla necessità di garantire la compatibilità tra componenti provenienti da diversi fornitori e di rispettare vincoli tecnici, normativi e ambientali.

Un esempio reale di utilizzo di un pannello di controllo di questo tipo si può

riscontrare in un impianto di distribuzione di fluidi industriali, come una stazione di pompaggio per il trasferimento di prodotti chimici. In tale contesto, il pannello di controllo consente all'operatore di monitorare in tempo reale lo stato di apertura o chiusura delle valvole installate lungo le diverse linee di processo e di comandare a distanza gli attuatori associati, regolando il flusso in base alle esigenze operative o alle condizioni di sicurezza.

Ad esempio, durante una fase di caricamento di un serbatoio, l'operatore può utilizzare il pannello per aprire la valvola di ingresso e, contemporaneamente, monitorare i parametri di pressione e livello tramite sensori collegati al sistema. In caso di superamento di soglie critiche, il pannello può attivare automaticamente la chiusura delle valvole interessate e segnalare l'anomalia tramite allarmi visivi e acustici, garantendo così la sicurezza dell'impianto e la protezione degli operatori. Inoltre, il pannello permette di programmare sequenze di apertura e chiusura delle valvole in modo automatico, ottimizzando i tempi di processo e riducendo il rischio di errori manuali.

1.4 Prodotto Software

1.4.1 La Necessità

La crescente complessità dei sistemi di automazione industriale e la varietà delle soluzioni richieste dal mercato hanno reso sempre più evidente l'inadeguatezza degli strumenti tradizionali per la gestione dei processi di configurazione e preventivazione. Attualmente, nelle aziende del settore, la predisposizione di offerte commerciali e la progettazione dei sistemi si basano prevalentemente sull'utilizzo di fogli di calcolo Excel di grandi dimensioni, nei quali vengono raccolte e correlate informazioni relative a componenti fisici, vincoli tecnici, normative di riferimento e costi.

Questa modalità operativa, seppur consolidata, presenta numerose criticità. In primo luogo, la gestione manuale dei dati comporta un notevole dispendio di tempo e risorse: ogni richiesta di preventivo richiede ai tecnici specializzati una riorganizzazione puntuale delle informazioni, con la necessità di verificare la disponibilità e la compatibilità dei componenti rispetto ai vincoli progettuali. Tali vincoli comprendono la selezione dei materiali più idonei, la verifica delle forze e delle pressioni supportate, il rispetto dei valori minimi e massimi di esercizio, nonché la compatibilità tra componenti di marchi diversi, aspetto che può precludere la possibilità di integrazione tra elementi non omogenei.

A ciò si aggiunge la necessità di rispettare le normative vigenti e le specifiche esigenze del cliente finale, che impongono il controllo di una molteplicità di variabili tecniche e commerciali, spesso interconnesse e soggette a frequenti aggiornamenti. Il processo, ripetuto per ogni nuovo progetto o cliente, espone l'azienda a un elevato rischio di errore umano, con possibili ripercussioni sulla qualità delle offerte e sulla competitività sul mercato. Inoltre, la difficoltà di mantenere una visione aggiornata e centralizzata dei dati può generare inefficienze, duplicazioni e ritardi nella risposta alle richieste dei clienti.

Queste criticità si riflettono negativamente sull'efficienza complessiva dell'organizzazione, rallentando i tempi di risposta e aumentando i costi operativi. In un contesto industriale sempre più orientato alla rapidità, alla precisione e alla personalizzazione delle soluzioni, emerge quindi la necessità di adottare strumenti software in grado di automatizzare e razionalizzare l'intero iter di configurazione e preventivazione. Un sistema di questo tipo consentirebbe non solo di ridurre drasticamente il carico lavorativo e i tempi di elaborazione, ma anche di ottimizzare la gestione delle informazioni, minimizzare il margine di errore e garantire una maggiore coerenza e tracciabilità dei processi. L'automazione rappresenta, pertanto, una risposta concreta alle esigenze di efficienza, accuratezza e scalabilità richieste dal mercato attuale, ponendo le basi per un significativo miglioramento della qualità e della competitività aziendale.

1.4.2 L'idea

Alla luce delle problematiche evidenziate, il prodotto software oggetto di questa tesi si propone di introdurre un approccio innovativo e sistematico alla gestione dei processi di configurazione e preventivazione dei sistemi di automazione industriale. L'obiettivo è la realizzazione di una piattaforma centralizzata e automatizzata, capace di gestire in modo efficiente la persistenza dei dati relativi a tutti i componenti aziendali, garantendo una visione unificata, aggiornata e facilmente accessibile delle caratteristiche tecniche, della disponibilità e dei costi di ciascun elemento.

Il sistema dovrà supportare l'utente nella creazione di progetti personalizzati, guidandolo nella selezione e nella combinazione dei componenti necessari alla realizzazione

delle soluzioni richieste dal cliente. Attraverso un'interfaccia interfaccia web[g]intuitiva e interattiva, sarà possibile comporre uno o più progetti, selezionando i componenti in base alle specifiche esigenze operative e ai vincoli tecnici del contesto applicativo. Particolare attenzione dovrà essere posta all'implementazione di logiche di vincolo che impediscano la selezione di componenti tra loro incompatibili, assicurando la coerenza e la validità tecnica delle configurazioni generate.

Esempio applicativo: si immagini un tecnico che, tramite la piattaforma, seleziona una valvola a sfera da utilizzare in un impianto di distribuzione di gas naturale. Il sistema, sulla base dei parametri inseriti (ad esempio pressione di esercizio, temperatura, natura del fluido), suggerisce automaticamente gli attuatore compatibili e i pannelli di controllo idonei, escludendo le combinazioni non conformi ai vincoli tecnici o normativi. L'utente può quindi aggiungere ulteriori componenti, visualizzare in tempo reale il costo complessivo della soluzione e generare la documentazione tecnica e commerciale necessaria per l'offerta al cliente.

Un ulteriore scenario potrebbe riguardare il settore Oil & Gas, dove la presenza di fluidi corrosivi o condizioni ambientali estreme richiede la selezione di materiali e certificazioni specifiche. In questo caso, il software guida l'utente nella scelta dei componenti idonei, tenendo conto delle normative ATEX e delle certificazioni richieste, e impedisce la selezione di accoppiamenti non conformi.

La piattaforma dovrà inoltre integrare strumenti avanzati per il calcolo automatico dei costi, la generazione della documentazione tecnica e commerciale e la produzione di output strutturati e pronti per la fase di offerta. Dal punto di vista architetturale, il sistema sarà progettato per garantire scalabilità, sicurezza, facilità di manutenzione e la possibilità di integrazione con i flussi di lavoro aziendali esistenti, così da adattarsi alle future evoluzioni organizzative e tecnologiche dell'azienda.

La proprietà e la gestione dell'applicativo saranno affidate direttamente al product owner[g], che potrà disporre di un sistema sempre aggiornato, sicuro e conforme alle esigenze operative e strategiche dell'organizzazione. In prospettiva, l'adozione di una soluzione di questo tipo consentirà di superare i limiti delle procedure manuali, introducendo un sistema intelligente e automatizzato che migliori in modo significativo l'efficienza, la qualità e la competitività aziendale nel settore dell'automazione industriale. Tale approccio, oltre a rispondere alle esigenze attuali, pone le basi per una futura evoluzione verso una gestione sempre più integrata e digitale dei processi aziendali.

Capitolo 2

Metodologie di sviluppo software

In questo capitolo vengono presentate le metodologie di sviluppo software adottate dal team di progetto. Dopo una breve introduzione ai principi dell'Agile, l'attenzione si concentra sulla metodologia Scrum, descrivendone struttura, ruoli e processi. Infine, viene illustrata l'applicazione concreta di Scrum nel contesto specifico del progetto, evidenziando adattamenti applicati al progetto.

2.1 Introduzione all'Agile

2.1.1 Introduzione

Negli ultimi decenni, il settore dell'ingegneria del software ha subito un'evoluzione radicale nelle metodologie di sviluppo, culminata nell'affermazione delle pratiche Agile. Queste metodologie sono nate come risposta alle limitazioni dei modelli tradizionali, come il modello a cascata[g], noti per la loro rigidità e la scarsa capacità di adattamento ai cambiamenti dei requisiti progettuali. Agile si fonda su principi di flessibilità, collaborazione continua e sviluppo iterativo, promuovendo un approccio orientato al valore per il cliente e alla capacità di rispondere rapidamente al cambiamento.

2.1.2 Origine del movimento Agile

Il movimento Agile prende ufficialmente vita nel febbraio 2001, durante un incontro presso il Lodge at Snowbird, nello Utah (USA). Qui, diciassette esperti di sviluppo software, tra cui figure di spicco come Kent Beck, Robert C. Martin e Martin Fowler, si riunirono per delineare una visione comune per le metodologie di sviluppo leggere.

Da questo incontro nacque il *Manifesto for Agile Software Development*, un documento fondante che stabilisce i valori e i principi cardine dell'approccio Agile.

La scelta del termine "Agile", preferito a "Lightweight" per evidenziare l'adattabilità e la rapidità di risposta ai cambiamenti, fu un punto di discussione tra i partecipanti. Ulteriori dettagli su questa curiosità sono riportati nell'approfondimento dedicato dell'Appendice A.

La pubblicazione del Manifesto, disponibile sul sito ufficiale[1], segnò un punto di svolta rispetto ai modelli documento-centrici e statici, promuovendo un nuovo paradigma basato su fiducia, collaborazione e centralità delle persone. Successivamente, gli

[1] 9.

autori del Manifesto fondarono l'Agile Alliance[g], un'organizzazione che ha contribuito a diffondere e far evolvere i principi agili nel mondo dello sviluppo software.

Più che una semplice metodologia, Agile rappresenta un vero e proprio cambiamento culturale, incentrato sulla valorizzazione del team, sulla trasparenza dei processi e sulla creazione di ambienti di lavoro che favoriscano l'espressione del potenziale individuale.

2.1.3 I valori principali

Il Manifesto Agile si articola attorno a quattro valori fondamentali, che rappresentano la base ideologica delle metodologie agili:

1. **Gli individui e le interazioni più che i processi e gli strumenti**: viene privilegiata la comunicazione diretta e la collaborazione tra i membri del team rispetto alla mera adozione di procedure o strumenti standardizzati.
2. **Il software funzionante più che la documentazione esaustiva**: l'obiettivo primario è la consegna di software funzionante, riducendo al minimo indispensabile la produzione di documentazione non essenziale.
3. **La collaborazione col cliente più che la negoziazione dei contratti**: si promuove un coinvolgimento attivo e continuo del cliente durante tutto il ciclo di sviluppo, superando la tradizionale separazione tra committente e fornitore.
4. **Rispondere al cambiamento più che seguire un piano**: la capacità di adattarsi rapidamente ai cambiamenti dei requisiti viene considerata più importante della rigorosa aderenza a un piano predefinito.

Questi valori non negano l'importanza degli elementi posti a destra, ma sottolineano la priorità di quelli a sinistra, in un'ottica di maggiore efficacia e adattabilità del processo di sviluppo.

2.1.4 Rilevanza per il contesto progettuale

Nel caso specifico di questo progetto, adottare una metodologia agile si è rivelato particolarmente efficace per diverse ragioni pratiche. Prima di tutto, il team di sviluppo è composto da sei persone: una dimensione che permette di lavorare in modo molto diretto e collaborativo, senza troppi passaggi intermedi o complicazioni nella comunicazione. Questo rende più semplice condividere idee, risolvere problemi rapidamente e prendere decisioni in tempi brevi.

Un altro punto di forza è il coinvolgimento costante del cliente. Durante tutto il ciclo di sviluppo, vengono organizzati incontri regolari in cui si presenta una demo delle funzionalità sviluppate nella settimana. Questi momenti sono fondamentali perché permettono al cliente di vedere concretamente i progressi fatti, di esprimere subito le proprie opinioni e di suggerire modifiche o miglioramenti. Insieme si discute cosa approfondire, cosa cambiare e come procedere, così da mantenere il prodotto sempre in linea con le reali esigenze di chi lo utilizzerà.

Inoltre, il progetto prevede una continua evoluzione: i requisiti possono essere rivisti e aggiornati anche in corso d'opera. Ad esempio, una volta soddisfatti i requisiti principali, è già previsto di estendere il sistema per integrare dati provenienti da un altro software aziendale. Questa flessibilità è uno degli aspetti che rendono l'approccio agile particolarmente adatto a situazioni in cui le necessità possono cambiare o emergere nuove opportunità durante lo sviluppo.

Infine, lavorare in modo agile aiuta a ridurre i rischi legati ai cambiamenti, sia nei requisiti che nelle tecnologie utilizzate. Grazie a un processo iterativo e incrementale, eventuali criticità vengono individuate e affrontate tempestivamente, evitando di accumulare problemi che potrebbero compromettere il risultato finale. In questo modo, il team può adattarsi rapidamente alle novità e garantire un prodotto di qualità, sempre allineato alle aspettative del cliente.

2.2 La metodologia Scrum

2.2.1 Introduzione

Come già approfondito nella sezione dedicata all'origine dell'Agile, la nascita delle metodologie agili ha rappresentato una svolta radicale nell'ingegneria del software, ponendo le basi per nuovi approcci più flessibili e collaborativi. All'interno di questo scenario, Scrum si distingue come uno dei framework agili più diffusi e strutturati, offrendo un insieme di regole, ruoli e pratiche che consentono di applicare in modo concreto i principi del Manifesto Agile.

Per ulteriori dettagli relativi all'origine del framework Scrum, alla sua etimologia e ai contributi dei principali autori, si invita il lettore a consultare l'Approfondimento in Appendice A.

È importante sottolineare che, mentre Agile rappresenta un insieme di valori e principi generali per lo sviluppo software, Scrum costituisce un framework specifico che traduce tali principi in un processo operativo ben definito, caratterizzato da ruoli chiari, artefatti e cerimonie ricorrenti. La popolarità di Scrum è dovuta proprio alla sua capacità di fornire una struttura semplice ma efficace, in grado di adattarsi a contesti progettuali diversi e di promuovere la collaborazione, la trasparenza e il miglioramento continuo.

2.2.2 Ruoli principali in Scrum

Scrum definisce tre ruoli fondamentali, ciascuno con responsabilità e compiti ben precisi, che garantiscono il corretto funzionamento del framework e la massima efficacia del processo di sviluppo.

Product Owner Il Product Owner rappresenta la voce del cliente e degli stakeholder all'interno del team. È responsabile della definizione e della gestione del Product Backlog, ovvero dell'elenco ordinato delle funzionalità, dei requisiti e delle attività necessarie per lo sviluppo del prodotto. Il Product Owner stabilisce le priorità, chiarisce i requisiti e si assicura che il team lavori sempre sugli elementi di maggior valore per il cliente. Inoltre, partecipa attivamente agli eventi Scrum, fornisce feedback continui e prende decisioni rapide per garantire l'allineamento tra le aspettative del committente e il prodotto realizzato.

Scrum Master Lo Scrum Master svolge il ruolo di facilitatore e coach del team Scrum. Ha il compito di rimuovere gli ostacoli che potrebbero impedire il progresso del team, assicurando che il processo Scrum venga seguito correttamente. Lo Scrum Master promuove la collaborazione, la trasparenza e il miglioramento continuo, aiutando il team a riflettere sulle proprie pratiche e a individuare soluzioni ai problemi che emergono durante lo sviluppo. È anche responsabile di proteggere il team da interferenze esterne e di favorire un ambiente di lavoro positivo e produttivo.

Development Team Il Development Team è composto dai professionisti che si occupano concretamente della realizzazione del prodotto. Si tratta di un gruppo auto-organizzato e multifunzionale, in cui ciascun membro contribuisce con le proprie competenze al raggiungimento degli obiettivi dello Sprint. Il team è responsabile della pianificazione, dello sviluppo, del testing e della documentazione delle funzionalità, lavorando in modo collaborativo per consegnare un incremento funzionante del prodotto al termine di ogni Sprint.

2.2.3 Artefatti Scrum

Scrum si basa su tre artefatti principali, che rappresentano gli strumenti fondamentali per la gestione e il monitoraggio del lavoro.

Product Backlog Il Product Backlog è una lista dinamica e ordinata di tutte le funzionalità, i miglioramenti, le correzioni e le attività necessarie per lo sviluppo del prodotto. È gestito dal Product Owner e viene costantemente aggiornato in base ai feedback del cliente, alle nuove esigenze emerse e alle priorità definite. Gli elementi del Product Backlog sono descritti in modo sufficientemente dettagliato da consentire al team di comprenderli e stimarli, ma possono essere ulteriormente raffinati durante il processo di sviluppo.

Sprint Backlog Lo Sprint Backlog è l'insieme degli elementi del Product Backlog selezionati dal team per essere realizzati durante uno specifico Sprint, insieme al piano dettagliato per il loro completamento. Il team si impegna a portare a termine tutti gli elementi dello Sprint Backlog entro la fine dello Sprint, monitorando quotidianamente lo stato di avanzamento e adattando il piano se necessario. Lo Sprint Backlog rappresenta uno strumento di trasparenza e di allineamento tra tutti i membri del team.

Increment L'Increment è il risultato tangibile del lavoro svolto durante uno Sprint. Si tratta di una versione funzionante del prodotto che include tutte le nuove funzionalità sviluppate e integrate con quelle già esistenti. Ogni Increment deve essere potenzialmente rilasciabile e soddisfare i criteri di qualità concordati dal team. L'Increment rappresenta il valore aggiunto consegnato al cliente al termine di ogni ciclo di sviluppo.

2.2.4 Eventi Scrum

Scrum prevede una serie di eventi strutturati, detti anche cerimonie, che scandiscono il ritmo del lavoro e favoriscono la trasparenza, la collaborazione e il miglioramento continuo.

Sprint Lo Sprint è il cuore pulsante di Scrum: si tratta di un intervallo di tempo fisso, generalmente compreso tra una e quattro settimane, durante il quale il team lavora per realizzare un incremento funzionante del prodotto. Ogni Sprint inizia con una pianificazione e termina con la revisione e la retrospettiva. Gli Sprint si susseguono senza interruzione, creando un ciclo continuo di sviluppo e miglioramento.

Sprint Planning Lo Sprint Planning è l'evento che dà inizio allo Sprint. Durante questa riunione, il team, insieme al Product Owner, seleziona gli elementi del Product Backlog da realizzare nello Sprint e definisce un piano di lavoro dettagliato. L'obiettivo è

chiarire cosa verrà consegnato e come il team intende raggiungere gli obiettivi prefissati. Lo Sprint Planning favorisce l'allineamento tra tutti i membri e consente di partire con una visione chiara e condivisa delle attività da svolgere.

Daily Scrum Il Daily Scrum è una breve riunione quotidiana durante la quale il team si confronta sui progressi compiuti, sugli ostacoli incontrati e sulle attività da svolgere nella giornata. Questo evento favorisce la trasparenza, la collaborazione e l'identificazione tempestiva di eventuali problemi, consentendo al team di adattare rapidamente il piano di lavoro. Generalmente dura circa 10 minuti, un tempo sufficiente per coprire gli argomenti essenziali. Curiosità: viene anche chiamato "stand up meeting", poiché nasce con l'idea di essere svolto in piedi, favorendo così un confronto rapido ed efficiente.

Sprint Review Al termine di ogni Sprint si svolge la Sprint Review, un incontro in cui il team presenta l'incremento realizzato agli stakeholder e raccoglie feedback sul lavoro svolto. Durante la Sprint Review vengono discusse le funzionalità implementate, le eventuali modifiche ai requisiti e le priorità per gli Sprint successivi. Questo evento rappresenta un momento fondamentale di confronto e di allineamento tra il team e il cliente.

Sprint Retrospective La Sprint Retrospective è l'evento conclusivo dello Sprint, dedicato alla riflessione interna del team sulle modalità di lavoro adottate. L'obiettivo è individuare punti di forza, aree di miglioramento e azioni concrete per ottimizzare i processi nei cicli successivi. La retrospettiva favorisce il miglioramento continuo e contribuisce a creare un clima di fiducia e collaborazione all'interno del team.

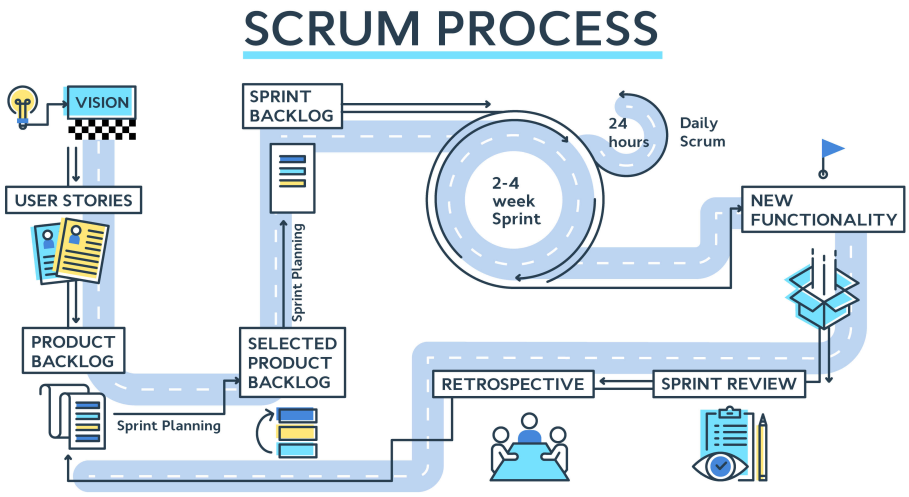

Figura 2.1: Schema del processo Scrum

Il diagramma illustra il flusso di lavoro di Scrum, che prende avvio dalla definizione della *Vision*[a] e dalla raccolta delle *User Stories*[b], organizzate nel *Product Backlog*. Attraverso la fase di *Sprint Planning*, il team seleziona gli elementi prioritari che confluiscono nello *Sprint Backlog*. Lo sviluppo avviene in *Sprint* di durata prefissata, scanditi da eventi ricorrenti come il *Daily Scrum*, la *Sprint Review* e la *Retrospective*. Al termine di ogni Sprint viene rilasciata nuova funzionalità, in un processo che garantisce trasparenza, adattamento continuo e valore incrementale per il cliente.

[a] La Vision rappresenta la finalità strategica e il quadro di riferimento generale del prodotto o progetto, orientando tutte le fasi dello sviluppo.

[b] Le User Stories sono brevi descrizioni, espresse dal punto di vista dell'utente finale, che specificano una funzionalità o un requisito desiderato.

2.3 Applicazione di Scrum nel progetto

2.3.1 Motivazioni della scelta di Scrum

La scelta di Scrum si fonda su motivazioni pratiche e teoriche. Il framework si adatta in modo ottimale a team di piccole dimensioni, come il nostro di sei persone, facilitando una comunicazione diretta ed efficace. Scrum è particolarmente indicato quando i requisiti sono soggetti a frequenti cambiamenti: grazie ai cicli brevi e iterativi degli Sprint, è possibile integrare rapidamente i feedback del cliente e adattare il prodotto in corso d'opera. Inoltre, la responsabilizzazione e l'auto organizzazione del team, caratteristiche centrali di Scrum, favoriscono la partecipazione attiva e la qualità del risultato finale. La struttura iterativa consente di individuare tempestivamente eventuali criticità, promuovendo il miglioramento continuo e rendendo Scrum ideale per progetti complessi e in continua evoluzione.

2.3.2 Adattamenti e personalizzazioni rispetto allo Scrum standard

Nel corso dello stage, la metodologia Scrum è stata adottata come modello organizzativo di riferimento, ma con alcuni adattamenti dettati dalle esigenze operative del team e dal contesto aziendale. In particolare, la durata degli sprint è stata flessibile, variando tra una e due settimane in base alla disponibilità dei membri, alcuni dei quali lavoravano part-time sul progetto. Gli incontri periodici, tipicamente previsti come Daily Scrum, sono stati riorganizzati con una cadenza più adatta alle esigenze del gruppo, tipicamente ogni due o tre giorni, così da garantire un coordinamento efficace nonostante la diversa presenza dei partecipanti.

Le Sprint Review sono state integrate con la presentazione di una demo delle funzionalità sviluppate, seguita da una discussione con il Product Owner per raccogliere feedback immediati e individuare criticità o nuove esigenze. La pianificazione dello sprint successivo veniva svolta in modo collaborativo, ridefinendo le priorità sulla base dei feedback e delle risorse disponibili. Inoltre, al termine degli incontri, il team svolgeva una retrospettiva interna per riflettere su processi, tecnologie e possibili miglioramenti organizzativi.

2.3.3 Strumenti e pratiche adottate

Per la gestione delle attività e la collaborazione, sono stati adottati strumenti digitali che hanno facilitato il lavoro sia in presenza sia da remoto. La comunicazione quotidiana è avvenuta principalmente tramite Microsoft Teams.

Per la gestione delle attività, del backlog e delle issues, è stata utilizzata la project board digitale di GitHub[g], che ha permesso di tracciare in modo trasparente lo stato di avanzamento delle task, assegnare compiti specifici e monitorare le scadenze. Oltre alle funzionalità base di VCS[g]offerte da GitHub, la board ha rappresentato un valido supporto per l'organizzazione delle attività secondo la logica Scrum.

La documentazione tecnica e i materiali di progetto sono stati condivisi tramite repository GitHub, garantendo accessibilità e aggiornamento costante delle informazioni. Le attività di sviluppo sono state svolte prevalentemente da remoto, con alcune occasioni di incontro in presenza dedicate al confronto diretto con il cliente sui requisiti e sulle funzionalità del prodotto.

Capitolo 3

Descrizione dello stage

Questo capitolo presenta il contesto e le modalità di svolgimento dello stage, illustrando la pianificazione delle attività, le scelte tecnologiche adottate e il contributo individuale fornito al progetto. Vengono inoltre descritte le principali dinamiche operative all'interno del team e le attività effettivamente realizzate durante il periodo di stage.

3.1 introduzione

La pianificazione delle attività relative allo stage è stata fortemente influenzata dall'adozione della metodologia agile Scrum, che ha rappresentato il modello organizzativo di riferimento per l'intero team di sviluppo. In particolare, il lavoro è stato suddiviso in sprint della durata di una o due settimane, in base alla disponibilità dei membri del team, alcuni dei quali lavoravano con frequenza ridotta (ad esempio tre giorni a settimana anziché cinque). Ogni sprint prevedeva la definizione di obiettivi specifici e la verifica periodica dello stato di avanzamento attraverso incontri.

Oltre alle riunioni di revisione a cadenza settimanale o bisettimanale, sono stati programmati incontri intermedi, con frequenza variabile (ogni due o tre giorni), finalizzati al monitoraggio costante delle attività e alla tempestiva risoluzione di eventuali criticità. L'interazione con il tutor aziendale e con il resto del team avveniva prevalentemente tramite la piattaforma Microsoft Teams, mentre le attività di sviluppo si svolgevano in modalità remota. Sono inoltre state previste alcune occasioni di incontro in presenza con il cliente, finalizzate al confronto diretto sui requisiti e sulle funzionalità del prodotto.

L'adozione di Scrum ha consentito di mantenere una pianificazione flessibile e adattiva, in cui le priorità e le attività da svolgere venivano ridefinite regolarmente sulla base del feedback del cliente e delle esigenze emergenti. Questo approccio iterativo e incrementale ha favorito una gestione dinamica del progetto, rendendo possibile un continuo allineamento tra le aspettative del committente e le soluzioni implementate dal team. È importante sottolineare che, sebbene la durata complessiva del progetto sia stata stimata in circa sei mesi, la mia partecipazione è stata limitata al periodo di stage, ovvero due mesi (circa otto settimane). La distribuzione delle attività durante il mio stage è stata quindi circoscritta a questo intervallo temporale, con una pianificazione soggetta a continui aggiustamenti, anche in considerazione della variabilità nella disponibilità oraria dei membri del team.

In sintesi, questa fase preliminare si è caratterizzata per un approccio collaborativo e adattivo, che ha permesso di affrontare in modo efficace le sfide poste dal progetto e

di garantire una costante attenzione alle esigenze del cliente, ponendo solide basi per le successive fasi di sviluppo.

3.2 Pianificazione Preventiva del Progetto

Prima dell'avvio dello stage, è stata definita una pianificazione preventiva delle attività, volta a strutturare un percorso formativo e operativo coerente con le esigenze progettuali e le metodologie del team. Tale pianificazione, concordata con il tutor aziendale e il responsabile accademico, ha suddiviso le attività su base settimanale secondo i principi dell'approccio Agile, prevedendo la possibilità di revisione periodica degli obiettivi in base all'evoluzione del progetto.

Nella fase iniziale era previsto un periodo di onboarding tecnico e funzionale[1], dedicato alla familiarizzazione con gli strumenti e le tecnologie adottate, tra cui MudBlazor[g], ASP.NET Core[g], MongoDB[g]e Docker[g]. Le settimane successive sarebbero state dedicate allo sviluppo delle interfacce front-end, delle logiche di back-end, all'integrazione con il database, alla gestione dei calcoli numerici e delle formule fisiche, oltre al deployment e alla documentazione tecnica, da redigere in modo continuativo.

La distribuzione delle ore di lavoro, pianificata prima dell'inizio dello stage, è la seguente:

- 60 ore per formazione, onboarding e studio delle tecnologie;
- 150 ore per attività di sviluppo (interfacce, logiche di configurazione, calcolo, integrazione e gestione degli ambienti);
- 70 ore per la partecipazione alle attività del team (sprint, meeting, code review, incontri con il cliente);
- 30 ore per la documentazione tecnica, la preparazione della demo finale e le attività di chiusura.

Complessivamente, la pianificazione preventiva prevedeva un totale di **310** ore distribuite su otto settimane, con attività organizzate in parallelo e una struttura flessibile, in linea con l'approccio iterativo dell'Agile. Lo stage ha avuto inizio il 28 aprile e si concluderà il 27 giugno.

[1] Onboarding: Processo di inserimento e formazione di un nuovo membro all'interno di un'organizzazione o progetto.

3.3 Pianificazione Effettiva del Progetto

La pianificazione effettiva delle attività è stata aggiornata settimanalmente in relazione al mio contributo individuale all'interno del progetto. Al termine di ogni settimana, ho condotto una retrospettiva personale per valutare i risultati conseguiti, individuare eventuali criticità e ridefinire le priorità operative per la settimana successiva. Di seguito viene presentata una sintesi dettagliata delle principali attività svolte, organizzate cronologicamente, che riflettono il percorso di stage.

3.3.1 Settimana 1

Introduzione

La prima settimana di stage è stata dedicata prevalentemente all'inserimento nel progetto e alla comprensione delle tecnologie e degli strumenti già adottati dal team di sviluppo. In questa fase iniziale, l'obiettivo principale era quello di ambientarsi rapidamente sia dal punto di vista tecnico sia nelle dinamiche di gruppo, al fine di poter fornire un contributo concreto nel minor tempo possibile.

Sviluppo

Sin dai primi giorni, mi sono focalizzato sull'apprendimento delle principali tecnologie utilizzate nel progetto, tra cui MudBlazor, .NET e MongoDB. Ho integrato lo studio teorico con esercitazioni pratiche, che mi hanno consentito di acquisire familiarità con l'architettura della piattaforma e con le modalità operative del team.

Un elemento rilevante per la mia produttività è stato l'utilizzo dell'ambiente di sviluppo integrato Rider[g], strumento che non avevo precedentemente approfondito. Ho potuto apprezzarne le numerose funzionalità, quali il refactoring intelligente[2], il supporto avanzato al debugging e la gestione integrata delle dipendenze, che si sono rivelate particolarmente efficaci nello sviluppo in ambienti .NET.

Tra le prime attività operative, mi sono occupato della progettazione e dello sviluppo di un modulo per l'inserimento delle valvole non Starline. Tale modulo richiedeva la gestione di diversi parametri tecnici (BTO[g], RTO[g], ETO[g], BTC[g], RTC[g], ETC[g]), fondamentali per la corretta caratterizzazione delle valvole. Ho inoltre implementato una funzione per la conversione delle unità di misura, al fine di uniformare i dati tecnici inseriti dagli utenti (ad esempio, la conversione di Lb-Ft, Lb-In e Kgf-m in Nm). Per garantire la correttezza delle conversioni, ho redatto alcuni test di unità che mi hanno permesso di individuare e correggere tempestivamente eventuali errori.

Dal punto di vista organizzativo, la comunicazione con il team è stata agevolata dall'utilizzo di Microsoft Teams, che ha facilitato la risoluzione di dubbi e la ricezione di feedback tempestivi sulle attività svolte. La chiara suddivisione dei compiti all'interno del gruppo ha inoltre contribuito a rendere più agevole l'inserimento e la comprensione delle aree di intervento.

Conclusione

In sintesi, la prima settimana si è rivelata fondamentale per acquisire familiarità con il progetto e con gli strumenti di lavoro. L'apprendimento approfondito di Rider come IDE[g]ha rappresentato un investimento significativo, consentendomi di operare in modo

[2]Processo di ristrutturazione del codice sorgente senza alterarne il comportamento esterno, per migliorarne la leggibilità e la manutenibilità.

più efficiente e organizzato. Le attività svolte hanno favorito un rapido coinvolgimento nello sviluppo e una migliore comprensione sia degli aspetti tecnici sia delle dinamiche collaborative del progetto. L'ambiente di lavoro si è dimostrato stimolante e ben strutturato, ponendo solide basi per un percorso di crescita personale e professionale.

3.3.2 Settimana 2

Introduzione

La seconda settimana è stata dedicata principalmente al consolidamento delle scelte tecniche e all'integrazione di strumenti fondamentali per la gestione dei dati e dei servizi backend. Superata la fase iniziale di ambientamento, l'attenzione si è focalizzata sulla definizione delle principali strutture architetturali e sull'adeguamento delle soluzioni alle specifiche esigenze del cliente, con particolare riferimento alla gestione di componenti caratterizzati da proprietà tecniche variabili.

Sviluppo

Una delle principali sfide affrontate in questa fase ha riguardato la necessità di gestire entità dotate di proprietà non fisse, variabili in funzione della tipologia di componente. Tale esigenza ha motivato la scelta di MongoDB come database principale: la sua natura NoSQL[g]si è dimostrata particolarmente adatta alla gestione di dati dinamici, consentendo di evitare la rigidità tipica dei database relazionali e semplificando la gestione di proprietà arbitrarie.

Contestualmente, ho iniziato a utilizzare strumenti quali Docker e SwaggerUI[g]. Docker si è rivelato estremamente utile per la creazione di ambienti di sviluppo replicabili e facilmente distribuibili, riducendo le problematiche derivanti dalle differenze tra ambienti locali e server di produzione. SwaggerUI, invece, ha facilitato la documentazione delle API e ha reso più agevole il testing delle chiamate backend, risultando particolarmente vantaggioso. Sebbene SwaggerUI sia stato scelto inizialmente per la sua immediatezza, già in questa fase si è valutata la possibilità di adottare strumenti più avanzati in futuro, in funzione dell'evoluzione delle esigenze progettuali.

Un ulteriore passo significativo è stato rappresentato dall'adozione di GitHub per la gestione delle issues, che ha semplificato il tracciamento delle attività, la segnalazione dei bug e il monitoraggio dello stato di avanzamento delle diverse funzionalità. La distinzione tra sviluppo di nuove feature e attività di refactoring è stata gestita con attenzione, applicando i principi SOLID[g]e le best practice di clean code, al fine di garantire un codice leggibile e facilmente manutenibile.

Nel corso della settimana, mi sono inoltre occupato della creazione di alcune strutture dati fondamentali, quali alcuni [g] e i moduli per la gestione dei brand. Ho iniziato a implementare le prime logiche di serializzazione e a effettuare piccoli refactoring volti a migliorare la chiarezza del codice. Parallelamente, il team ha introdotto strumenti di monitoraggio come OpenTelemetry e alcune dashboard di controllo, che si sono rivelate utili per individuare tempestivamente eventuali anomalie o colli di bottiglia.

Conclusione

In sintesi, la seconda settimana ha rappresentato un importante avanzamento sia dal punto di vista tecnico sia organizzativo. L'introduzione di Docker e SwaggerUI ha contribuito a standardizzare e rendere più affidabili i processi di sviluppo e testing, mentre la gestione sistematica delle issues tramite GitHub ha migliorato la comunica-

zione e la tracciabilità delle attività. Il costante confronto con le richieste del cliente e l'attenzione al refactoring hanno rafforzato la qualità del codice e la coesione del team. Nel complesso, il lavoro svolto ha posto basi solide per le fasi successive, rendendo il sistema più flessibile e predisposto all'integrazione di nuove funzionalità.

3.3.3 Settimana 3

Introduzione

La terza settimana è stata caratterizzata da un intenso lavoro volto al consolidamento dell'architettura e al miglioramento della qualità del codice. Dopo aver definito le principali strutture dati e le logiche di base nelle settimane precedenti, l'attenzione si è concentrata sull'ottimizzazione delle funzionalità già implementate e sull'introduzione di strumenti avanzati per la gestione delle entità e il monitoraggio del sistema, aspetti fondamentali per garantire coerenza e affidabilità progettuale.

Sviluppo

Gran parte delle attività settimanali ha riguardato il refactoring del codice: sono stati rinominati progetti, riorganizzate le cartelle, eliminate dipendenze obsolete e ottimizzati i DTO[g], con l'obiettivo di rendere il codice più chiaro, modulare e predisposto a future estensioni. Tali interventi hanno contribuito a migliorare la leggibilità e la manutenibilità del progetto, facilitando sia il lavoro quotidiano sia l'integrazione di nuove funzionalità.

Un'importante novità è stata l'introduzione della dashboard Aspire e del modulo OpenTelemetry[g], che hanno incrementato significativamente la capacità di monitoraggio e tracciamento delle attività di sistema. Questi strumenti si sono rivelati particolarmente efficaci nell'individuare tempestivamente eventuali criticità e nel fornire una visione più completa dello stato dell'applicazione.

Durante l'integrazione tra frontend e backend sono emerse alcune problematiche, soprattutto nella gestione delle operazioni CRUD[g]e nell'associazione degli attributi tra le varie entità. In particolare, si sono verificati diversi errori di tipo 500 Internal Server Error. La gestione sistematica delle issues tramite GitHub si è dimostrata molto efficace: ogni problematica è stata tracciata, discussa e risolta in tempi rapidi, favorendo la condivisione delle soluzioni all'interno del team.

Un cambiamento significativo introdotto in questa settimana è stato il passaggio da Swagger a Scalar[g]per la documentazione delle API. Scalar si è rivelato più adatto alle esigenze del progetto, offrendo una documentazione più chiara, personalizzabile e meglio integrata con lo stack tecnologico adottato. Sebbene il cambiamento abbia richiesto un breve periodo di adattamento, i benefici in termini di usabilità e chiarezza sono stati immediati per l'intero team.

Infine, è stato introdotto l'utilizzo dei pattern disposable sui componenti MudBlazor, al fine di gestire in modo più efficiente le risorse e prevenire problemi di memory leak[3], incrementando la stabilità dell'applicazione nel tempo.

Conclusione

In conclusione, la terza settimana ha rappresentato un netto avanzamento in termini di solidità e qualità del sistema. Il refactoring e l'ottimizzazione del codice hanno

[3]Fenomeno per cui la memoria allocata da un programma non viene più liberata, causando un consumo crescente di risorse.

migliorato la manutenibilità e la chiarezza dell'architettura, mentre l'introduzione di strumenti come Scalar, Aspire e OpenTelemetry ha rafforzato il controllo e il monitoraggio delle attività. La gestione efficace delle issues e l'attenzione alle problematiche di integrazione hanno permesso di risolvere rapidamente i bug più critici, consolidando la coesione del team e la resilienza del sistema. Queste attività hanno posto basi solide per affrontare con maggiore sicurezza le sfide delle settimane successive.

3.3.4 Settimana 4

Introduzione

La quarta settimana ha rappresentato un punto di svolta per il progetto, sia dal punto di vista dell'esperienza utente sia per quanto concerne la flessibilità nella gestione dei dati. L'attenzione si è focalizzata sull' arricchimento del frontend e sull'implementazione di soluzioni volte a consentire una gestione più dinamica e personalizzata delle proprietà dei componenti, con l'obiettivo di rendere il sistema maggiormente scalabile e predisposto ad adattarsi a nuove esigenze operative.

Sviluppo

In questa fase è emersa con particolare evidenza la necessità di gestire proprietà custom per i componenti. In concreto, i dati trattati risultano spesso molto variabili e non sempre prevedibili a priori: la possibilità di aggiungere proprietà personalizzate ha conferito al sistema una notevole flessibilità, rendendolo più aderente alle reali esigenze operative. L'integrazione di tali proprietà nei filtri e nei bindings di progetto ha migliorato sensibilmente la precisione delle ricerche e la gestione delle informazioni, rendendo le funzionalità di ricerca e manipolazione dei dati più efficaci e immediate.

Dal punto di vista tecnico, una delle principali sfide affrontate ha riguardato la serializzazione con MongoDB, in particolare per quanto concerne le proprietà di tipo intervallo. Per ovviare a tali problematiche, sono stati introdotti DTO e serializer specifici, che hanno garantito la corretta persistenza e il recupero dei dati, prevenendo perdite di informazioni o incoerenze tra frontend e backend.

Il refactoring continuo e la gestione dei merge tra i diversi branch sono stati agevolati dall'utilizzo di JetBrains Rider, che si è confermato uno strumento estremamente valido sia per la pulizia del codice sia per la risoluzione dei conflitti e l'ottimizzazione della struttura progettuale.

Sebbene nel frontend non si parli propriamente di "moduli", l'adozione di MudBlazor e la creazione di componenti personalizzati hanno contribuito a rendere il codice più leggibile e riutilizzabile. La costruzione di componenti riutilizzabili è ormai una prassi consolidata, che facilita la manutenzione e l'estensione futura del sistema. Per quanto riguarda l'interfaccia utente, sono state introdotte nuove funzionalità di autocomplete[4] per la selezione dei brand, sono state ottimizzate le query e migliorata la visualizzazione dei dati. Le griglie per la gestione di materiali, componenti e tipologie di componenti sono state completate, rendendo le operazioni di inserimento, modifica e ricerca più semplici e immediate.

[4]Autocomplete: Funzionalità dell'interfaccia utente che suggerisce automaticamente valori possibili durante la digitazione.

Conclusione

La quarta settimana si è conclusa con un netto miglioramento della qualità e della modularità del sistema. L'introduzione di proprietà dinamiche e di filtri avanzati ha incrementato la flessibilità nella gestione delle entità, mentre la risoluzione delle problematiche di serializzazione e il refactoring continuo hanno rafforzato la robustezza e l'affidabilità del software. L'arricchimento dell'interfaccia utente e la gestione avanzata delle griglie hanno reso il sistema più usabile e intuitivo.

3.3.5 Settimana 5

Introduzione

Durante la quinta settimana il progetto è entrato in una fase di affinamento delle funzionalità legate alla gestione dei dati, con particolare attenzione agli strumenti di ricerca, filtro e ordinamento. Questi aspetti si sono rivelati fondamentali per migliorare l'accessibilità e la rapidità di consultazione delle informazioni, elementi chiave per la crescita e l'usabilità della piattaforma. Parallelamente, il team ha lavorato sull'ottimizzazione della gestione delle proprietà dinamiche, puntando su soluzioni semplici, riusabili e robuste dal punto di vista architetturale.

Sviluppo

Le nuove funzionalità di filtro e ordinamento sono state implementate sfruttando il componente datagrid di MudBlazor, che si è dimostrato particolarmente efficace nell'integrare campi di filtro e sorting direttamente nell'interfaccia utente. Tuttavia, la logica di ordinamento e filtraggio è stata delegata al backend: il frontend invia le istruzioni e il backend le applica sui dati, garantendo così prestazioni ottimali e risultati coerenti anche in presenza di grandi volumi informativi.

Per la gestione delle proprietà dinamiche tra componenti e project filters, è stata adottata una struttura a dictionary, collegando le chiavi di entrambi i contesti al nome del parametro. Questo approccio ha semplificato la logica di associazione e manipolazione delle proprietà, aumentando la flessibilità del sistema e facilitando il riutilizzo del codice.

Durante la settimana sono emerse diverse problematiche legate alla gestione delle eccezioni, in particolare nelle fasi di salvataggio e aggiornamento delle entità. Sono state quindi riviste le condizioni di abilitazione dei pulsanti di salvataggio nelle dialog di gestione componenti, al fine di prevenire errori di inserimento e migliorare l'esperienza utente. La tempestiva risoluzione dei bug e il continuo refactoring hanno contribuito a mantenere elevata la qualità del software, con particolare attenzione alla pulizia del codice e alla gestione delle eccezioni.

Un ulteriore passo avanti è stato rappresentato dall'introduzione di filtri specifici per le proprietà numeriche dei componenti, che consentono ricerche personalizzate anche sulle proprietà custom. Tale logica di filtro e ordinamento è stata estesa anche ai bindings e alle relative proprietà, garantendo coerenza e uniformità su tutte le principali entità del sistema.

Conclusione

La quinta settimana ha segnato un deciso passo avanti nella gestione delle informazioni e nell'usabilità della piattaforma. L'ottimizzazione dei filtri e delle funzionalità di ordinamento ha reso il sistema più efficiente e intuitivo, facilitando la ricerca e la

selezione dei dati. L'attenzione ai dettagli, la risoluzione rapida dei bug e il refactoring continuo hanno contribuito a mantenere elevata la qualità del software. Le discussioni tecniche interne hanno favorito la standardizzazione e la riusabilità delle soluzioni adottate, ponendo basi solide per una gestione delle proprietà dinamiche sempre più efficace e scalabile nelle future evoluzioni del progetto.

3.3.6 Settimana 6

Introduzione

Durante la sesta settimana il progetto ha raggiunto una fase di maturazione avanzata, con un focus sull'efficienza operativa e sul controllo della qualità dei dati. L'attenzione si è concentrata sull'implementazione di strumenti volti a facilitare la gestione massiva delle informazioni e sulla definizione di procedure di validazione rigorose, rispondendo direttamente alle esigenze operative espresse dagli utenti e dal committente.

Sviluppo

Una delle principali novità introdotte in questa settimana è stata la funzionalità di duplicazione di progetti e componenti. Tale esigenza, emersa dalle richieste del cliente, ha permesso di creare rapidamente entità simili senza doverle ricostruire, mantenendo vincoli e caratteristiche ereditate. Questo ha notevolmente velocizzato la creazione di strutture complesse e incrementato la produttività degli utenti.

Parallelamente, è stata definita e implementata la relazione tra offerte e progetti: i progetti vengono utilizzati come "stampini" per la preparazione di nuove offerte, facilitando la personalizzazione delle proposte commerciali in base ai vincoli specifici di ciascun cliente. L'ereditarietà dei filtri e la possibilità di modificarli manualmente garantiscono la necessaria flessibilità e coerenza nella gestione delle offerte.

Per quanto riguarda la qualità dei dati, sono stati introdotti controlli stringenti per la validazione e la prevenzione dei record duplicati. In particolare, sono stati implementati controlli lato backend che impediscono l'inserimento di entità già esistenti, contribuendo a mantenere elevata l'affidabilità e la coerenza delle informazioni gestite dal sistema.

Nel corso della settimana sono anche iniziati i lavori per strutturare le offerte e le linee di valutazione, con una prima discussione sulle modalità di importazione dati (Excel rispetto a una grid interattiva), che sarà approfondita nelle fasi successive. Sono stati apportati diversi miglioramenti nella gestione delle note, nella logica di salvataggio (in particolare per array e deep copy[5] degli oggetti progetto) e nella validazione dei form. Inoltre, sono stati risolti alcuni bug sul frontend, soprattutto relativi alle funzionalità di autocomplete e alla gestione dei menu a tendina.

Un ulteriore avanzamento ha riguardato la gestione dei materiali: è stata aggiunta la colonna "TYPE" nella tabella materiali e creata la tabella "MATERIALS TYPE", permettendo una categorizzazione più precisa e strutturata dei materiali stessi.

Conclusione

La sesta settimana ha segnato un importante salto di qualità nella gestione dei dati e nell'organizzazione dei processi di business. L'introduzione della duplicazione di progetti e componenti, insieme all'ottimizzazione delle logiche di validazione e alla strutturazione dei tipi di materiale, ha incrementato la produttività, la flessibilità e

[5] Deep copy: Copia completa di un oggetto, comprensiva di tutte le sue dipendenze e riferimenti, distinta dalla shallow copy che copia solo i riferimenti.

l'affidabilità della piattaforma. La definizione delle modalità di importazione delle linee di offerta rappresenta un passo strategico verso una piena operatività e scalabilità del sistema. Nel complesso, il lavoro svolto testimonia la maturità raggiunta dal progetto e la sua capacità di adattarsi alle reali esigenze operative e di mercato.

3.3.7 Settimana 7

Introduzione

La settima settimana ha rappresentato un momento di significativa innovazione, in particolare per quanto riguarda l'interfaccia utente e la gestione dei dati. Il focus principale si è concentrato sull'implementazione di strumenti avanzati per l'autocompletamento e sull'ottimizzazione dei processi di inserimento e gestione delle informazioni, con l'obiettivo di incrementare la rapidità operativa e ridurre il rischio di errori. Parallelamente, sono state rafforzate le logiche di modularità e riusabilità, aspetti fondamentali per garantire la futura espandibilità e la manutenibilità della piattaforma.

Sviluppo

L'esigenza di sviluppare componenti personalizzati per l'autocomplete è nata dalla necessità di prevenire errori di scrittura che avrebbero potuto compromettere la qualità dei dati e la funzionalità dell'applicazione. La definizione preventiva delle proprietà, unitamente all'utilizzo di sistemi di autocompletamento durante la selezione, ha assicurato coerenza e precisione nell'inserimento dei dati. I nuovi componenti di autocomplete per materiali, brand e tipologie di componenti sono stati progettati per effettuare ricerche dirette su database, favorendo il riutilizzo e la coerenza delle informazioni gestite.

Contestualmente, è stata ottimizzata la gestione dei materiali tramite l'introduzione e la gestione della colonna "TYPE" nella tabella materiali, con selezione facilitata da menu a tendina alimentato dalla nuova tabella "MATERIALS TYPE". Questo intervento ha migliorato la categorizzazione e la precisione nella gestione delle entità materiali, rendendo più semplice e veloce la selezione dei materiali corretti nelle diverse fasi operative.

Un ulteriore aspetto rilevante ha riguardato la riorganizzazione delle dipendenze e delle cartelle di progetto, con particolare attenzione alla separazione logica dei moduli di lookup. Tale riorganizzazione ha contribuito a una maggiore chiarezza architetturale e a una più agevole manutenzione del codice, facilitando anche il processo di onboarding per i nuovi membri del team.

Il refactoring della gestione dei bindings nei progetti ha portato alla sostituzione dei bindings con le properties direttamente all'interno del dialog di progetto, rispondendo alle esigenze di maggiore immediatezza e semplicità espresse dal cliente. Questa scelta ha migliorato la coerenza della piattaforma, eliminando passaggi ridondanti e semplificando la gestione dei vincoli.

Per ottimizzare la serializzazione dei materiali e delle proprietà, sono stati adottati extension method di C#, che consentono di aggiungere metodi di istanza a classi di cui non si è proprietari, come quelle delle librerie esterne. Questa strategia ha permesso di gestire in modo più flessibile e modulare i processi di serializzazione, con particolare attenzione ai casi d'uso nei bindings e nei progetti.

le nuove implementazioni sono state valutare positivamente, grazie alla maggiore rapidità e precisione nell'inserimento delle informazioni e alla riduzione degli errori. Gli utenti hanno riscontrato un'interfaccia più intuitiva e una gestione dei dati più fluida.

Conclusione

La settima settimana ha evidenziato un deciso avanzamento nella standardizzazione e nell'efficienza dei processi di inserimento dati. L'introduzione di componenti di autocompletamento avanzati ha reso l'interfaccia utente più modulare e riusabile, migliorando sensibilmente la rapidità e la precisione nell'inserimento delle informazioni. La strutturazione dei tipi di materiale e la gestione diretta delle properties nei progetti hanno favorito una maggiore coerenza e flessibilità nella gestione dei dati. Il refactoring continuo e l'ottimizzazione della serializzazione hanno contribuito a rafforzare la qualità architetturale della piattaforma, ponendo solide basi per l'implementazione di ulteriori funzionalità avanzate nelle fasi successive. Nel complesso, il lavoro svolto in questa settimana ha migliorato sia l'esperienza utente sia la qualità complessiva dei dati gestiti dal sistema.

3.3.8 Settimana 8

Introduzione

L'ottava settimana ha rappresentato un punto di svolta per il consolidamento delle logiche di calcolo e per il miglioramento dell'affidabilità della piattaforma. Il team si è concentrato prevalentemente sul refactoring della pagina di valutazione, con l'obiettivo di incrementare la precisione nelle conversioni delle unità di misura e di ottimizzare la gestione delle attività di sviluppo. Parallelamente, sono state affrontate e risolte alcune problematiche residue, ponendo le basi per l'introduzione di nuove funzionalità in linea con le esigenze in evoluzione del sistema.

Sviluppo

Il refactoring della pagina di valutazione è stato guidato dall'esigenza di ottenere calcoli più precisi, in particolare per quanto concerne l'arrotondamento dei valori di torsione e la gestione dei tipi numerici. In passato, l'utilizzo del tipo `double` aveva evidenziato limiti in termini di accuratezza, soprattutto nelle operazioni che richiedevano una gestione fine dei decimali. La migrazione verso il tipo `decimal` ha consentito di ridurre gli errori di approssimazione e di rappresentare i valori numerici in modo più fedele, incrementando l'affidabilità delle valutazioni tecniche.

La revisione delle logiche di conversione delle unità di misura ha avuto un impatto concreto sulla precisione dei calcoli. L'adozione di procedure più rigorose ha eliminato molte delle discrepanze riscontrate in precedenza, assicurando coerenza tra i dati inseriti e quelli elaborati dal sistema. Questo ha permesso di offrire agli utenti risultati più affidabili e di ridurre il rischio di errori dovuti a conversioni imprecise.

Nel corso della settimana sono state inoltre risolte alcune problematiche residue, tra cui bug relativi all'autocomplete dei brand e l'aggiornamento delle dipendenze di progetto. Questi interventi hanno contribuito a migliorare la stabilità e la fluidità dell'esperienza utente, mantenendo il progetto aggiornato agli standard tecnologici attuali.

Per migliorare la chiarezza e la tracciabilità delle attività di sviluppo, è stato rafforzato l'utilizzo dei "todo" nel codice. Grazie agli strumenti di sviluppo come Rider, tali promemoria risultano facilmente individuabili e consentono di monitorare le attività ancora da completare, favorendo una pianificazione più efficace e una maggiore trasparenza nel workflow.

Guardando alle fasi future, sono già state individuate nuove direttrici di sviluppo, tra cui l'integrazione degli step di creazione dell'offerta. Oltre alla valutazione dei valori delle valvole, si prevede di aggiungere funzionalità per la selezione del progetto di riferimento e dei componenti da includere nell'offerta, così da rendere la configurazione delle proposte tecniche ancora più flessibile e personalizzabile.

Conclusione

L'ottava settimana ha segnato un passo avanti significativo nella qualità e nell'affidabilità della piattaforma. Il refactoring delle logiche di valutazione e la revisione delle procedure di conversione hanno incrementato la precisione dei calcoli e ridotto le criticità residue, mentre una migliore organizzazione delle attività ha reso più efficiente il lavoro del team. Le soluzioni adottate testimoniano l'attenzione costante verso l'innovazione e la qualità del software, ponendo basi solide per l'implementazione di nuove funzionalità nelle prossime fasi. Nel complesso, il lavoro svolto ha rafforzato la coerenza architetturale del sistema e ha contribuito a migliorare in modo tangibile l'esperienza utente.

3.4 Conclusione della Pianificazione Effettiva

La pianificazione effettiva delle attività di stage si è conclusa in modo conforme a quanto preventivato, con il completamento delle otto settimane e il raggiungimento del monte ore stabilito in fase iniziale, pari a **310** ore complessive. L'articolazione settimanale delle attività ha consentito di rispettare la distribuzione temporale e qualitativa degli obiettivi formativi.

È importante sottolineare che, pur avendo concluso il percorso di stage secondo quanto programmato, il progetto oggetto di analisi proseguirà il proprio sviluppo per un periodo stimato di ulteriori tre o quattro mesi. Tale prosecuzione sarà finalizzata al completamento delle funzionalità e dei requisiti individuati nell'analisi iniziale, nonché all'ulteriore affinamento del sistema, in vista del raggiungimento di una versione definitiva e pienamente operativa.

Capitolo 4

Analisi dei requisiti

In questo capitolo vengono presentati i casi d'uso che descrivono le principali funzionalità del sistema e le modalità di interazione tra gli utenti e l'applicazione. Attraverso diagrammi e descrizioni dettagliate, si illustrano i processi fondamentali che guidano la configurazione automatizzata dei pannelli di automazione. La suddivisione in macro-aree consente di evidenziare la struttura modulare della soluzione e di facilitare la comprensione delle attività svolte dagli attori principali

4.1 Casi d'uso

Per l'analisi funzionale del sistema sono stati realizzati appositi diagrammi dei casi d'uso (*Use Case Diagram*), secondo lo standard UML[g]. Questi diagrammi rappresentano graficamente le principali funzionalità offerte dall'applicazione web per la configurazione automatizzata dei pannelli di automazione, evidenziando le modalità di interazione tra gli attori (tecnici aziendali e amministratori) e il sistema. La suddivisione dei casi d'uso in tre macro-aree riflette la struttura modulare della soluzione proposta, volta a semplificare e ottimizzare le attività dell'utente finale.

4.1.1 Gestione Progetti

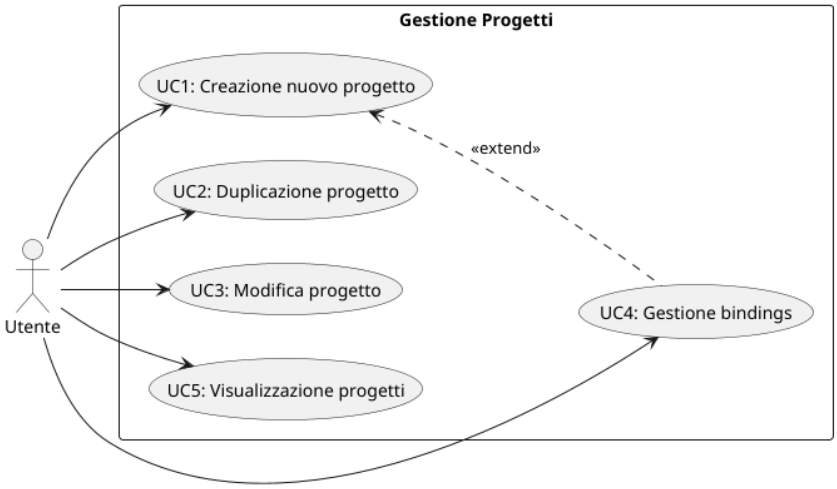

Figura 4.1: Diagramma dei casi d'uso: Gestione Progetti

La figura 4.1 illustra le funzionalità relative alla gestione dei progetti, che costituiscono il nucleo operativo per la definizione delle commesse aziendali. L'utente può creare, duplicare e modificare progetti, ciascuno caratterizzato da specifiche tecniche, note interne e vincoli (bindings) personalizzabili. La possibilità di gestire i bindings anche successivamente alla creazione del progetto garantisce flessibilità nella definizione dei requisiti tecnici, permettendo di adattare rapidamente le specifiche alle esigenze del cliente o alle evoluzioni del progetto stesso. La visualizzazione dell'elenco dei progetti consente inoltre una rapida consultazione e selezione delle commesse esistenti.

4.1.2 Gestione Offerte

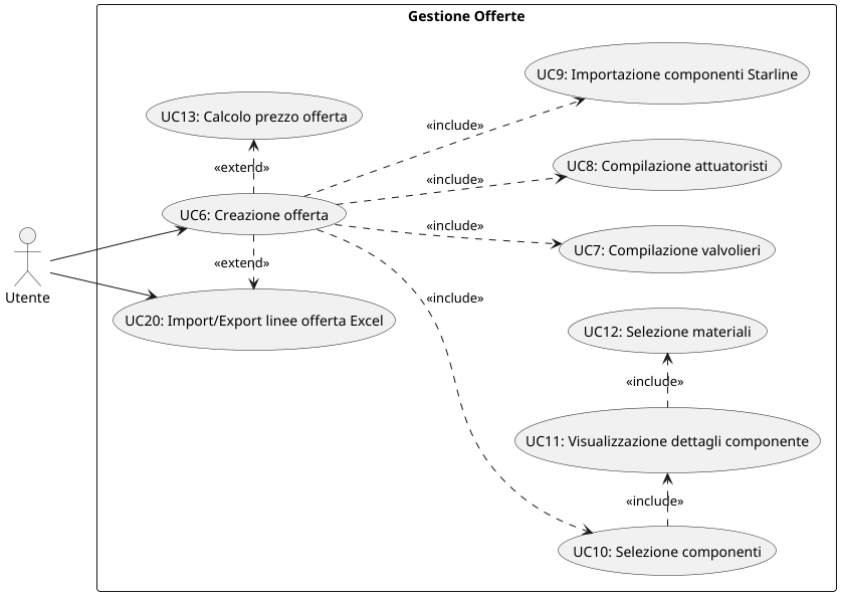

Figura 4.2: Diagramma dei casi d'uso: Gestione Offerte

Come evidenziato in figura 4.2, la gestione delle offerte rappresenta il processo centrale per la configurazione e la valutazione economica dei pannelli di controllo. L'utente può avviare la creazione di un'offerta associando un progetto di riferimento e selezionando i componenti conformi ai vincoli tecnici predefiniti. Il sistema prevede la compilazione di form specifici per l'inserimento dei dati relativi a valvole o attuatori non standard, nonché l'importazione automatica dei dati tecnici dei componenti Starline tramite API[g]. La selezione dei componenti e dei materiali avviene in modo guidato, con la possibilità di visualizzare i dettagli tecnici di ciascun elemento. Il calcolo del prezzo complessivo dell'offerta è gestito automaticamente dal sistema, mentre le funzionalità di importazione ed esportazione delle linee in formato Excel agevolano la gestione massiva delle offerte e la produzione di preventivi dettagliati.

4.1.3 Gestione Anagrafiche

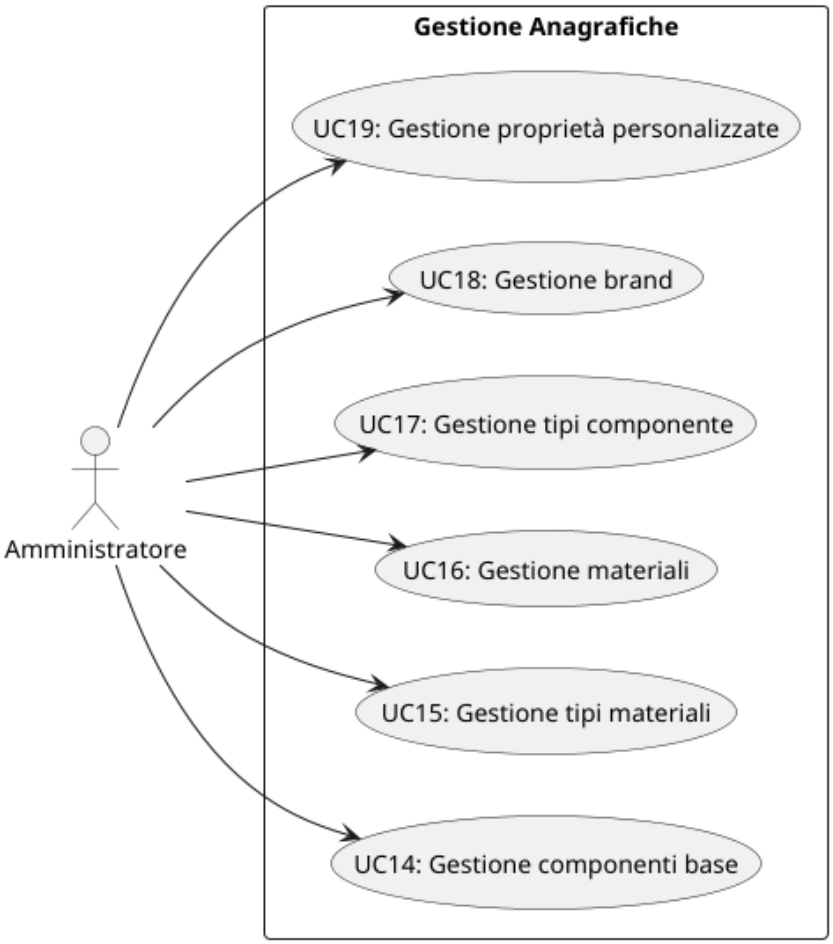

Figura 4.3: Diagramma dei casi d'uso: Gestione Anagrafiche

La figura 4.3 presenta le funzionalità riservate all'amministratore per la gestione delle anagrafiche di sistema. In questa sezione è possibile definire, aggiornare e mantenere l'elenco dei componenti base, dei materiali, dei brand, delle tipologie di componente e delle proprietà tecniche personalizzate. Tali funzionalità costituiscono il fondamento informativo su cui si basano la configurazione dei progetti e la generazione delle offerte, garantendo coerenza, integrità e aggiornamento costante dei dati tecnici e commerciali disponibili all'interno dell'applicazione.

4.1.4 Elenco dei Casi d'uso

UC1: Creazione di un nuovo progetto

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema e intende avviare la definizione di una nuova commessa aziendale.

Descrizione: L'utente crea un nuovo progetto inserendo il nome, i riferimenti alle specifiche tecniche, una lista di note cliente, una lista di note interne e un insieme di vincoli tecnici (bindings). I bindings possono riferirsi sia a proprietà standard sia a proprietà custom dei componenti, come ad esempio la selezione di un solo brand o il controllo di intervalli di valori per proprietà tecniche. Se la creazione va a buon fine, il sistema assegna automaticamente un codice identificativo a 4 cifre sequenziale e salva il progetto nel database.

Postcondizioni: Un nuovo progetto, completo di tutte le informazioni richieste, è stato creato e salvato nel database con identificatore univoco.

UC2: Duplicazione di un progetto esistente

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema e seleziona un progetto esistente da duplicare.

Descrizione: L'utente duplica un progetto già presente nel sistema. Tutte le informazioni del progetto originale vengono copiate, ma viene generato un nuovo codice identificativo sequenziale a 4 cifre per il progetto duplicato.

Postcondizioni: Un nuovo progetto, identico nei dati a quello selezionato ma con codice identificativo diverso, è stato creato e salvato nel database.

UC3: Modifica delle informazioni di progetto

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema e seleziona un progetto esistente.

Descrizione: L'utente aggiorna le informazioni di un progetto esistente, modificando nome, riferimenti, note cliente, note interne o bindings. Il progetto viene aggiornato mantenendo il codice identificativo originario.

Postcondizioni: Le informazioni del progetto sono state aggiornate e salvate nel database.

UC4: Gestione bindings di progetto

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema e sta creando o modificando un progetto.

Descrizione: L'utente seleziona, aggiunge, modifica o elimina i bindings associati a un progetto. I bindings possono riferirsi a qualsiasi proprietà dei componenti, sia standard

sia custom.

Postcondizioni: L'insieme dei bindings del progetto è stato aggiornato secondo le modifiche apportate dall'utente.

UC5: Visualizzazione dell'elenco dei progetti

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema.

Descrizione: L'utente consulta la griglia dei progetti disponibili, con possibilità di ricerca e filtro per facilitare l'individuazione dei progetti di interesse.

Postcondizioni: L'utente ha visualizzato l'elenco dei progetti e può selezionarne uno per ulteriori operazioni.

UC6: Creazione di un'offerta (Valuation)

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema e intende avviare la costruzione di un'offerta per un pannello di controllo.

Descrizione: L'utente seleziona, se necessario, il form per l'inserimento dei dati di valvole o attuatori non Starline, oppure seleziona componenti Starline o salta i form se non necessari. Successivamente, seleziona un progetto da associare all'offerta, che vincola i componenti proposti. L'utente seleziona i tipi e le quantità di componenti compatibili da includere nel pannello di controllo. Il sistema genera un preventivo di costi e materiali per la costruzione del pannello.

Postcondizioni: È stata generata un'offerta dettagliata, comprensiva di componenti selezionati, quantità, costi e materiali.

UC7: Compilazione del form valvolieri

Attori Principali: Utente.

Precondizioni: L'utente sta creando un'offerta che prevede l'inserimento di dati relativi a valvole non Starline.

Descrizione: L'utente inserisce i valori tecnici richiesti dal modulo valvolieri: BTO, RTO, ETO, BTC, RTC, ETC, MAST[g], tutti espressi in Newton-Metri (Nm).

Postcondizioni: I dati tecnici relativi alla valvola non Starline sono stati salvati e resi disponibili per la costruzione dell'offerta.

UC8: Compilazione del form attuatoristi

Attori Principali: Utente.

Precondizioni: L'utente sta creando un'offerta che prevede l'inserimento di dati relativi ad attuatori non Starline.

Descrizione: L'utente inserisce i dati tecnici richiesti dal modulo attuatoristi. I campi obbligatori sono ancora in fase di definizione.

Postcondizioni: I dati tecnici relativi all'attuatore non Starline sono stati salvati e resi disponibili per la costruzione dell'offerta.

UC9: Importazione dati componenti Starline tramite API

Attori Principali: Utente.

Precondizioni: L'utente sta creando un'offerta e intende utilizzare componenti Starline.

Descrizione: L'utente importa automaticamente i dati tecnici dei componenti Starline tramite integrazione API, per la costruzione dell'offerta.

Postcondizioni: I dati tecnici dei componenti Starline sono stati importati e associati all'offerta in fase di costruzione.

UC10: Selezione e visualizzazione dei componenti proposti

Attori Principali: Utente.

Precondizioni: L'utente sta costruendo un'offerta e ha selezionato un progetto di riferimento.

Descrizione: Il sistema propone automaticamente una lista di componenti conformi ai vincoli del progetto; l'utente seleziona quelli desiderati e le relative quantità da includere nel pannello di controllo.

Postcondizioni: I componenti selezionati sono stati associati all'offerta e saranno inclusi nel preventivo.

UC11: Visualizzazione dettagliata di un componente

Attori Principali: Utente.

Precondizioni: L'utente ha selezionato un componente dalla lista.

Descrizione: L'utente accede ai dettagli tecnici, materiali, brand e proprietà del componente selezionato.

Postcondizioni: L'utente ha visualizzato tutte le informazioni dettagliate del componente selezionato.

UC12: Visualizzazione e selezione dei materiali associati ai componenti

Attori Principali: Utente.

Precondizioni: L'utente sta visualizzando o modificando un componente.

Descrizione: L'utente consulta le informazioni sui materiali associati ai componenti e può selezionare tra quelli disponibili, identificati da nome specifico e tipo generico di materiale.

Postcondizioni: Il materiale selezionato è stato associato al componente.

UC13: Calcolo e visualizzazione del prezzo complessivo dell'offerta

Attori Principali: Sistema.

Precondizioni: L'utente ha selezionato i componenti e le quantità da includere nell'offerta.

Descrizione: Il sistema calcola e mostra in tempo reale il prezzo totale dell'offerta,

aggiornandolo in base alle selezioni effettuate.

Postcondizioni: Il prezzo complessivo dell'offerta è stato calcolato e visualizzato all'utente.

UC14: Gestione dei componenti base

Attori Principali: Amministratore.

Precondizioni: L'amministratore ha effettuato l'accesso al sistema.

Descrizione: L'amministratore accede all'anagrafica dei componenti base, dove può aggiungere, modificare o eliminare componenti. Ogni componente è caratterizzato da Type, Code, Description, Brand, Material Name, Material Type, Temperature minima e massima, Price e proprietà custom definite dinamicamente in base al tipo di componente.

Postcondizioni: Il catalogo dei componenti base è stato aggiornato secondo le operazioni svolte dall'amministratore.

UC15: Gestione dei tipi di materiali

Attori Principali: Amministratore.

Precondizioni: L'amministratore ha effettuato l'accesso al sistema.

Descrizione: L'amministratore gestisce l'elenco dei tipi di materiali (Material Type), aggiungendo nuove voci o aggiornando quelle esistenti. Esempi di tipi di materiali includono: Stainless Steel, Plastic, Brass, ecc.

Postcondizioni: L'elenco dei tipi di materiali è stato aggiornato e reso disponibile per l'associazione ai materiali specifici.

UC16: Gestione dei materiali

Attori Principali: Amministratore.

Precondizioni: L'amministratore ha effettuato l'accesso al sistema.

Descrizione: L'amministratore gestisce l'elenco dei materiali, aggiungendo nuove voci o aggiornando quelle esistenti. Ogni materiale è identificato da un nome (ad esempio, SS316[g]) e associato a un tipo di materiale già definito.

Postcondizioni: L'elenco dei materiali è stato aggiornato e i materiali sono disponibili per l'associazione ai componenti.

UC17: Gestione dei tipi di componente

Attori Principali: Amministratore.

Precondizioni: L'amministratore ha effettuato l'accesso al sistema.

Descrizione: L'amministratore gestisce la classificazione delle tipologie di componente (Component Type), aggiornando, aggiungendo o eliminando codici e descrizioni. Ogni component type è identificato da un codice (es. SH06-F1-MD-X1) e da una descrizione tecnica.

Postcondizioni: L'elenco delle tipologie di componente è stato aggiornato e reso disponibile per la definizione dei componenti.

UC18: Gestione dei brand

Attori Principali: Amministratore.

Precondizioni: L'amministratore ha effettuato l'accesso al sistema.

Descrizione: L'amministratore gestisce l'elenco dei marchi (brand) disponibili per i componenti, aggiungendo, modificando o eliminando voci dal database.

Postcondizioni: L'elenco dei brand è stato aggiornato e i marchi sono disponibili per l'associazione ai componenti.

UC19: Gestione delle proprietà personalizzate

Attori Principali: Amministratore.

Precondizioni: L'amministratore ha effettuato l'accesso al sistema.

Descrizione: L'amministratore definisce, modifica o elimina proprietà tecniche personalizzate (Property) da associare ai componenti. Ogni proprietà è identificata da un nome e da un tipo (ad esempio, pressione_max_bar di tipo Double).

Postcondizioni: L'elenco delle proprietà personalizzate è stato aggiornato e le nuove proprietà sono disponibili per l'associazione ai componenti.

UC20: Importazione ed esportazione delle linee dell'offerta in Excel

Attori Principali: Utente.

Precondizioni: L'utente ha effettuato l'accesso al sistema e ha selezionato un'offerta in fase di compilazione o già esistente.

Descrizione: L'utente può esportare le linee dell'offerta (LINE) selezionate in un file Excel, comprensivo di tutte le informazioni necessarie per la valutazione economica (ad esempio: codice componente, descrizione, quantità, prezzo unitario, prezzo totale, materiali, brand, proprietà tecniche, vincoli soddisfatti). Il file generato può essere utilizzato come preventivo di costo. L'utente può inoltre importare un file Excel strutturato secondo il formato previsto dal sistema, al fine di aggiornare o aggiungere linee all'offerta in modo massivo. Il sistema valida i dati importati, assicurando la coerenza e la correttezza delle informazioni prima di applicare le modifiche all'offerta.

Postcondizioni: Le linee dell'offerta sono state esportate in un file Excel per la generazione del preventivo, oppure importate e aggiornate nel sistema, con conferma dell'avvenuta operazione e segnalazione di eventuali errori di validazione.

4.2 Tracciamento dei requisiti

A seguito di un'approfondita analisi dei requisiti e dei casi d'uso condotta sul progetto, è stata elaborata una tabella di tracciamento che mette in relazione ciascun requisito individuato con i corrispondenti use case. Questo strumento risulta fondamentale per garantire la copertura completa delle esigenze espresse dagli stakeholder e per assicurare che ogni funzionalità prevista sia effettivamente contemplata all'interno del sistema. La tracciabilità dei requisiti rispetto agli use case rappresenta inoltre un elemento chiave sia in fase di progettazione sia durante le successive attività di verifica e validazione, facilitando l'individuazione di eventuali lacune o ridondanze.

Nel corso dell'analisi sono stati identificati diversi tipi di requisiti, ciascuno caratterizzato da specifiche finalità e livelli di priorità. Per agevolare la gestione e la consultazione dei requisiti, si è adottato un sistema di codifica strutturato, che consente di distinguere in modo univoco la natura e l'importanza di ciascun requisito. In particolare, il codice identificativo dei requisiti è articolato secondo la seguente sintassi: R(F/Q/V)(N/D/Z), dove:

R = requisito

F = funzionale, ovvero relativo alle funzionalità che il sistema deve offrire

Q = qualitativo, riferito agli aspetti non funzionali come prestazioni, usabilità, sicurezza, affidabilità, ecc.

V = di vincolo, che impone limitazioni o condizioni specifiche derivanti da normative, standard tecnici o esigenze progettuali

N = obbligatorio (necessario), requisito imprescindibile per il corretto funzionamento del sistema

D = desiderabile, requisito la cui implementazione è auspicabile ma non strettamente necessaria

Z = opzionale, requisito accessorio la cui presenza può arricchire il sistema senza comprometterne la funzionalità di base

Tale sistema di codifica favorisce una gestione strutturata e trasparente dei requisiti, facilitando sia la comunicazione tra i membri del team di progetto sia il monitoraggio dello stato di implementazione durante le diverse fasi del ciclo di vita del software.

Nelle tabelle 4.1, 4.2 e 4.3 sono riportati in modo sintetico i requisiti individuati, suddivisi per tipologia, e il relativo tracciamento rispetto agli use case definiti in fase di analisi. Questa rappresentazione consente di verificare in modo immediato la copertura dei requisiti e di garantire la coerenza tra le specifiche raccolte e le funzionalità effettivamente implementate nel sistema, costituendo un riferimento essenziale per le successive attività di sviluppo e collaudo.

4.2.1 Requisiti funzionali

I requisiti funzionali rappresentano le specifiche operative che il sistema deve soddisfare.

Tabella 4.1: Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Use Case
RFN-1	Il sistema deve permettere l'autenticazione degli utenti tramite login e registrazione, distinguendo tra utenti normali e amministratori.	Tutti
RFN-2	Il sistema deve consentire la creazione, modifica, duplicazione e visualizzazione di progetti con inserimento di nome (obbligatorio), riferimenti alle specifiche, note cliente, note interne e vincoli (bindings).	UC1, UC2, UC3, UC4, UC5
RFN-3	Il sistema deve impedire la creazione di componenti con codice duplicato e richiedere la compilazione dei campi obbligatori (type, code, brand, material).	UC14
RFN-4	Il sistema deve consentire la gestione dei materiali e dei tipi di materiale, con validazione dei campi obbligatori.	UC15, UC16
RFN-5	Il sistema deve consentire la gestione dei brand e dei tipi di componente, con validazione dei campi obbligatori.	UC17, UC18
RFN-6	Il sistema deve consentire la definizione di proprietà personalizzate (custom) da parte dell'amministratore.	UC19
RFN-7	Il sistema deve consentire la creazione di offerte e la compilazione dei form specifici per valvolieri e attuatoristi.	UC6, UC7, UC8
RFD-8	Il sistema dovrebbe consentire l'importazione dei dati relativi ai componenti tramite integrazione con l'API Starline.	UC9
RFN-9	Il sistema deve consentire la selezione dei componenti proposti e la generazione automatica del preventivo.	UC10, UC13
RFD-10	Il sistema dovrebbe consentire l'importazione e l'esportazione delle linee dell'offerta in formato Excel, garantendo la validazione dei dati e fornendo feedback all'utente.	UC20
RFN-11	Il sistema deve prevedere messaggi di errore chiari e notifiche a frontend in caso di errori nelle operazioni di salvataggio, eliminazione, import/export.	Tutti

4.2.2 Requisiti qualitativi

I requisiti qualitativi riguardano le proprietà non funzionali del sistema, quali l'usabilità e l'affidabilità.

Tabella 4.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQN-1	L'interfaccia utente deve essere responsive e utilizzabile sia da desktop che da dispositivi mobili.	Tutti
RQN-2	I dati devono essere memorizzati in un database aziendale affidabile.	Tutti

4.2.3 Requisiti di vincolo

I requisiti di vincolo specificano le tecnologie e i vincoli progettuali imposti sul sistema.

Tabella 4.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVN-1	Il sistema deve essere sviluppato utilizzando MudBlazor, ASP.NET Core, MongoDB, .NET e Docker.	Tutti
RVN-2	Il sistema deve prevedere una gestione utenti con ruoli distinti (utente, amministratore).	Tutti

Capitolo 5

Progettazione e Tecnologie

In questo capitolo vengono descritte le principali scelte progettuali e implementative adottate nello sviluppo dell'applicativo. Dopo una panoramica sulle tecnologie e sugli strumenti utilizzati, si illustrano l'architettura software e i criteri di organizzazione del codice, sia lato backend sia frontend. L'obiettivo è fornire una visione chiara e strutturata delle soluzioni tecniche adottate, evidenziando i principi di modularità, manutenibilità e scalabilità che hanno guidato la realizzazione del sistema.

5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

5.1.1 MudBlazor

MudBlazor è una libreria open source di componenti UI basata su Material Design, specificamente progettata per applicazioni sviluppate con Blazor, il framework di Microsoft per la creazione di interfacce web interattive in C#. MudBlazor consente la realizzazione di interfacce utente moderne e responsive, riducendo la necessità di ricorrere a JavaScript e promuovendo la coerenza nello sviluppo front-end.

Nel contesto del presente progetto, MudBlazor si è rivelato particolarmente utile grazie alla sua capacità di facilitare la creazione di componenti personalizzati. Ad esempio, sono stati definiti componenti condivisi (*shared*), riutilizzabili in diversi punti dell'applicazione. Un caso emblematico è rappresentato dal componente `BrandDialog.razor`, utilizzato per la gestione dei brand. Questo approccio ha favorito la modularità e la manutenibilità del codice, semplificando l'implementazione di funzionalità complesse attraverso l'uso di componenti dichiarativi e tipizzati.

Segue un esempio di implementazione del componente `BrandDialog.razor`:

Listing 5.1: Esempio di componente BrandDialog.razor in Blazor/MudBlazor.

```
<MudDialog Options="MudDialog.Options">
    <TitleContent>
        <MudStack Justify="Justify.Center" AlignItems="AlignItems
            .Center">
            <MudText Typo="Typo.h6">
                @if (IsEdit)
```

```
                {
                    @("Edit Brand")
                }
                else
                {
                    @("Add new Brand")
                }
            </MudText>
        </MudStack>
    </TitleContent>
    <DialogContent>
        <MudStack>
            @if (IsEdit)
            {
                <MudText Style="font-size: 7pt">Brand Id: @Brand.
                    Id</MudText>
            }
            <MudTextField Typo="Typo.h5" AutoFocus T="string"
                Label="Brand Name" @bind-Value="@Brand.Name"
                Immediate/>

            <MudStack Style="margin: 10px 5px">
                <MudButton StartIcon="@Icons.Material.Filled.Save
                    "
                           OnClick="CloseDialog"
                           Color="Color.Primary" Variant="Variant
                               .Filled"
                           Disabled="DisableSave()">Save
                </MudButton>
            </MudStack>
        </MudStack>
    </DialogContent>
</MudDialog>
```

L'utilizzo di MudBlazor, come evidenziato dallo snippet, permette di definire in modo chiaro e conciso la logica di presentazione e interazione dei componenti, migliorando la leggibilità e la riusabilità del codice.

MudBlazor è stato scelto rispetto ad alternative come "Telerik UI for Blazor" e "Radzen Blazor" principalmente per la sua natura open source, l'assenza di costi di licenza e la forte integrazione con il framework Blazor di Microsoft. A differenza di Telerik, MudBlazor consente una maggiore flessibilità nella personalizzazione e non impone vincoli economici. Inoltre, il supporto nativo alla compilazione in WebAssembly si è rivelato particolarmente vantaggioso in termini di efficienza e rapidità di esecuzione, aspetto rilevante per il contesto applicativo del progetto. La scelta è stata quindi guidata sia da considerazioni tecniche che da esigenze di sostenibilità economica e di efficienza operativa.

5.1.2 Scalar

Scalar è una piattaforma open source dedicata alla documentazione e al testing delle API. Consente la redazione di guide tecniche, la gestione dei riferimenti API tramite standard OpenAPI e la possibilità di testare direttamente gli endpoint esposti (GET, POST, PUT, DELETE) senza la necessità di sviluppare interfacce utente dedicate.

Scalar supporta inoltre l'integrazione con GitHub, facilitando la gestione collaborativa dei contenuti e la tracciabilità delle modifiche.

Nel contesto progettuale, Scalar è stato utilizzato per documentare e testare gli endpoint relativi al componente `Project`, migliorando la qualità della documentazione tecnica e semplificando le attività di verifica e validazione delle API.

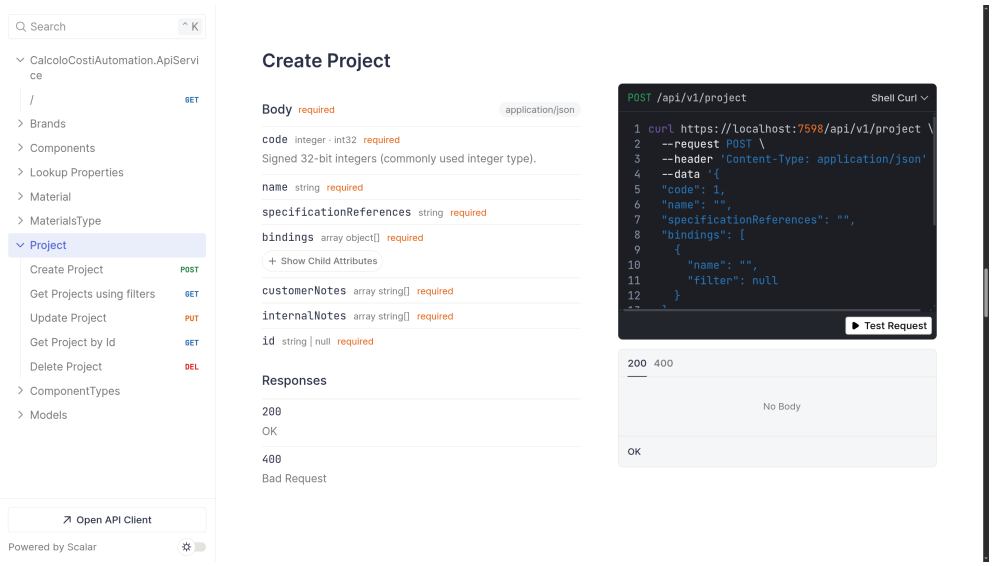

Figura 5.1: Gestione degli endpoint tramite Scalar per il componente Project.

Inizialmente, la documentazione delle API è stata gestita tramite Swagger UI, uno standard ampiamente diffuso che offre una buona visualizzazione degli endpoint. Tuttavia, con l'avanzare del progetto, è emersa la necessità di strumenti più avanzati per il testing interattivo. Scalar si è distinto per la capacità di riconoscere automaticamente la struttura delle richieste, identificando i parametri necessari per le chiamate (ad esempio GET), e per la maggiore facilità d'uso nel testing degli endpoint rispetto a Swagger UI, che non offre lo stesso livello di automazione. Questa caratteristica ha reso Scalar la soluzione preferibile per la gestione e il collaudo delle API.

5.1.3 MongoDB

MongoDB è un database NoSQL orientato ai documenti, apprezzato per la sua scalabilità e flessibilità nella gestione di dati semi-strutturati o non strutturati. Utilizza un modello di dati basato su documenti BSON, che permette una rappresentazione dinamica delle informazioni e facilita l'evoluzione dello schema dati in risposta alle esigenze applicative.

La scelta di MongoDB è stata motivata dalla necessità di gestire entità caratterizzate da proprietà non fisse(proprieta' dei db NoSQL) e variabili in funzione della tipologia di componente. A differenza dei database relazionali, che richiedono uno schema rigido e predefinito, MongoDB consente una struttura dati più flessibile, risultando particolarmente adatto alla gestione di dati dinamici e alla semplificazione della gestione di proprietà arbitrarie. Questa caratteristica si è rivelata fondamentale per il successo del progetto,.

Nello specifico, tra i database NoSQL, MongoDB è stato scelto anche perché in Starline era già presente un'istanza attiva con i dati necessari.

5.1.4 Docker

Docker è una piattaforma open source per la containerizzazione delle applicazioni. Nel progetto, Docker è stato utilizzato per containerizzare sia il database sia l'applicazione, principalmente in fase di produzione. Durante lo sviluppo, tuttavia, l'uso di Docker per l'applicazione è stato limitato per evitare rallentamenti nel ciclo di sviluppo.

La scelta di Docker è stata pressoché scontata, in quanto rappresenta lo standard de facto per la containerizzazione. Pur esistendo alternative come Podman o l'uso di macchine virtuali, Docker si è imposto per la semplicità d'uso, l'ampia disponibilità di immagini e la profonda integrazione con gli altri strumenti adottati. Questi elementi hanno reso Docker la soluzione più efficace e affidabile per garantire portabilità e coerenza tra ambienti di sviluppo e produzione.

5.1.5 Rider

Rider è un IDE sviluppato da JetBrains, specificamente orientato allo sviluppo .NET e .NET Core. Offre funzionalità avanzate per la scrittura, il refactoring e il debugging del codice, integrando strumenti di analisi statica e supportando una vasta gamma di linguaggi e framework. Rider si distingue per l'efficienza, la personalizzazione e l'integrazione con sistemi di versionamento come Git.

Nel contesto del progetto, Rider si è dimostrato uno strumento fondamentale per la gestione dei pacchetti NuGet, delle soluzioni e dei progetti. Rispetto ad altri editor come Visual Studio Code, Rider ha rappresentato un significativo passo avanti in termini di produttività, ampliando la mia conoscenza degli strumenti JetBrains.

Rider è stato scelto rispetto a Visual Studio Code e Visual Studio per la sua specializzazione nello sviluppo .NET, la ricchezza di funzionalità avanzate (come il refactoring, il debugging integrato e l'analisi statica del codice) e la profonda integrazione con strumenti di gestione dei container Docker e sistemi di versionamento come Git. Visual Studio Code, pur essendo leggero e versatile, non offre lo stesso livello di supporto nativo per progetti complessi in ambiente .NET. Inoltre, la diffusione di Rider tra i colleghi in azienda ha facilitato lo scambio di conoscenze e l'adozione di best practice condivise, contribuendo a migliorare l'efficienza e la collaborazione all'interno del team.

5.2 Architettura Esagonale

Molte delle problematiche riscontrabili sia lato utente sia lato server sono riconducibili a una carente separazione tra la logica di business e la gestione delle interazioni con le entità esterne. La regola fondamentale da seguire è che il codice relativo alla parte interna non deve mai propagarsi verso la parte esterna, preservando così l'integrità e la manutenibilità del nucleo applicativo.

Eliminando ogni asimmetria convenzionale , l'applicazione può essere descritta come un nucleo centrale che comunica con l'esterno tramite delle "porte" (*ports*). Il concetto di porta richiama l'analogia con le porte di un sistema operativo o di un dispositivo elettronico: qualsiasi componente che rispetti il protocollo della porta può essere collegato, indipendentemente dalla sua implementazione concreta. In questo contesto, il protocollo della porta si realizza attraverso una API, ovvero un contratto di comunicazione tra il dominio applicativo e l'entità esterna.

Per ciascun servizio o dispositivo esterno è previsto un adattatore (*adapter*) che si occupa di tradurre le chiamate API nei segnali comprensibili da quella specifica tecnologia e viceversa. Ad esempio, un'interfaccia grafica utente rappresenta un adattatore che converte le azioni dell'utente in chiamate verso le API dell'applicazione. Altri esempi di adattatori includono i test automatici, le interfacce HTTP e i connettori verso i database.

Dal punto di vista architetturale, il protocollo di comunicazione con il database dovrebbe essere progettato in modo indipendente dalla tecnologia effettivamente utilizzata (SQL, NoSQL, ecc.). Questo approccio consente di sostituire il meccanismo di persistenza senza modificare la logica di business, ma semplicemente implementando un nuovo adattatore per la stessa porta. In tal modo, il sistema mantiene elevata flessibilità e adattabilità rispetto a future evoluzioni tecnologiche.

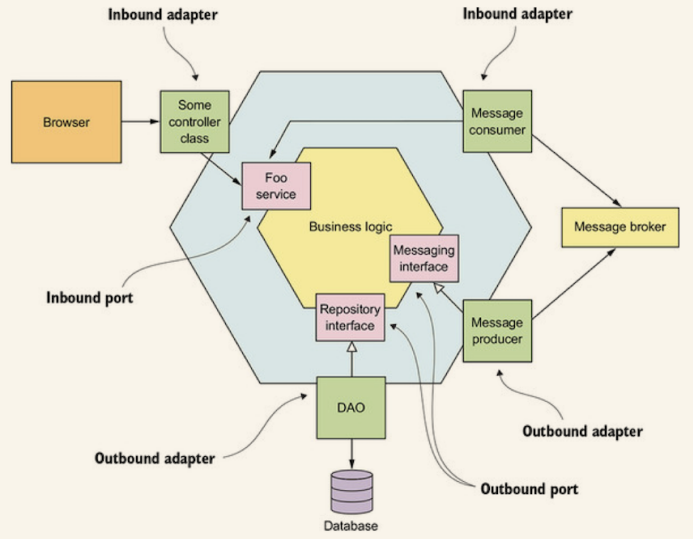

Figura 5.2: Rappresentazione schematica dell'architettura esagonale[4].

Nel nostro progetto, questa impostazione si traduce in una separazione rigorosa tra il dominio applicativo (logica di business) e i meccanismi di interazione con il mondo

esterno (API HTTP, database MongoDB, Scalar), garantendo così manutenibilità, testabilità e flessibilità evolutiva.

5.2.1 Implementazione della Dependency Injection

Elemento abilitante dell'architettura esagonale è la *Dependency Injection*, che consente di iniettare le dipendenze necessarie invece di istanziarle direttamente.

All'avvio dell'applicazione, la configurazione dei servizi e la mappatura degli endpoint avvengono secondo la seguente logica implementativa:

Listing 5.2: Configurazione dei servizi e mappatura degli endpoint

```
var app = builder.Build();

// Configurazione pipeline HTTP
app.UseExceptionHandler();

if (app.Environment.IsDevelopment())
{
    app.MapOpenApi();
    app.MapScalarApiReference(configureOptions =>
        configureOptions.WithTitle("Costi Automation API V1"));
    app.MapGet("/", () => "Hello World!");
}

app.UseHttpsRedirection();
app.UseCors();

// Mappatura dinamica degli endpoint dei moduli
app.MapModulesEndpoints();

app.Run();
```

La funzione `MapModulesEndpoints` consente di registrare dinamicamente gli endpoint esposti da ciascun modulo, secondo il seguente schema:

Listing 5.3: Registrazione dinamica degli endpoint dei moduli

```
public static WebApplication MapModulesEndpoints(this
    WebApplication app)
{
    var logger = app.Logger;
    var api = app.MapGroup("/api");
    foreach (var module in RegisteredModules)
    {
        logger.LogDebug("Mapping endpoints for module {ModuleName
            }", module.GetType().Name);
        module.MapEndpoints(api);
    }
    return app;
}
```

Ogni modulo di dominio (ad esempio, il modulo `Component`) implementa l'interfaccia `IModule` e registra i propri servizi tramite il metodo `RegisterModule`, sfruttando la *Dependency Injection* per l'iniezione delle dipendenze:

Listing 5.4: Registrazione dei servizi di dominio tramite Dependency Injection

```
public class ComponentModule : IModule
{
    public bool IsEnabled => true;
    public int Order => 10;

    public WebApplicationBuilder RegisterModule(
        WebApplicationBuilder builder)
    {
        builder.Services.AddComponent(); // Registrazione dei
            servizi di dominio
        return builder;
    }

    public IEndpointRouteBuilder MapEndpoints(
        IEndpointRouteBuilder application)
    {
        var mapGroup = application.MapGroup("/v1/component")
            .WithTags("Components");

        mapGroup.MapPost("/", ComponentEndpoints.
            HandleCreateComponent)
            .Produces(StatusCodes.Status400BadRequest)
            .Produces(StatusCodes.Status200OK)
            .WithSummary("Create Component")
            .WithOpenApi();

        // ... altri endpoint ...

        return application;
    }
}
```

In questo modo, la logica di business dei componenti resta completamente indipendente dai dettagli implementativi degli endpoint e dei servizi di persistenza, coerentemente con i principi dell'architettura esagonale.

5.2.2 Esempio pratico con/senza Dependency Injection

Ecco un esempio comparativo base per capire la logica della Dependency Injection.

Senza Dependency Injection In questo caso, la classe crea direttamente l'oggetto di cui ha bisogno:

Listing 5.5: Senza Dependency Injection

```
public class ComponentService
{
private ComponentRepository repository = new ComponentRepository
    ();

public void CreateComponent(Component component)
{
    repository.Save(component);
```

```
}
}
```

Qui, `ComponentService` dipende direttamente da `ComponentRepository`. Se si vuole cambiare il tipo di repository, bisogna modificare il codice della classe.

Con Dependency Injection Con la Dependency Injection, la dipendenza viene passata dall'esterno:

Listing 5.6: Con Dependency Injection

```
public class ComponentService
{
private IComponentRepository repository;

public ComponentService(IComponentRepository repository)
{
    this.repository = repository;
}

public void CreateComponent(Component component)
{
    repository.Save(component);
}
}
```

In questo modo, `ComponentService` non si preoccupa di quale repository viene usato: può essere facilmente sostituito senza modificare la classe.

5.3 Progettazione Backend

5.3.1 Business Logic

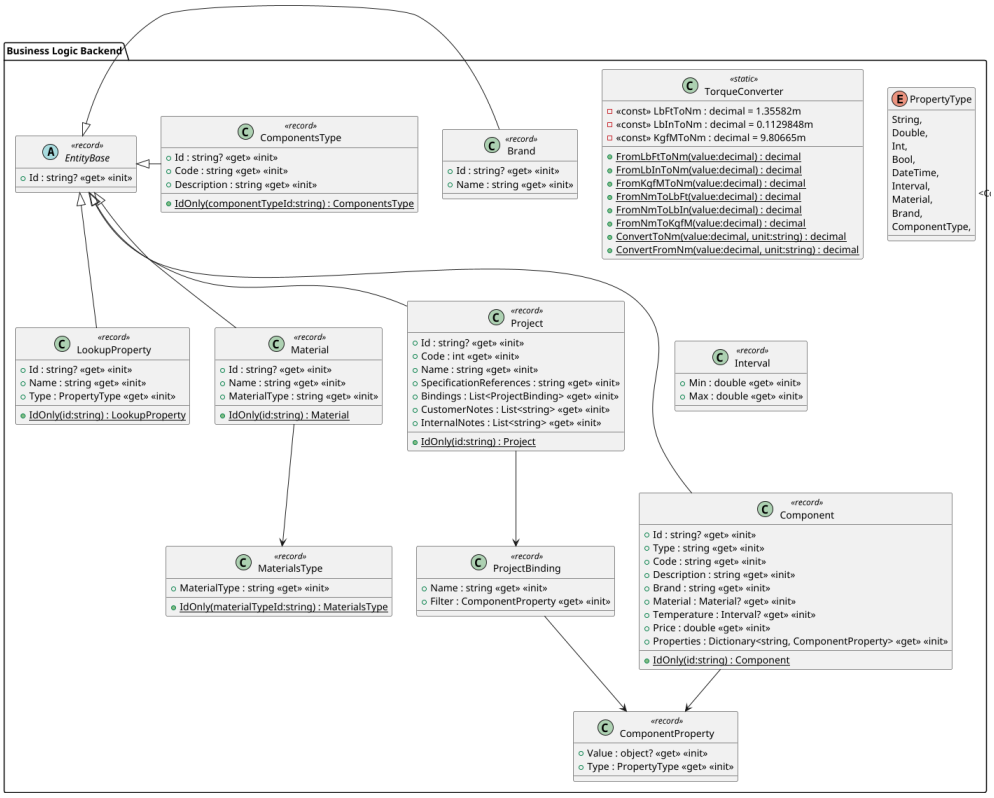

Figura 5.3: Schema della logica di business

La logica di business illustrata in Figura 5.3 è stata progettata con l'obiettivo di garantire la massima indipendenza da librerie e dipendenze esterne, così da ottenere un codice altamente manutenibile e facilmente modificabile. Al centro dell'architettura si trova l'interfaccia `EntityBase`, dalla quale derivano le principali entità del dominio. Questo approccio consente di manipolare in modo dinamico e polimorfico i diversi tipi della logica di business, facilitando l'implementazione di operazioni generiche e la gestione coerente delle entità, anche in scenari di evoluzione futura del dominio applicativo.

Le classi principali individuate sono `Component`, `Project`, `ComponentsType`, `MaterialType`, `Material`, `Brand` e `Property`. Un elemento centrale è rappresentato dalla classe `ComponentProperty`, utilizzata sia da `Component` sia da `Project`. Questo permette di modellare proprietà customizzabili tramite un dizionario, associando a ciascun componente un insieme dinamico di attributi, ciascuno caratterizzato da un tipo e da un valore. Tale soluzione consente di estendere le proprietà dei componenti senza modificare la struttura delle classi, in linea con il principio di open/closed. Inoltre, la presenza di proprietà customizzabili rende il sistema estremamente flessibile

e adattabile a requisiti che possono evolvere nel tempo, senza necessità di interventi invasivi sull'architettura.

Il campo `Value` di `ComponentProperty` è di tipo `object`, permettendo così di assegnare valori eterogenei (ad esempio stringhe, numeri interi, numeri decimali, booleani, date, intervalli, materiali, brand, tipi di componente) a seconda del tipo specificato dal campo `Type`, che è un enumerativo (`PropertyType`). Quest'ultimo comprende i principali tipi di dato gestiti dal sistema: `String`, `Double`, `Int`, `Bool`, `DateTime`, `Interval`, `Material`, `Brand`, `ComponentType`, garantendo una forte tipizzazione e una maggiore sicurezza a runtime. L'adozione di questa soluzione, pur offrendo grande flessibilità, impone particolare attenzione nella validazione dei dati e nella gestione della coerenza tra le entità che condividono proprietà customizzabili, come `Component` e `Project`. In tal senso, la logica di business può essere facilmente estesa per includere regole di validazione centralizzate o strategie di gestione delle dipendenze tra proprietà.

La classe `Interval` viene utilizzata principalmente per rappresentare intervalli di temperatura, ma, essendo prevista in `PropertyType`, può essere impiegata per qualsiasi proprietà customizzabile che richieda un range di valori. Questo consente di modellare in modo naturale vincoli e limiti sulle proprietà dei componenti, aumentando la capacità espressiva del modello.

Un ulteriore elemento di rilievo è la classe `TorqueConverter`, che incapsula la logica di conversione dei valori di momento torcente e viene utilizzata, ad esempio, nella gestione dei form lato front-end. L'incapsulamento di logiche di calcolo specifiche in classi dedicate favorisce la separazione delle responsabilità, la riusabilità del codice e la facilità di testing, in linea con le best practice della programmazione orientata agli oggetti.

Nel complesso, la logica di business presentata si configura come un esempio di progettazione robusta e flessibile, in cui l'indipendenza dalle dipendenze esterne, la gestione dinamica delle proprietà, l'estendibilità e la separazione delle responsabilità costituiscono i pilastri fondamentali dell'architettura. Tale impostazione garantisce una solida base per l'evoluzione futura del sistema e per l'integrazione con eventuali nuove funzionalità o domini applicativi.

5.3.2 Infrastruttura di persistenza e serializzazione

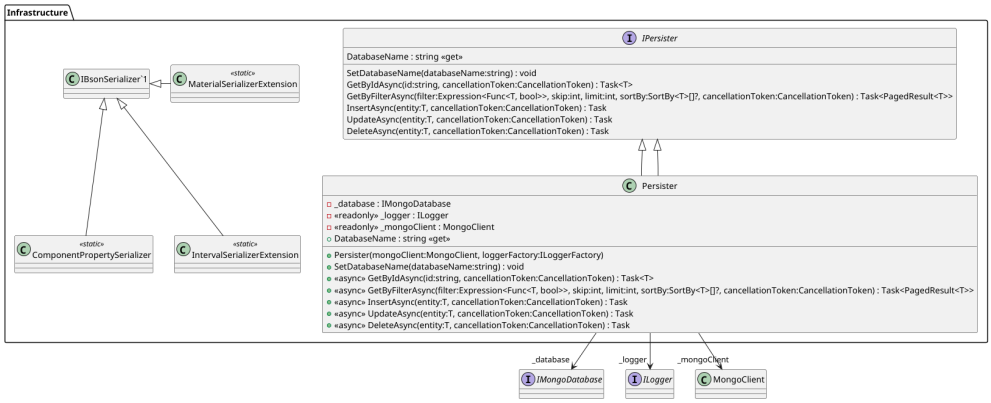

Figura 5.4: Schema dell'infrastruttura

L'infrastruttura illustrata in Figura 5.4 costituisce il livello di astrazione responsabile della gestione delle operazioni di persistenza dei dati su database MongoDB, garantendo al contempo la separazione tra la logica di business e i dettagli implementativi dell'accesso ai dati. Al centro di questa architettura si colloca l'interfaccia `IPersister`, che definisce un insieme di operazioni generiche per la manipolazione delle entità applicative: recupero per identificativo, query filtrate, inserimento, aggiornamento e cancellazione. Tale interfaccia viene implementata dalla classe `Persister`, la quale si occupa di interagire concretamente con il database tramite le dipendenze `IMongoDatabase`, `ILogger` e `MongoClient`, iniettate tramite costruttore secondo i principi dell'inversione delle dipendenze e della testabilità del codice.

Il flusso operativo prevede che la logica di business, attraverso pattern come Facade e Service, si interfacci esclusivamente con `IPersister`, demandando a quest'ultima ogni dettaglio relativo alla persistenza. In questo modo, la logica applicativa rimane completamente indipendente dalla tecnologia di storage sottostante, facilitando la sostituibilità e la manutenibilità del sistema. Tale approccio favorisce l'evoluzione futura dell'applicazione, sia in termini di scalabilità che di adattamento a nuove tecnologie di persistenza.

Un aspetto di particolare rilievo riguarda la gestione della serializzazione degli oggetti. Poiché MongoDB memorizza i dati in formato BSON, analogo al JSON, risulta necessario serializzare e deserializzare correttamente le entità applicative, soprattutto quando queste includono tipi complessi o customizzati non nativamente supportati dal driver MongoDB. A tal fine, l'infrastruttura prevede l'implementazione di serializer personalizzati, come evidenziato dalla presenza delle classi `MaterialSerializerExtension`, `IntervalSerializerExtension` e `ComponentPropertySerializer`, tutte derivate dall'interfaccia generica `IBsonSerializer<T>`. Queste estensioni consentono di definire regole specifiche per la serializzazione di oggetti complessi, garantendo la coerenza e l'integrità dei dati memorizzati.

L'adozione di metodi statici per l'estensione dei serializer permette di mantenere il codice modulare e facilmente estendibile, caratteristica fondamentale in contesti in cui la struttura dei dati può evolvere o richiedere requisiti specifici di serializzazione. In particolare, la serializzazione customizzata si rivela essenziale per la gestione di proprietà dinamiche, intervalli di valori e riferimenti a entità complesse, come materiali o brand, assicurando la corretta rappresentazione e persistenza di tali oggetti all'interno del database.

Nel complesso, l'infrastruttura di persistenza e serializzazione presentata si configura come un esempio di progettazione robusta e scalabile, in cui l'astrazione dei dettagli implementativi, la modularità e la possibilità di estendere facilmente il supporto a nuovi tipi di dati costituiscono i principali punti di forza. Tale impostazione garantisce non solo la manutenibilità e la testabilità del sistema, ma anche la sua capacità di adattarsi efficacemente a futuri requisiti evolutivi o a cambiamenti tecnologici.

5.3.3 Infrastruttura di gestione delle operazioni CRUD per il dominio Progetto

La struttura illustrata in Figura 5.5 rappresenta il livello di backend responsabile della gestione delle operazioni CRUD relative al dominio `Project`. Tale infrastruttura si configura come un esempio di architettura stratificata, in cui la separazione delle responsabilità tra i diversi componenti garantisce modularità, manutenibilità e testabilità del sistema.

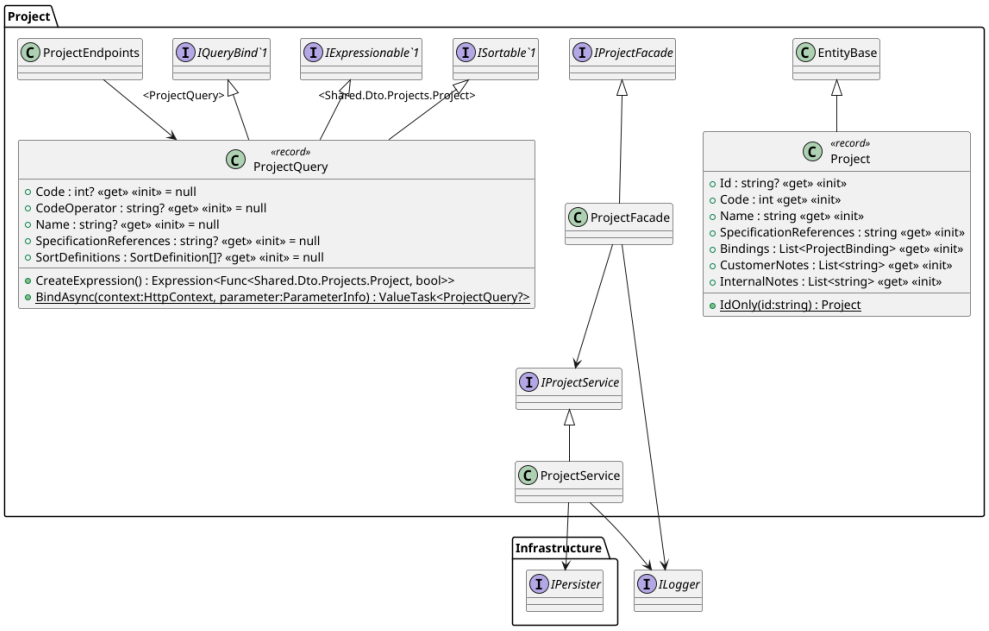

Figura 5.5: Schema dell'infrastruttura di backend per la gestione dei progetti

Al vertice del flusso operativo si colloca la classe `ProjectEndpoints`, incaricata di esporre gli endpoint HTTP che costituiscono il punto di ingresso per le richieste provenienti dal client. Ogni endpoint corrisponde a una specifica operazione sul dominio `Project` (ad esempio, recupero, inserimento, aggiornamento o cancellazione di un progetto) e si avvale di parametri di query e di filtri tipizzati, come evidenziato dalla dipendenza verso la classe `ProjectQuery`. Quest'ultima implementa le interfacce `IQueryBind`, `IExpressionable` e `ISortable`, consentendo la definizione di criteri di filtraggio e ordinamento dinamici e fortemente tipizzati.

La logica di business vera e propria è incapsulata all'interno della classe `ProjectFacade`, che implementa l'interfaccia `IProjectFacade`. La `Facade` funge da intermediario tra gli endpoint e i servizi di dominio, orchestrando le operazioni richieste e delegando l'esecuzione delle stesse al servizio sottostante, rappresentato dalla classe `ProjectService`. Quest'ultima implementa l'interfaccia `IProjectService` e si occupa di interagire con il livello di persistenza tramite l'interfaccia generica `IPersister`, garantendo così l'astrazione dai dettagli implementativi dell'accesso ai dati.

Il modello di dominio `Project`, derivato da `EntityBase`, rappresenta l'entità applicativa centrale, caratterizzata da attributi quali identificativo, codice, nome, riferimenti a specifiche, associazioni e note. La presenza di metodi statici, come `IdOnly`, facilita la creazione di istanze parziali utili in contesti di referenziazione o validazione.

Il flusso delle operazioni prevede che una richiesta proveniente dal client venga ricevuta da `ProjectEndpoints`, la quale, attraverso la dependency injection, si avvale dei servizi offerti da `ProjectFacade`. Quest'ultima, a sua volta, invoca i metodi di `ProjectService`, che interagisce con il repository di persistenza (`IPersister`) per eseguire le operazioni richieste sul database. Tale catena di responsabilità consente di isolare la logica applicativa dai dettagli di storage, favorendo la sostituibilità e la

scalabilità del sistema.

Un aspetto rilevante dell'architettura è l'adozione sistematica della dependency injection per la gestione delle dipendenze tra componenti, in particolare per quanto riguarda i servizi di dominio (`IProjectService`) e i meccanismi di logging (`ILogger`). Questo approccio incrementa la testabilità e la coerenza dell'intero sistema, permettendo l'introduzione di strategie di mocking o di logging avanzato senza impattare la logica di business.

In sintesi, l'infrastruttura di backend per la gestione dei progetti si configura come un esempio di buona progettazione software, in cui la separazione delle responsabilità, l'astrazione dei dettagli implementativi e l'adozione di pattern consolidati (come Facade e Service) garantiscono robustezza, estensibilità e facilità di manutenzione. Tale impostazione risulta particolarmente efficace in contesti enterprise, dove la complessità delle operazioni e la necessità di evoluzione continua richiedono architetture solide e scalabili.

Estensione del modello agli altri componenti di backend

L'architettura e il flusso operativo descritti per il componente `Project` sono stati utilizzati come modello per tutti i componenti di backend dell'applicazione. Ogni dominio applicativo segue la stessa struttura, composta da endpoint, facade, servizi di dominio e repository di persistenza, con le opportune personalizzazioni in base alle specifiche esigenze.

5.4 Progettazione Frontend

L'architettura del front-end dell'applicativo è stata progettata sfruttando la tecnologia Blazor e la libreria MudBlazor per la realizzazione di un'interfaccia utente moderna, modulare e facilmente estendibile. MudBlazor mette a disposizione un ricco set di componenti UI riutilizzabili, che consentono di strutturare il codice in maniera modulare e di mantenere un elevato grado di coerenza stilistica e funzionale tra le diverse sezioni dell'applicativo.

Il sistema è suddiviso in diverse pagine principali, ciascuna delle quali è dedicata a specifiche funzionalità operative e amministrative. Tale organizzazione facilita la separazione delle responsabilità, migliora la manutenibilità e rende l'esperienza utente più lineare e intuitiva.

5.4.1 Pagina "Valuation"

La pagina "Valuation" è dedicata alla costruzione dell'offerta per la realizzazione di un pannello di controllo. L'utente può inserire i parametri tecnici tramite due form distinti: il "form valvolieri"(Valve) e il "form attuatoristi"(Actuator). Questi form sono resi disponibili solo qualora sia necessario specificare dati relativi a valvole o attuatori non appartenenti al marchio Starline; in caso contrario, possono essere omessi, consentendo di proseguire direttamente alla selezione del progetto e alla generazione dell'offerta sulla base dei vincoli tecnici già presenti.

Nel caso di valvole non Starline, il form prevede la compilazione di campi obbligatori relativi ai valori di torque, espressi in Newton-Metri (Nm), con possibilità di conversione automatica da altre unità di misura tramite un'apposita interfaccia. La validazione dei dati avviene in fase di salvataggio, garantendo la coerenza e la correttezza delle

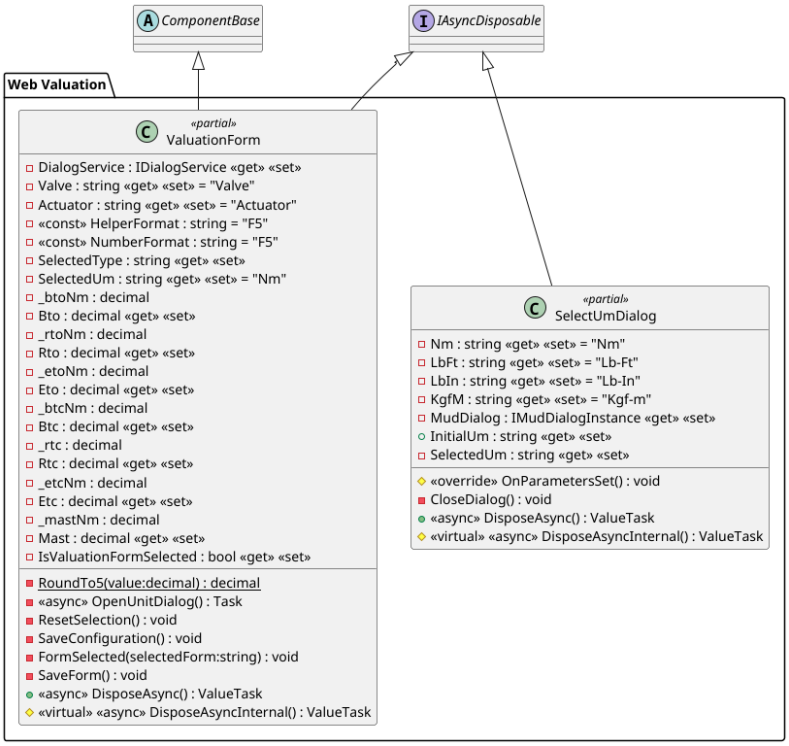

Figura 5.6: Diagramma UML della struttura del Valuation Form

informazioni. Per il form attuatoristi, la definizione dei campi obbligatori è ancora in fase di definizione.

Quando si utilizzano componenti Starline, è possibile importare i dati necessari tramite un'API dedicata, assicurando uniformità e completezza delle informazioni richieste. L'offerta risultante comprende i valori selezionati tramite i form, il progetto associato, i filtri tecnici (bindings) ereditati dal progetto selezionato e un contenitore di "LINE", ovvero gli item proposti dal sistema in base ai vincoli tecnici. L'utente può selezionare solo i componenti conformi ai vincoli previsti, senza possibilità di aggiunta manuale di elementi non conformi.

5.4.2 Pagina "Progetti"

La pagina "Progetti" offre una panoramica completa di tutti i progetti disponibili, presentati in una griglia che consente operazioni di creazione, duplicazione e modifica. Ogni progetto è caratterizzato da un'anagrafica dettagliata che include: codice numerico automatico (normalizzato a quattro cifre), denominazione libera, riferimenti testuali alle specifiche, elenco di vincoli tecnici (bindings), annotazioni del cliente e note interne.

I vincoli tecnici sono selezionabili da una lista predefinita, gestita in una tabella dedicata e modificabile dall'amministratore. La duplicazione di un progetto comporta la copia integrale di tutte le informazioni, compresi vincoli, specifiche e note. Non sono previste relazioni gerarchiche tra progetti. I vincoli tecnici rimangono modificabili anche dopo l'associazione del progetto a un'offerta, garantendo flessibilità nella gestione

dei requisiti e consentendo eventuali aggiornamenti in corso d'opera.

5.4.3 Pagina "Admin Panel"

La pagina "Admin Panel" è riservata agli utenti con privilegi amministrativi e consente la gestione completa dei gruppi di componenti e dei componenti base necessari per la costruzione dei progetti e dei relativi vincoli. In questa sezione è possibile creare, modificare ed eliminare brands, tipologie e materiali dei componenti, nonché definire proprietà personalizzate e vincoli tecnici (bindings) applicabili ai progetti.

Le componenti base sono utilizzate per la costruzione di:

- **Componenti**: elementi fisici del pannello di controllo, caratterizzati da attributi come `Type`, `Code`, `Description`, `Brand`, `Material`, `Temperature`, `Price` e un insieme di proprietà tecniche specifiche.
- **Vincoli (Bindings)**: filtri tecnici applicabili ai progetti, ciascuno definito da un nome e da un filtro su una proprietà tecnica specifica.

5.4.4 Gestione e Manipolazione dei JSON nel Front-End

Un aspetto centrale dell'implementazione front-end riguarda la gestione e la manipolazione dei dati tramite oggetti JSON. Per ogni componente e progetto, sono stati definiti specifici tipi di dato, come `ComponentJson` e `ProjectJson`, che rappresentano la versione serializzata e strutturata delle entità di dominio. L'intero front-end opera su questi oggetti JSON, questo permette di disaccoppiarsi con il backend.

Come illustrato in Figura 5.7, la classe `ComponentJson` eredita dalle interfacce `ICloneable` e `IDtoConvertible<T>`, e incapsula tutte le proprietà rilevanti di un componente, quali identificativi, descrittori, caratteristiche tecniche e commerciali, oltre a un dizionario di proprietà aggiuntive (`Properties`). La presenza di metodi come `ToDto()`, `Clone()` e l'override di `Equals` e `GetHashCode` consente di convertire, duplicare e confrontare agevolmente gli oggetti, facilitando le operazioni di manipolazione sia lato front-end che in eventuali sincronizzazioni con il back-end.

Questa struttura dati viene utilizzata in molteplici contesti: dalla visualizzazione nelle griglie, ai dialog di creazione e modifica, fino alle operazioni di cancellazione. L'adozione di un modello dati standardizzato come `ComponentJson` assicura coerenza e robustezza nel frontend e disaccoppiamento con il backend.

Dal punto di vista architetturale, anche il front-end si interfaccia con il back-end adottando un approccio basato su *facade* e *services*, analogamente a quanto avviene lato server. I servizi del front-end orchestrano le chiamate alle API e gestiscono la logica di interazione con i dati, mentre le facade forniscono un livello di astrazione ulteriore, centralizzando la gestione degli oggetti JSON e semplificando l'accesso alle funzionalità. Questo modello favorisce il disaccoppiamento tra le componenti dell'interfaccia utente e la logica applicativa, migliorando la manutenibilità, la testabilità e la scalabilità dell'intero sistema.

5.4.5 Disposable Pattern

Un aspetto cruciale nella progettazione di componenti complessi in Blazor, come la classe `ValuationForm` (si veda Figura 5.6), è la corretta gestione delle risorse, siano esse gestite o non gestite dal garbage collector. In questo contesto, il **Disposable Pattern** rappresenta una best practice fondamentale per garantire il rilascio tempestivo

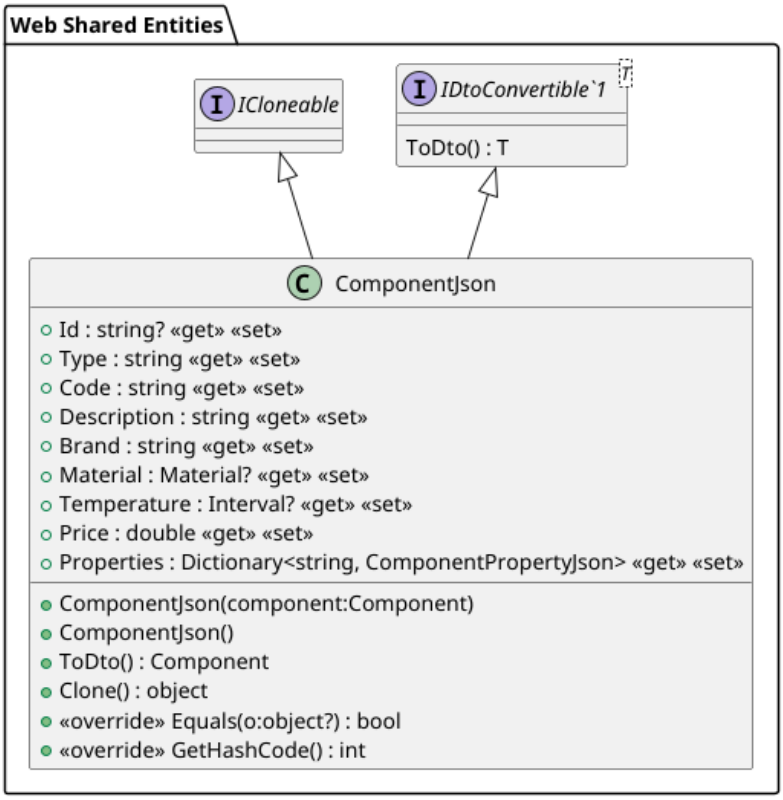

Figura 5.7: Struttura UML della classe `ComponentJson` e delle sue relazioni

e sicuro delle risorse, prevenendo memory leak e assicurando la stabilità e l'efficienza dell'applicazione.

Nel caso specifico di `ValuationForm`, la classe implementa l'interfaccia `IAsyncDisposable`, rendendo disponibile il metodo `DisposeAsync()`. Tale metodo viene utilizzato per liberare risorse asincrone, come ad esempio la cancellazione di sottoscrizioni a eventi, la chiusura di connessioni o la liberazione di oggetti che richiedono una pulizia esplicita e asincrona.

Un esempio di implementazione del Disposable Pattern per la classe `ValuationForm` è il seguente:

```
public class ValuationForm : IAsyncDisposable
{
public async ValueTask DisposeAsync()
{
await DisposeAsyncInternal();
GC.SuppressFinalize(this);
}

protected virtual async ValueTask DisposeAsyncInternal()
{
    await Task.Yield();
```

```
}
}
```

In questa implementazione, il metodo `DisposeAsync()` richiama una funzione di pulizia interna (`DisposeAsyncInternal()`) e, successivamente, sopprime la finalizzazione dell'oggetto tramite `GC.SuppressFinalize(this)`, ottimizzando così la gestione della memoria.

Capitolo 6

Conclusioni

Presentazione dell'interfaccia grafica allo stato attuale e dei requisiti soddisfatti.

6.1 Prodotto allo stato attuale

Al termine del periodo di stage, il prodotto software realizzato si configura come una piattaforma gestionale solida, in cui la maggior parte delle logiche di business e delle funzionalità di backend risultano già quasi completamente implementate. Gran parte del lavoro svolto ha riguardato la progettazione e lo sviluppo delle API, delle logiche di persistenza e delle operazioni di gestione dati, garantendo così la robustezza dell'infrastruttura sottostante.

Per quanto concerne l'interfaccia utente, essa si presenta attualmente in una versione prototipale che consente comunque di esplorare e utilizzare le principali funzionalità offerte dal sistema. Tuttavia, è opportuno sottolineare che molte delle attività di sviluppo future saranno necessariamente focalizzate sul perfezionamento e sull'ottimizzazione dell'esperienza utente, sia dal punto di vista grafico sia da quello funzionale. In particolare, l'interfaccia, pur risultando già funzionale, necessita di ulteriori sviluppi e rifiniture per raggiungere un livello di maturità e completezza coerente con la solidità del backend.

Le immagini e le descrizioni presentate nelle sezioni seguenti testimoniano lo stato attuale dell'interfaccia principale alla conclusione dello stage, fornendo una base concreta e documentata per i successivi interventi di miglioramento e ottimizzazione.

6.1.1 Struttura generale

L'applicativo è caratterizzato da un menù laterale che consente agli utenti di navigare tra le diverse sezioni del sistema. Le principali pagine accessibili sono:

- la pagina di costruzione della *valuation*;
- la pagina di gestione dei progetti;
- l'*admin panel*, riservato agli amministratori.

Le prime due sezioni sono accessibili agli utenti standard, mentre l'*admin panel* è dedicato alla gestione avanzata delle entità del sistema.

6.1.2 Admin Panel

La dashboard dell'*admin panel* (Figura 6.1) costituisce il punto di accesso alle funzionalità di amministrazione. Essa presenta, tramite apposite *cards*, tutte le entità gestite dal sistema: *Components*, *Brands*, *Components Type*, *Materials*, *Materials Type* e *Properties*. Ogni card dispone di un pulsante *Manage* che permette di accedere alle relative pagine di gestione.

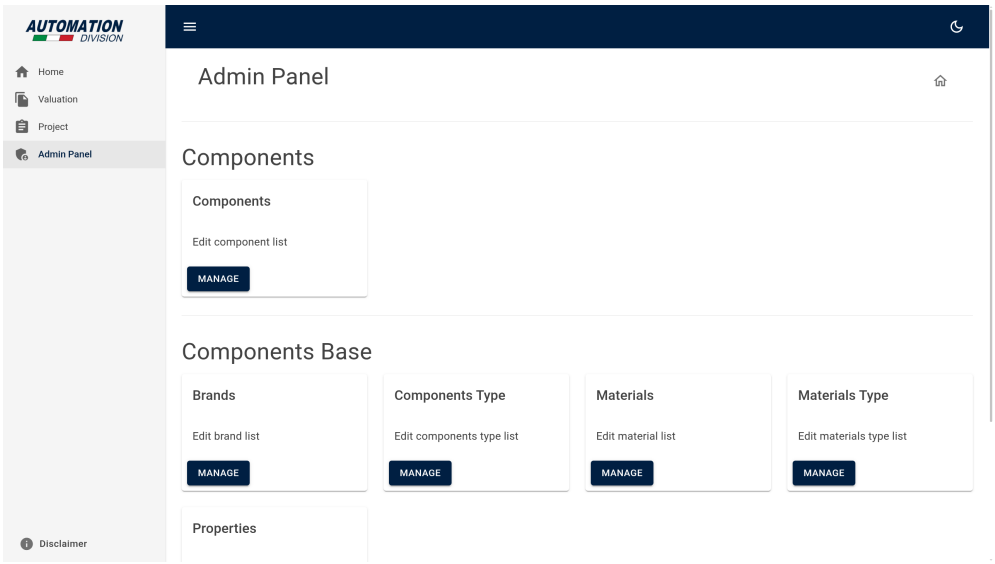

Figura 6.1: Dashboard dell'*admin panel* con accesso alle diverse entità gestionali.

Gestione dei materiali

Accedendo alla sezione dedicata ai materiali, viene visualizzata una griglia (Figura 6.2) che elenca tutti i materiali presenti nel database. Su ciascun elemento è possibile eseguire operazioni di creazione, lettura, aggiornamento e cancellazione.

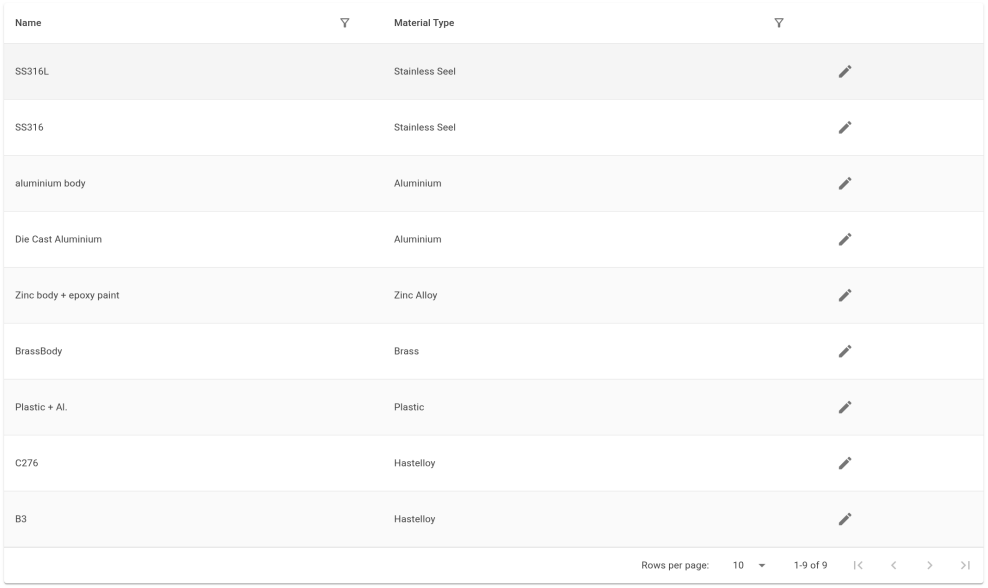

Figura 6.2: Griglia di gestione dei materiali.

La creazione di un nuovo materiale avviene tramite un dialog dedicato (Figura 6.3), accessibile tramite il pulsante *Add new Material*. In questo dialog, i campi necessari vengono inseriti manualmente, mentre il tipo di materiale è selezionabile tramite un componente *autocomplete* nativo di MudBlazor, che consente di evitare errori di digitazione e garantisce la coerenza dei dati.

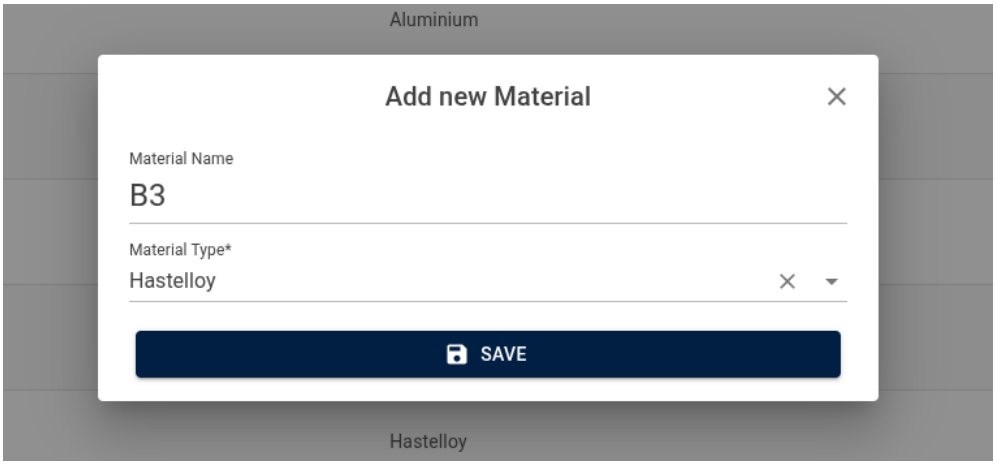

Figura 6.3: Dialog di creazione di un nuovo materiale con selettore *autocomplete*.

Dopo aver completato l'inserimento e confermato l'operazione tramite il pulsante *Save*, il materiale viene salvato nel database e la griglia aggiornata in tempo reale,

se tutto avviene con successo viene visuaizzato un messaggio di conferma in basso a sinistra (Figura 6.4).

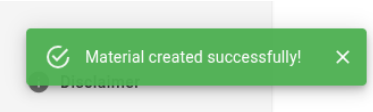

Figura 6.4: Griglia aggiornata con il nuovo materiale inserito.

Gestione dei componenti

La gestione dei componenti rappresenta una funzionalità centrale e più articolata del sistema. I componenti modellano le entità applicate ai pannelli di controllo e sono caratterizzati da una maggiore complessità informativa. La griglia di gestione (Figura 6.5) mostra i campi di default di ciascun componente e consente le operazioni CRUD.

Type	Code	Description	Brand	Material Name	Material Type	Temp Min
AFR	342A8401AD	Air Filter Regulator are used to remove liquid water and particulate matter from compressed air sources	Asco	SS316L	Stainless Seel	0

Rows per page: 10 1-1 of 1

Figura 6.5: Griglia di gestione dei componenti.

La creazione di un nuovo componente avviene tramite un dialog avanzato (Figura 6.6), in cui i campi principali (Type, Description, Brand, Material) sono selezionabili tramite *autocomplete*, mentre i restanti campi vengono compilati manualmente. Il sistema consente inoltre di associare al componente proprietà personalizzate (*custom properties*) definite dall'amministratore.

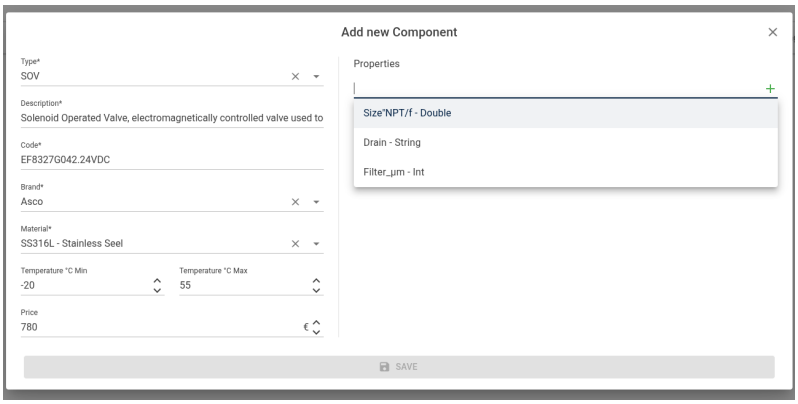

Figura 6.6: Dialog di creazione di un nuovo componente con selezione di proprietà custom.

Ad esempio, selezionando la proprietà *size NPT/f*[1], il sistema permette di inserire il valore desiderato (ad esempio, "1/4 NPT/f") secondo il tipo previsto dalla proprietà (stringa, numero, booleano, ecc.). Una volta salvato, il componente viene aggiunto alla griglia (Figura 6.7).

Type	Code	Description	Brand	Material Name	Material Type	Temp Min
AFR	342A8401AD	Air Filter Regulator are used to remove liquid water and particulate matter from compressed air sources	Asco	SS316L	Stainless Seel	0
SOV	EF8327G042.24VDC	Solenoid Operated Valve, electromagnetically controlled valve used to regulate fluid or gas flow	Asco	SS316L	Stainless Seel	-20

Rows per page: 10 1-2 of 2

Figura 6.7: Componente creato con proprietà customizzata.

Funzionalità avanzate di griglia

Le griglie di MudBlazor implementate nel sistema permettono di ordinare e filtrare i dati secondo diversi criteri. Ad esempio, è possibile ordinare i componenti per tipo in ordine discendente o per prezzo in ordine ascendente (Figura 6.8).

[1]NPT (National Pipe Thread) è uno standard statunitense per filettature di tubi, utilizzato per connettori filettati in impianti idraulici e pneumatici. Un esempio: "1/4 NPT" indica una filettatura di tubo con diametro nominale di 1/4 di pollice.

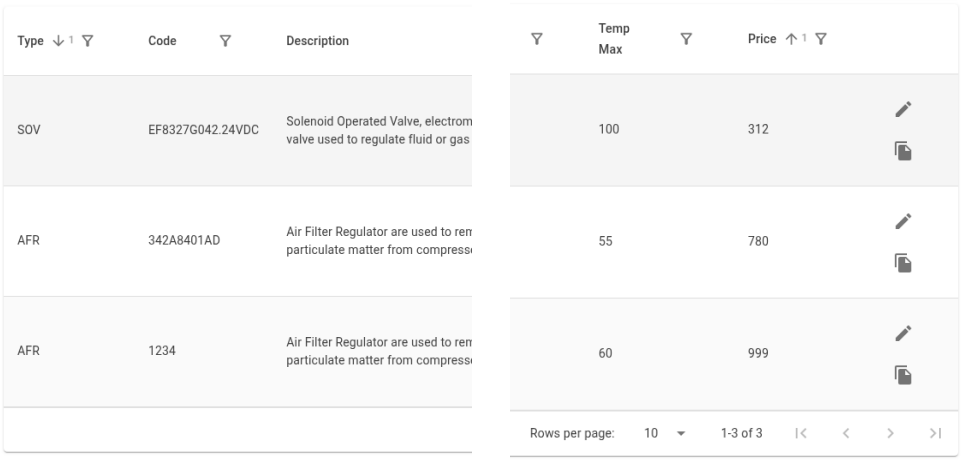

Figura 6.8: Esempi di ordinamento per tipo (sinistra) e per prezzo (destra).

Inoltre, sono disponibili filtri su tutti i campi di default. La Figura 6.9 mostra l'interfaccia di creazione di un filtro personalizzato, in cui l'utente può selezionare i criteri desiderati (ad esempio, brand e soglia di prezzo).

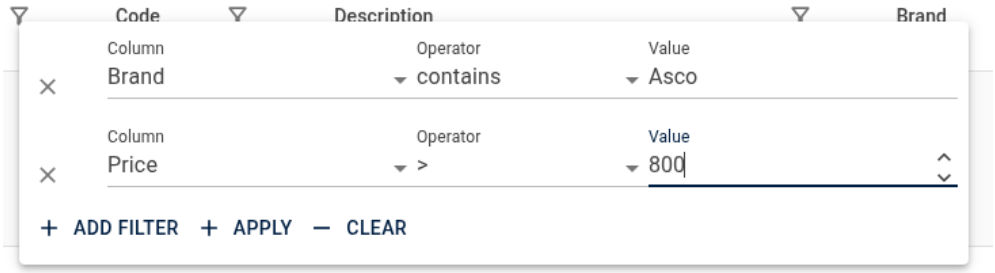

Figura 6.9: Interfaccia di creazione di un filtro personalizzato sui componenti.

Applicando il filtro, come illustrato nella Figura 6.10, vengono selezionati e visualizzati solo i componenti che soddisfano i criteri impostati, ad esempio quelli con brand "Asco" e prezzo superiore a 800.

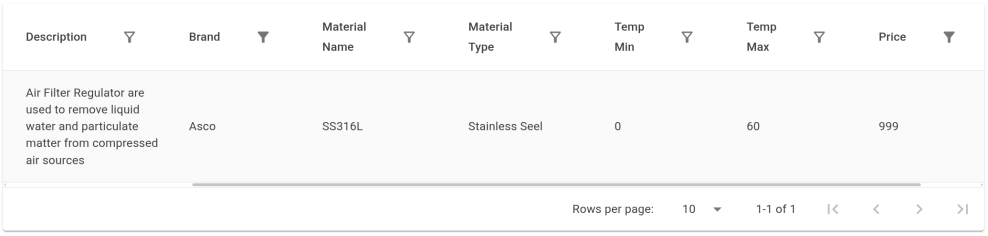

Figura 6.10: Filtro applicato per brand "Asco" e prezzo superiore a 800.

6.1.3 Gestione dei progetti

La pagina *Progetti* rappresenta il punto di accesso principale per la creazione, la visualizzazione e la gestione dei progetti da parte dell'utente. L'interfaccia si presenta come una griglia che elenca tutti i progetti disponibili, consentendo operazioni di creazione, duplicazione e modifica. Ogni progetto è strutturato per raccogliere e documentare in modo sistematico le specifiche tecniche e operative richieste dal cliente o definite nell'ambito del progetto stesso.

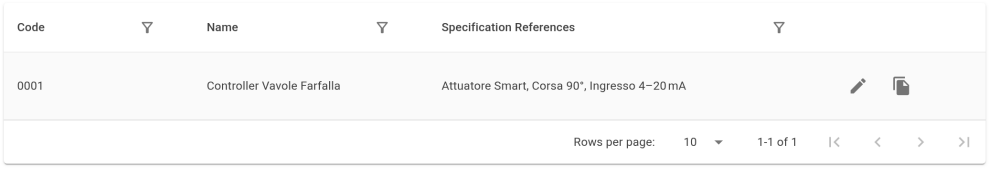

Figura 6.11: Vista iniziale della pagina *Progetti*: la griglia mostra i progetti disponibili con i relativi codici, nomi e riferimenti alle specifiche tecniche. Tramite il pulsante *Add New Project* è possibile avviare la creazione di un nuovo progetto.

I principali campi che caratterizzano ciascun progetto sono i seguenti:

- **Project Code**: identificativo numerico progressivo, generato automaticamente dal sistema e normalizzato a quattro cifre;
- **Project Name**: campo testuale libero per la denominazione del progetto;
- **Specification References**: campo testuale libero per l'indicazione delle specifiche tecniche di riferimento;
- **Bindings**: elenco dinamico di vincoli tecnici associati al progetto, selezionabili da una lista predefinita;
- **Customer Notes**: elenco di annotazioni e richieste specifiche del cliente;
- **Internal Notes**: elenco di annotazioni interne, destinate all'uso esclusivo del team di progetto.

La creazione di un nuovo progetto avviene tramite un'interfaccia dedicata, che si apre selezionando il pulsante *Add New Project*. In questa fase, l'utente può inserire i dati principali del progetto, le note per il cliente, le note interne e i primi vincoli tecnici (*bindings*), come illustrato in Figura 6.12.

Figura 6.12: Interfaccia di creazione di un nuovo progetto: inserimento dei dati principali, delle note cliente e interne.

L'utente può aggiungere ulteriori vincoli tecnici e specifiche, selezionandoli da una lista predefinita. Questa flessibilità consente di adattare la struttura del progetto alle esigenze applicative, senza imporre una rigidità strutturale. La Figura 6.13 mostra la scheda progetto completata con l'inserimento di ulteriori vincoli e parametri tecnici.

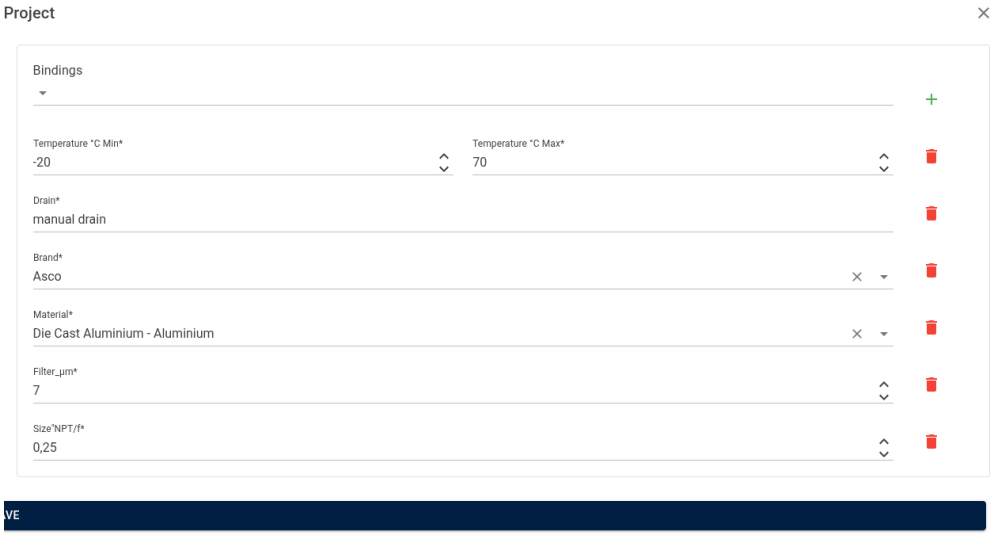

Figura 6.13: Completamento della scheda progetto con l'inserimento di ulteriori vincoli tecnici e parametri specifici, selezionabili dalla lista predefinita.

Un aspetto rilevante della progettazione di questa sezione è proprio la flessibilità nella gestione dei vincoli tecnici (*bindings*). Ogni progetto può prevedere un numero variabile di vincoli, che vengono aggiunti secondo necessità. La lista dei vincoli

disponibili è gestita e aggiornata dall'amministratore nella tabella dedicata, garantendo così coerenza e aggiornamento centralizzato.

La funzionalità di duplicazione permette di creare una copia integrale di un progetto esistente, replicando tutte le informazioni associate, inclusi vincoli tecnici, specifiche, note del cliente e note interne.

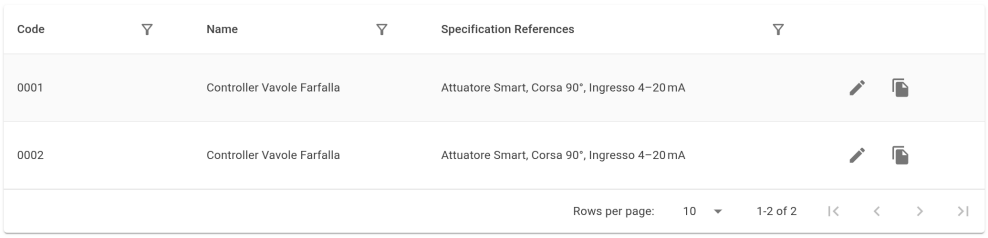

Figura 6.14: Griglia aggiornata dei progetti dopo la creazione di una nuova voce.

6.1.4 Pagina Valuation

La pagina denominata *Valuation* consente la costruzione di un'offerta relativa alla realizzazione di un pannello di controllo per un cliente. Attualmente, il frontend implementato si limita alla gestione del *form valvolieri*, mentre ulteriori funzionalità sono in fase di sviluppo.

In questa sezione l'utente può definire i parametri dell'offerta tramite la compilazione del *Valve Form*, qualora sia necessario inserire dati relativi a valvole non appartenenti al marchio Starline. Il form prevede l'inserimento dei seguenti campi obbligatori, tutti espressi in Newton-metri (Nm): BTO, RTO, ETO, BTC, RTC, ETC, MAST. Il sistema accetta valori decimali e consente la conversione automatica da unità di misura alternative (Lb-Ft, Lb-In, Kgf-m) tramite un'interfaccia dedicata, garantendo la normalizzazione interna dei dati in Nm.

La Figura 6.15 mostra il form valvolieri compilato con valori espressi direttamente in Newton-metri.

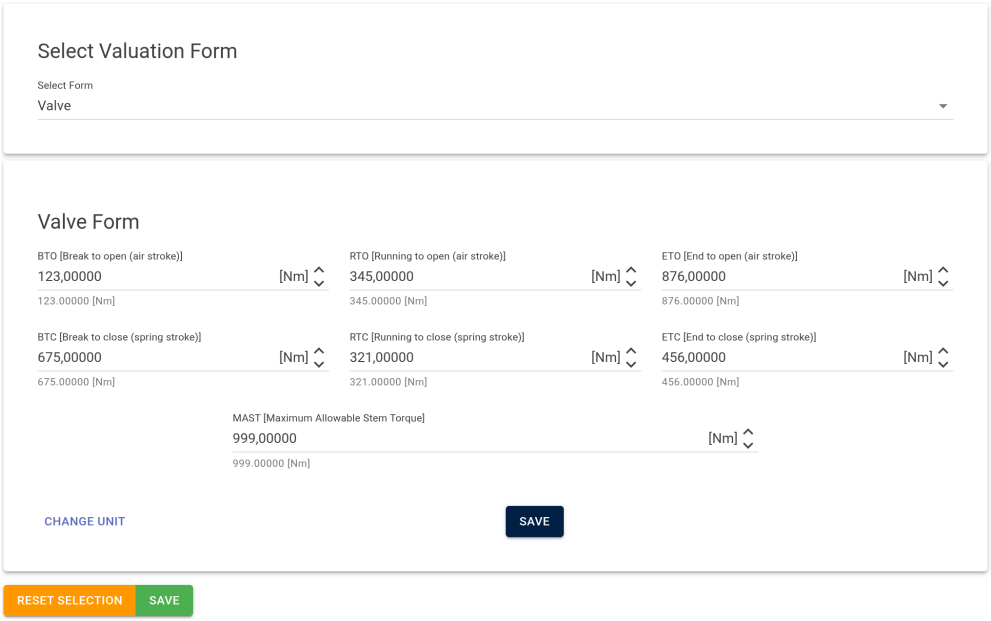

Figura 6.15: Form valvolieri compilato con valori espressi in Newton-metri (Nm).

Nel caso in cui i dati siano forniti in unità di misura diverse, l'utente può selezionare l'opzione *Change Unit* per aprire un dialog che consente di scegliere l'unità di misura desiderata, come illustrato in Figura 6.16.

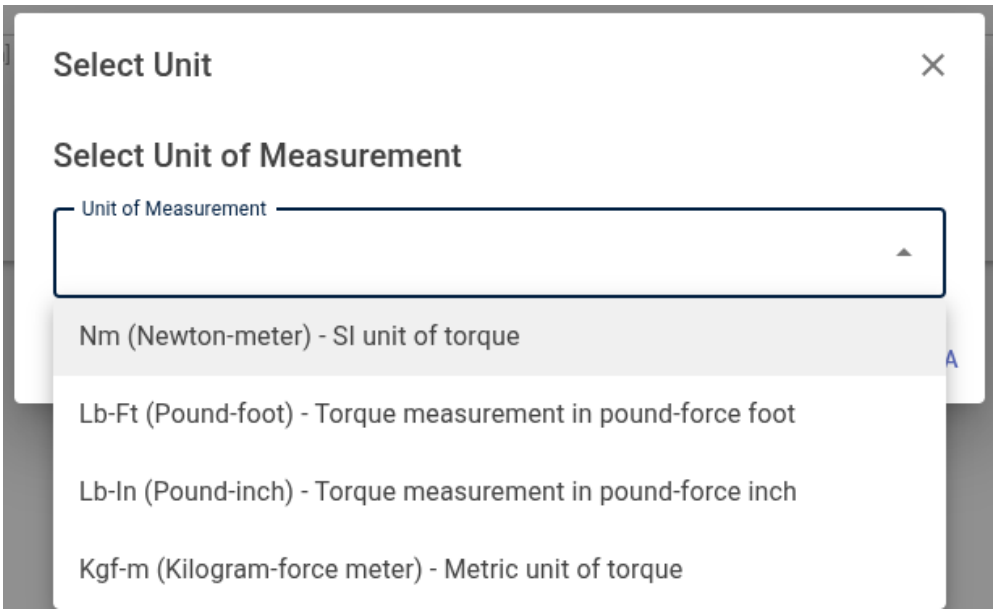

Figura 6.16: Dialog di selezione dell'unità di misura: l'utente può scegliere tra Nm, Lb-Ft, Lb-In e Kgf-m.

Una volta selezionata l'unità di misura alternativa e inseriti i valori, il sistema effettua automaticamente la conversione e normalizza i dati in Nm, come visibile in Figura 6.17.

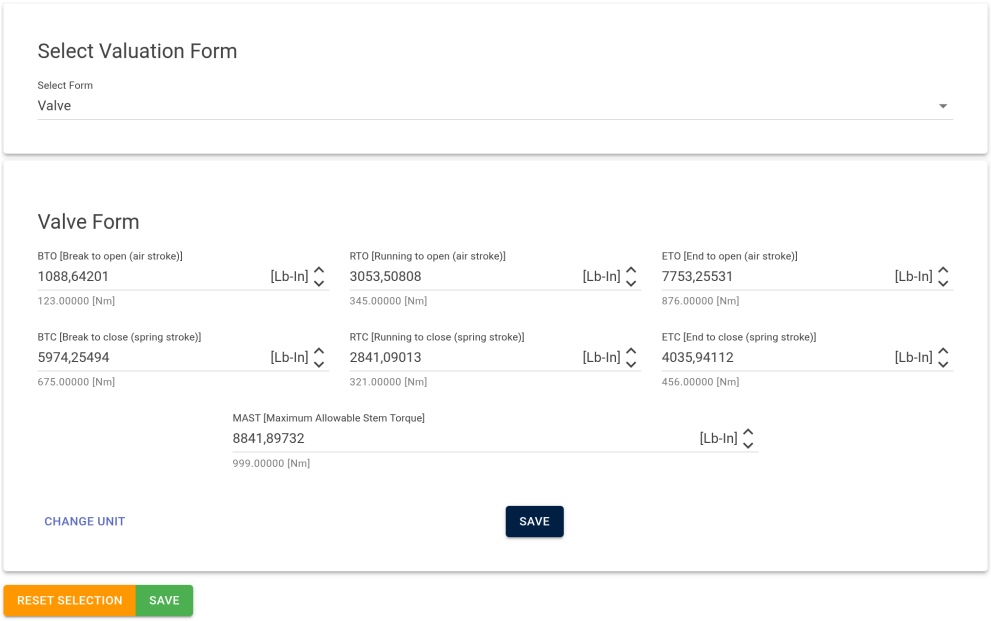

Figura 6.17: Esempio di form compilato con valori inseriti in Lb-In e convertiti automaticamente in Newton-metri (Nm).

La validazione dei dati inseriti avviene in fase di salvataggio, assicurando la coerenza e la correttezza dei valori. Per quanto riguarda il *form attuatoristi*, la definizione dei campi obbligatori è ancora oggetto di discussione con il cliente e non è attualmente disponibile nel frontend.

Nelle prossime fasi di sviluppo, la pagina *Valuation* sarà arricchita con la possibilità di importare progetti esistenti e selezionare le relative *LINE* (item proposti), generati automaticamente dal sistema sulla base dei vincoli tecnici associati al progetto. Sarà inoltre implementata l'integrazione con le API Starline per l'importazione automatica dei dati relativi ai componenti, garantendo uniformità e completezza delle informazioni. L'utente potrà quindi selezionare il progetto di interesse, visualizzare i filtri di progetto (bindings in sola lettura) e scegliere tra i componenti suggeriti quelli che soddisfano i requisiti specifici, senza possibilità di aggiunta manuale di componenti non conformi ai vincoli.

6.2 Stato dei requisiti

La seguente tabella riassume lo stato di soddisfacimento dei requisiti progettuali al termine del periodo di stage. Si fa riferimento ai requisiti definiti nella Sezione 4.2. La tabella consente di valutare in modo sintetico e trasparente il livello di copertura delle specifiche individuate, evidenziando le aree già completate e quelle che necessitano di ulteriori sviluppi.

Tabella 6.1: Stato di soddisfacimento dei requisiti al termine dello stage

Requisito	Descrizione	Soddisfatto
RFN-1	Autenticazione utenti (login/registrazione, ruoli)	Parziale
RFN-2	Gestione progetti (creazione, modifica, visualizzazione)	Sì
RFN-3	Validazione componenti (codice univoco, campi obbligatori)	Sì
RFN-4	Gestione materiali e tipi di materiale	Sì
RFN-5	Gestione brand e tipi di componente	Sì
RFN-6	Definizione proprietà personalizzate	Sì
RFN-7	Creazione offerte e compilazione form specifici	Parziale
RFD-8	Importazione dati componenti via API Starline	No
RFN-9	Selezione componenti e generazione automatica preventivo	No
RFD-10	Import/export offerte in Excel	No
RFN-11	Gestione messaggi di errore e notifiche frontend	Sì
RQN-1	Interfaccia responsive	Sì
RQN-2	Database aziendale affidabile	Sì
RVN-1	Tecnologie richieste	Sì
RVN-2	Gestione utenti con ruoli distinti	Sì

6.3 Valutazione critica e prospettive di sviluppo

Nonostante i risultati conseguiti durante il periodo di stage, il prodotto presenta alcuni limiti e aree di miglioramento che meritano una riflessione critica. In primo luogo, l'interfaccia grafica, sebbene già funzionale nelle sue componenti essenziali, si trova ancora in una fase prototipale e necessita di ulteriori interventi per ottimizzare l'esperienza utente, la coerenza stilistica e l'accessibilità. In particolare, l'assenza di alcune funzionalità avanzate, come l'importazione automatica dei dati tramite API Starline, la generazione automatica dei preventivi e l'esportazione delle offerte in formato Excel, limita attualmente la piena operatività della piattaforma.

Un ulteriore aspetto migliorabile riguarda la gestione dell'autenticazione e dei ruoli utente, attualmente implementata solo parzialmente. La sicurezza e la granularità dei permessi rappresentano elementi cruciali per un sistema gestionale destinato a un contesto aziendale, e richiedono pertanto uno sviluppo più approfondito, sia dal punto di vista tecnico sia da quello organizzativo.

Dal punto di vista architetturale, la robustezza del backend costituisce una solida base per le evoluzioni future, ma sarà necessario garantire un costante allineamento tra le logiche di business e le esigenze operative che emergeranno con l'utilizzo reale del sistema. In questa prospettiva, l'adozione di metodologie di sviluppo agili e l'implementazione di test automatici potranno favorire una maggiore reattività e affidabilità nelle fasi di manutenzione e aggiornamento.

6.4 Valutazione personale dell'esperienza di stage

L'esperienza di stage ha rappresentato per me un'occasione di crescita significativa, sia dal punto di vista tecnico che personale. Pur avendo già partecipato a progetti software durante il corso di Ingegneria del Software, ho potuto constatare come la realtà aziendale sia molto diversa dal contesto accademico. In particolare, è stato interessante osservare come ogni team di sviluppo adatti le metodologie agili, come Scrum, alle proprie esigenze specifiche. Ho compreso che, al di là delle linee guida teoriche, ogni gruppo struttura il proprio lavoro in modo flessibile, modificando strumenti e processi in funzione delle tempistiche, delle risorse e delle competenze disponibili.

Questa esperienza mi ha permesso di apprezzare il valore della collaborazione e del confronto diretto con i colleghi. L'interazione quotidiana con membri del team più esperti si è rivelata estremamente formativa: spesso, un suggerimento pratico o una spiegazione informale si sono dimostrati più efficaci di quanto si possa apprendere dai manuali o dalla documentazione online. In questo modo, ho potuto acquisire nuove competenze tecniche e scoprire strumenti utili per facilitare lo sviluppo, che difficilmente avrei incontrato durante il solo percorso universitario.

Infine, lavorare in un ambiente professionale mi ha permesso di comprendere meglio le dinamiche organizzative e le responsabilità individuali, rafforzando la mia motivazione a proseguire nel settore dello sviluppo software. Ritengo che questa esperienza abbia arricchito il mio percorso formativo e mi abbia fornito una prospettiva più concreta sulle sfide e sulle opportunità offerte dal mondo del lavoro.

Appendice A

Origine dell'Agile e di SCRUM (Appendice al Capitolo 2)

> I've sat through far too many "crying in the beer" sessions where all the energy for change was dissipated in the intensity of the complaining. Once you see an idea for improvement that makes sense to you, do it.
>
> Kent Beck

A.1 Origine del Manifesto Agile

L'11 febbraio 2001, un gruppo di diciassette esperti di sviluppo software si riunì al Lodge at Snowbird, un resort situato tra le montagne innevate dello Utah. L'obiettivo era chiaro: discutere un'alternativa ai modelli di sviluppo software tradizionali, pesanti e incentrati sulla documentazione. L'incontro non fu casuale; un anno prima, nel 2000, Kent Beck aveva organizzato dei primi confronti presso il Rogue River Lodge in Oregon, dove già si delineavano idee di maggiore leggerezza e flessibilità nei processi di sviluppo.

La scelta del termine "Agile" non fu immediata. Inizialmente, si parlava di "Lightweight" per indicare la natura snella di queste metodologie. Tuttavia, questo termine non convinceva del tutto. Durante le discussioni emerse la proposta di "Agile", che meglio rappresentava l'adattabilità e la velocità di risposta ai cambiamenti. Martin Fowler, in modo ironico, notò che molti americani avrebbero avuto difficoltà a pronunciare correttamente "Agile", ma il termine fu approvato con entusiasmo.

L'incontro si svolse in un clima di apertura e collaborazione: oltre alle discussioni tecniche, i partecipanti si concessero momenti di relax sulle piste da sci. Bob Martin, con il suo solito senso dell'umorismo, definì quel momento "mushy", esprimendo gratitudine per aver condiviso idee con persone animate dagli stessi valori di fiducia e rispetto reciproco.

Fu in quel contesto che nacque l'Agile Alliance, un gruppo impegnato nella diffusione dei principi del Manifesto. Anche la scelta di Snowbird come location fu significativa: Chicago venne scartata per il freddo e la mancanza di svaghi, mentre Snowbird, con le

sue piste da sci, rappresentava un perfetto equilibrio tra lavoro e relax. Non mancarono episodi divertenti, come le difficoltà di Martin Fowler nel mantenersi in piedi sugli sci.

Da quell'incontro tra neve e idee, Agile è diventato un pilastro dello sviluppo software, trasformando il modo di lavorare dei team di sviluppo in tutto il mondo.

A.1.1 Approfondimento: Kent Beck

Kent Beck è un ingegnere del software statunitense, noto per essere il creatore di Extreme Programming (XP), una metodologia di sviluppo software che enfatizza la collaborazione, la semplicità e il feedback continuo. È stato uno dei 17 firmatari originali del Manifesto Agile nel 2001. Laureato in informatica all'Università dell'Oregon, ha contribuito significativamente allo sviluppo dei design pattern e ha co-creato JUnit, un framework di unit testing per Java, insieme a Erich Gamma. Beck ha lavorato presso Facebook come Technical Coach dal 2011 al 2018 e, dal 2019, è impiegato presso Gusto, dove guida i team ingegneristici nello sviluppo di sistemi di gestione delle retribuzioni per le piccole imprese.

A.1.2 Approfondimento: Robert C. Martin

Robert Cecil Martin, conosciuto anche come "Uncle Bob", è un ingegnere del software, autore e consulente statunitense. È uno dei firmatari del Manifesto Agile e ha avuto un ruolo fondamentale nella definizione dei principi SOLID, un insieme di linee guida per la progettazione di software orientato agli oggetti. Martin ha fondato Object Mentor, una società di consulenza focalizzata su C++, Java, progettazione a oggetti, UML, sviluppo agile e Extreme Programming. Tra le sue opere più influenti si ha "Clean Code", testo nel quale esprime le buone prassi per produrre codice pulito e della programmazione software.

A.1.3 Approfondimento: Martin Fowler

Martin Fowler è un ingegnere del software britannico, autore e speaker internazionale, specializzato in analisi e progettazione orientata agli oggetti, UML, pattern e metodologie di sviluppo agile, inclusa Extreme Programming. Nato a Walsall, Inghilterra, nel 1963, ha iniziato la sua carriera nel settore software negli anni '80. Dal 2000 lavora presso ThoughtWorks, una società di consulenza e integrazione di sistemi, dove ricopre il ruolo di Chief Scientist. Fowler è noto per aver introdotto e diffuso il concetto di "Refactoring" con il suo libro omonimo del 1999 e per aver contribuito alla definizione del pattern architetturale "Presentation Model".

A.2 Scrum: Origine e Significato

Il termine "Scrum"[1] trae origine dal rugby, dove indica la "mischia": una particolare formazione di gioco caratterizzata da una stretta collaborazione e coordinazione tra i giocatori per il raggiungimento di un obiettivo comune. Nella mischia del rugby, infatti, tutti i componenti della squadra si dispongono compatti, spingendo insieme e sostenendosi a vicenda per avanzare e conquistare terreno contro la resistenza dell'avversario. Ogni giocatore ha un ruolo specifico, ma il successo dipende dalla capacità del gruppo di agire come un'unica entità, in cui ciascuno mette le proprie

[1] 13.

forze al servizio della collettività. Questo concetto è stato trasposto nello sviluppo software per rappresentare un modello di lavoro in cui il team agisce come un'unità coesa, supportandosi reciprocamente e mantenendo una comunicazione costante e trasparente. Così come nella mischia del rugby ogni giocatore collabora attivamente per "spingere avanti" la squadra, anche nel team Scrum ciascun membro contribuisce al successo collettivo, intervenendo in caso di difficoltà e condividendo responsabilità e risultati, con l'obiettivo comune di far progredire ("spingere") il prodotto software verso il traguardo.

Dal punto di vista storico, Scrum nasce come evoluzione delle riflessioni sul lavoro di team multidisciplinari e auto-organizzati, concetti già evidenziati da Takeuchi e Nonaka nel 1986. Tuttavia, è con il contributo di Jeff Sutherland e Ken Schwaber che Scrum viene applicato e formalizzato come framework per lo sviluppo software, fino alla sua presentazione ufficiale alla conferenza OOPSLA nel 1995. L'affermazione definitiva di Scrum si realizza nel 2001, con la pubblicazione del Manifesto Agile, che ne riconosce e valorizza i principi fondamentali, sancendone l'appartenenza tra i metodi agili di riferimento.

A.2.1 Approfondimento: Hirotaka Takeuchi e Ikujiro Nonaka

Hirotaka Takeuchi e Ikujiro Nonaka sono due accademici giapponesi che, con l'articolo "The New New Product Development Game" pubblicato su Harvard Business Review nel 1986, hanno introdotto la metafora della "scrum" nel rugby per descrivere l'efficacia di team multidisciplinari, auto-organizzati e collaborativi nello sviluppo di nuovi prodotti. Questo lavoro ha rappresentato la base teorica e ispirazionale per la nascita del framework Scrum, sottolineando l'importanza della flessibilità, della rapidità e della collaborazione continua nei processi di innovazione.

A.2.2 Approfondimento: Jeff Sutherland

Jeff Sutherland è un informatico statunitense, co-creatore di Scrum. Dopo una carriera nella ricerca medica e nell'ingegneria del software, negli anni '90 ha collaborato con Ken Schwaber per formalizzare Scrum come framework per la gestione di progetti complessi. Sutherland ha presentato Scrum per la prima volta alla conferenza OOPSLA nel 1995 e ha partecipato attivamente alla stesura del Manifesto Agile nel 2001. È autore di numerosi testi e CEO di Scrum Inc., società di consulenza internazionale.

A.2.3 Approfondimento: Ken Schwaber

Ken Schwaber è un ingegnere del software statunitense, co-ideatore di Scrum insieme a Jeff Sutherland. Schwaber ha contribuito in modo determinante alla diffusione di Scrum, promuovendo la formazione e la certificazione attraverso la fondazione della Scrum Alliance. È stato tra i firmatari del Manifesto Agile e autore di testi fondamentali come "Agile Project Management with Scrum".

Acronimi e abbreviazioni

ANSI/ASME Acronimi di American National Standards Institute (ANSI) e American Society of Mechanical Engineers (ASME), due enti statunitensi che definiscono standard tecnici internazionalmente riconosciuti per la progettazione, la costruzione e il collaudo di componenti meccanici, tra cui le valvole industriali. Le normative ANSI/ASME stabiliscono criteri di classificazione, dimensionamento e resistenza alla pressione per garantire sicurezza, affidabilità e intercambiabilità dei prodotti. 80

API Application Programming Interface. Interfaccia software che consente l'interazione e lo scambio di dati tra applicazioni diverse, permettendo l'integrazione di funzionalità e servizi esterni all'interno di un sistema software. Le API sono fondamentali per l'interoperabilità tra sistemi eterogenei e per l'automazione dei processi. 37, 44, 45, 47, 48, 56, 57, 60, 70, 71

API American Petroleum Institute. Organizzazione statunitense che sviluppa standard internazionali per la certificazione, la progettazione e la produzione di componenti e sistemi destinati all'industria petrolifera e del gas. Gli standard API sono ampiamente riconosciuti a livello globale per la sicurezza e la qualità dei prodotti industriali. 6, 22, 23, 33, 41

ATEX Acronimo di "ATmosphères EXplosibles", indica la Direttiva Europea 2014/34/UE relativa agli apparecchi e sistemi di protezione destinati a essere utilizzati in atmosfere potenzialmente esplosive. La direttiva stabilisce i requisiti essenziali di sicurezza e salute per la progettazione, la costruzione e la commercializzazione di tali apparecchiature all'interno dell'Unione Europea. [1]. 4, 7, 11

bar Unità di misura della pressione non appartenente al Sistema Internazionale, comunemente utilizzata in ambito industriale. 1 bar equivale a 100.000 Pascal (Pa) nel Sistema Internazionale. 6

BTC Break to close (spring stroke). Coppia necessaria per avviare la chiusura della valvola nella fase di corsa a molla. 21, 36, 68

BTO Break to open (air stroke). Coppia necessaria per avviare l'apertura della valvola nella fase di corsa pneumatica. 21, 36, 68

CRUD Create, Read, Update, Delete. Operazioni fondamentali per la gestione dei dati in un sistema informativo, rappresentano le quattro funzioni base delle applicazioni di persistenza dati. 23, 53, 63

DTO Data Transfer Object. Oggetto utilizzato per trasferire dati tra processi o livelli di un'applicazione, riducendo la dipendenza tra componenti e facilitando la serializzazione dei dati. 22–24

ETC End to close (spring stroke). Coppia necessaria per completare la chiusura della valvola alla fine della corsa a molla. 21, 36, 68

ETO End to open (air stroke). Coppia necessaria per completare l'apertura della valvola alla fine della corsa pneumatica. 21, 36, 68

FAT Il Factory Acceptance Test (FAT) è una procedura di verifica condotta presso il sito del fornitore per accertare che sistemi e componenti soddisfino i requisiti contrattuali e le specifiche tecniche prima della spedizione.[7]. 3

IDE Integrated Development Environment. Ambiente di sviluppo integrato che offre strumenti per la scrittura, il debug e la gestione del codice, migliorando l'efficienza dello sviluppo software. 21, 46

ISO International Organization for Standardization. Ente internazionale che elabora e pubblica standard tecnici e normativi riconosciuti a livello mondiale, finalizzati a garantire la qualità, la sicurezza e l'efficienza dei prodotti, dei servizi e dei sistemi nei diversi settori industriali. 6

MAST Maximum Allowable Stem Torque. Coppia massima ammissibile sull'albero della valvola, valore limite da non superare per evitare danni meccanici allo stelo della valvola. 36, 68

NoSQL Not Only SQL. Categoria di database non relazionali, progettati per gestire grandi quantità di dati non strutturati, offrendo flessibilità nello schema dei dati e scalabilità orizzontale. 22, 45, 47

PN La pressione nominale (PN) indica la massima pressione operativa per cui una valvola o un componente è progettato, secondo standard internazionali. È un parametro fondamentale per la selezione e la progettazione di apparecchiature industriali. [11]. 3

psi Pound per square inch. Unità di misura della pressione nel sistema imperiale anglosassone, corrispondente a una forza di una libbra applicata su un pollice quadrato. L'equivalente nel Sistema Internazionale (SI) è circa 6,895 kPa (chilopascal). 6

RTC Running to close (spring stroke). Coppia richiesta per mantenere in movimento la chiusura della valvola durante la corsa a molla. 21, 36, 68

RTO Running to open (air stroke). Coppia richiesta per mantenere in movimento l'apertura della valvola durante la corsa pneumatica. 21, 36, 68

SIL Il Safety Integrity Level (SIL) è una misura quantitativa della sicurezza funzionale di un sistema, definita dalla norma IEC 61508. Indica il livello di riduzione del rischio garantito da un sistema di sicurezza. Esistono 4 livelli SIL, numerati da 1 a 4, dove SIL 1 è il livello minimo e SIL 4 quello massimo. Ogni livello rappresenta

un intervallo di probabilità di guasto accettabile, con requisiti crescenti in termini di affidabilità, diagnostica e ridondanza. SIL 3, ad esempio, rappresenta un livello elevato di sicurezza, richiesto quando un guasto potrebbe avere gravi conseguenze per persone, ambiente o impianti. [12]. 3, 6

SOLID Insieme di principi di progettazione orientata agli oggetti: Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion. I principi SOLID favoriscono la manutenibilità, la scalabilità e la robustezza del codice. 22

SS316 Stainless Steel 316. Tipo di acciaio inossidabile, particolarmente resistente alla corrosione, utilizzato in applicazioni industriali critiche, incluse quelle chimiche e marine. 38

UML Unified Modeling Language. Linguaggio di modellazione standardizzato per la progettazione, la visualizzazione e la documentazione di sistemi software complessi. UML offre una serie di diagrammi (tra cui i diagrammi dei casi d'uso e delle classi) utili a rappresentare graficamente le diverse componenti di un sistema e le loro interazioni. 31

VCS Acronimo di Version Control System (VCS), un sistema per la gestione delle versioni di un progetto software. Esempio: Git è un VCS distribuito che permette di tracciare le modifiche al codice, mentre GitHub è una piattaforma che ospita repository Git, facilitando la collaborazione e l'integrazione con altri strumenti di sviluppo. 18

Glossario

Agile Alliance Organizzazione no-profit fondata nel 2001 con l'obiettivo di promuovere i principi e i valori dell'Agile Manifesto attraverso iniziative di educazione, eventi e ricerca. L'Agile Alliance si impegna a supportare la comunità Agile e a favorire l'adozione di pratiche di sviluppo software agili in tutto il mondo. 13

ASP.NET Core Framework di sviluppo open source per applicazioni web, parte della piattaforma .NET. Per approfondimenti si rimanda al capitolo sulle tecnologie. 20, 42

attuatore elettrico/pneumatico/idraulico/gas-over-oil Dispositivo che trasforma un segnale di comando (elettrico, pneumatico, idraulico o combinato gas-over-oil) in movimento meccanico, consentendo l'apertura, la chiusura o la regolazione di valvole e altri organi di controllo. Gli attuatori elettrici utilizzano motori elettrici, quelli pneumatici aria compressa, quelli idraulici fluidi in pressione, mentre i sistemi gas-over-oil combinano gas e olio per applicazioni in condizioni estreme. [2]. 4, 11

classe 2500 Classificazione secondo le norme ASNI/ASME che identifica la resistenza meccanica delle valvole in funzione della pressione e della temperatura di esercizio. 3, 6

coppia Nel contesto meccanico, la coppia è una grandezza fisica (momento torcente) che misura la tendenza di una forza a far ruotare un oggetto attorno a un asse. Si esprime in Newton-metro (Nm) nel Sistema Internazionale. 4, 7

Docker Piattaforma per la creazione, distribuzione e gestione di container applicativi. Per maggiori dettagli si rimanda al capitolo dedicato alle tecnologie. 20, 22, 42, 46

GitHub Piattaforma di versionamento e collaborazione per il codice sorgente, basata su Git. Si rimanda al capitolo dedicato alle tecnologie per ulteriori dettagli. 18, 22, 23

interfaccia web Componente software che consente l'interazione tra l'utente e un'applicazione tramite un browser internet, offrendo funzionalità di visualizzazione, inserimento e gestione dei dati in modalità remota o locale. 11

modello a cascata Un modello di sviluppo software lineare e sequenziale, in cui ogni fase del processo deve essere completata prima che inizi la successiva. Le fasi

tipiche includono analisi dei requisiti, progettazione, implementazione, verifica e manutenzione. È caratterizzato da una scarsa flessibilità ai cambiamenti durante il ciclo di vita del progetto. 12

MongoDB Database NoSQL orientato ai documenti, adatto alla gestione di dati non strutturati e dinamici. Si veda il capitolo dedicato alle tecnologie per ulteriori dettagli. 20–22, 24, 42, 45, 48, 53

MudBlazor Framework open source per la realizzazione di interfacce utente in Blazor, basato su Material Design. Per ulteriori dettagli si veda il capitolo dedicato alle tecnologie utilizzate. 20, 21, 23–25, 42, 62, 64

Oil & Gas Settore industriale che comprende l'esplorazione, l'estrazione, la raffinazione, il trasporto e la commercializzazione di petrolio greggio e gas naturale. È uno dei comparti più rilevanti a livello globale per impatto economico e tecnologico. 3, 4, 6, 11

OpenTelemetry Framework open source per la raccolta di dati di telemetria (tracing, metrics, logs) dalle applicazioni.. 22–24

product owner Figura chiave nel framework Agile/Scrum, responsabile della definizione e della gestione dei requisiti funzionali di un prodotto software, della prioritizzazione delle funzionalità e dell'interfaccia tra il team di sviluppo e gli stakeholder aziendali.[15] . 11

Rider IDE di JetBrains, specializzato per lo sviluppo .NET. Approfondimenti disponibili nel capitolo sulle tecnologie. 21, 24, 28, 46

Scalar Alternativa a Swagger per la documentazione delle API. Ulteriori approfondimenti nel capitolo sulle tecnologie. 23, 24

SwaggerUI Strumento per la documentazione e il testing interattivo delle API.. 22

valvola a sfera Dispositivo di intercettazione utilizzato negli impianti industriali per regolare o interrompere il flusso di un fluido. La chiusura è garantita da una sfera forata che ruota su un asse per aprire o chiudere il passaggio. Le valvole a sfera sono apprezzate per la loro affidabilità, rapidità di manovra e tenuta ermetica.[3]. 3–7, 11

Bibliografia

Riferimenti bibliografici

[6] John Doe. *A Test Book for Demonstration*. Test Publisher, 2023.

Siti web consultati

[1] *ATEX Directives - Wikipedia*. Consultato il 2025-05-07. 2025. URL: https://en.wikipedia.org/wiki/ATEX_directives (cit. a p. 77).

[2] *Attuatore - Wikipedia*. Consultato il 2025-05-07. 2025. URL: https://it.wikipedia.org/wiki/Attuatore (cit. a p. 80).

[3] *Ball Valve - Wikipedia*. Consultato il 2025-05-06. 2025. URL: https://en.wikipedia.org/wiki/Ball_valve (cit. a p. 81).

[4] Riccardo Cardin. *Hexagonal Architecture Example*. Consultato il 2025-06-20. 2023. URL: https://github.com/rcardin/hexagonal (cit. a p. 47).

[5] Alistair Cockburn. *Hexagonal Architecture*. Consultato il 2025-06-20. 2005. URL: https://alistair.cockburn.us/hexagonal-architecture/.

[7] *Factory Acceptance Test - TÜV Italia*. Consultato il 2025-05-07. 2025. URL: https://www.tuv.com/italy/it/factory-acceptance-test.html (cit. a p. 78).

[8] *Hexagonal architecture (software) - Wikipedia*. Consultato il 2025-06-20. 2024. URL: https://en.wikipedia.org/wiki/Hexagonal_architecture_(software).

[9] *Manifesto for Agile Software Development*. Sito ufficiale del Manifesto Agile. Consultato il 2025-05-15. 2025. URL: https://agilemanifesto.org/ (cit. a p. 12).

[10] *Nuvem Srl - Home*. Consultato il 2025-05-07. 2025. URL: https://nuvemsrl.it/ (cit. a p. 1).

[11] *Pressione Nominale - Wikipedia*. Consultato il 2025-05-07. 2025. URL: https://it.wikipedia.org/wiki/Pressione_nominale (cit. a p. 78).

[12] *Safety Integrity Level - Wikipedia*. Consultato il 2025-05-07. 2025. URL: https://it.wikipedia.org/wiki/Safety_Integrity_Level (cit. a p. 79).

[13] *Scrum (software development) - Wikipedia*. Consultato il 2025-05-15. 2025. URL: https://en.wikipedia.org/wiki/Scrum_(software_development) (cit. a p. 75).

[14] *Starline - Home*. Consultato il 2025-05-07. 2025. URL: https://starline.it/ (cit. a p. 3).

[15] *What is a Product Owner? - Scrum.org*. Consultato il 2025-05-07. 2025. URL: https://www.scrum.org/resources/what-is-a-product-owner (cit. a p. 81).