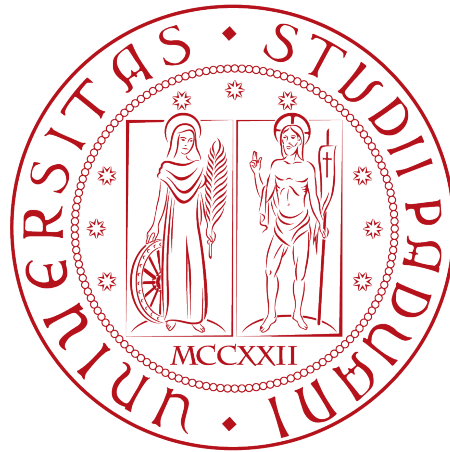


**Università degli Studi di Padova**

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**Automazione del Workflow tra Microsoft  
Planner e Taiga: una soluzione di  
sincronizzazione dinamica**

*Tesi di laurea*

*Relatore*

Prof.ssa Ombretta Gaggi

*Laureando*

Giacomo Gualato

*Matricola* 2043953

---

ANNO ACCADEMICO 2023-2024

Giacomo Gualato: *Automazione del Workflow tra Microsoft Planner e Taiga: una soluzione di sincronizzazione dinamica*, Tesi di laurea, © Dicembre 2024.

# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di *stage*, della durata complessiva di circa trecentoventi ore, dal laureando Giacomo Gualato presso l'azienda Wintech S.p.A.

Gli obiettivi principali del progetto includevano:

- Analisi approfondita delle piattaforme di pianificazione aziendale *Microsoft Planner* e *Taiga*.
- Studio delle modalità d'uso e delle metodologie di lavoro adottate dagli utenti su ciascuna delle due piattaforme.
- Approfondimento delle [Application Program Interface \(API\)](#) di *Taiga* e *Microsoft Graph*, con particolare riferimento agli strumenti della *suite Office 365*, inclusi *Planner* e *SharePoint*, per comprenderne le possibilità di integrazione ed interazione.
- Sviluppo di un sistema di sincronizzazione automatizzata lato [server-to-server](#) tra *Microsoft Planner* e *Taiga*, con l'obiettivo di migliorare l'efficienza operativa e garantire la coerenza dei dati, rimuovendo i possibili errori umani.
- Integrazione del sistema all'interno dei processi aziendali esistenti, con l'obiettivo di facilitare la gestione coordinata dei progetti e delle risorse aziendali.

“Il computer non è una macchina intelligente che aiuta le persone stupide, anzi è una macchina stupida che funziona solo nelle mani delle persone intelligenti.”

— Umberto Eco

## Ringraziamenti

*Desidero innanzitutto esprimere la mia sincera gratitudine alla Prof.ssa Ombretta Gaggi, relatrice della presente tesi, per il supporto e la guida fornitomi durante la fase di stesura di questo lavoro.*

*Un ringraziamento speciale va all'azienda Wintech S.p.A. e, in particolare, al mio tutor aziendale, Enrico Merigliano, per il sostegno costante e la disponibilità dimostrata durante il mio percorso. L'ambiente di lavoro offerto da Wintech, caratterizzato da un clima collaborativo e familiare, ha rappresentato un prezioso contesto di crescita e formazione.*

*Un ringraziamento speciale va alla mia compagna, l'amore della mia vita, per la forza, il sostegno e l'ispirazione che mi ha dato in ogni momento. La sua presenza costante e il suo amore sono stati per me un punto di riferimento fondamentale in questo percorso.*

*Un ringraziamento sentito va anche ai miei genitori, per il supporto incondizionato, la pazienza e la costante vicinanza in ogni fase del mio percorso di studi. La loro presenza è stata essenziale per il raggiungimento di questo traguardo.*

*Infine, rivolgo un pensiero affettuoso ai miei amici, con cui ho condiviso esperienze indimenticabili e avventure che hanno arricchito questi anni di studio.*

Padova, Dicembre 2024

Giacomo Gualato

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Descrizione del progetto	1
1.2	L'azienda	2
1.2.1	Certificazioni Aziendali	3
1.2.2	Sviluppo Sicuro del Software	3
1.3	Organizzazione del documento	4
<b>2</b>	<b>Il progetto di stage</b>	<b>5</b>
2.1	La necessità	5
2.2	Processo Aziendale di Utilizzo di Microsoft Planner e Taiga	6
2.2.1	Strutturazione dei progetti su Microsoft Planner	6
2.2.2	Transizione delle attività da Planner a Taiga	7
2.2.3	Ciclo di Vita delle user story in Taiga	8
2.2.4	Gestione delle issue in Taiga	9
2.2.5	Organizzazione dei bucket in Planner e Stati Associati in Taiga	10
2.3	Competenze previste	11
2.4	Obiettivi	12
2.4.1	Obiettivi Obbligatori	12
2.4.2	Obiettivi Desiderabili	12
2.4.3	Obiettivi Facoltativi	12
<b>3</b>	<b>Tecnologie utilizzate</b>	<b>13</b>
3.1	Microsoft Office 365	13
3.1.1	Teams	14
3.1.2	Outlook	14
3.1.3	OneDrive	15
3.1.4	SharePoint	17
3.1.5	Microsoft Power Automate	18
3.1.6	Microsoft Azure	19
3.1.7	Microsoft Graph	19
3.2	Taiga	20
3.2.1	Caratteristiche principali di Taiga	20
3.3	Postman	21
3.3.1	Analisi delle Risposte JSON	21
3.4	Java	22
3.5	IDE: IntelliJ IDEA (versione Community)	22
3.6	Spring Boot e Thymeleaf	22
3.6.1	Moduli di Spring Boot Utilizzati	22

3.7	Apache Maven . . . . .	23
3.7.1	Gestione delle Dipendenze . . . . .	23
3.7.2	Configurazione e Automazione . . . . .	23
3.8	Git . . . . .	24
3.8.1	Gestione del Codice con Git . . . . .	24
<b>4</b>	<b>Analisi dei requisiti</b>	<b>25</b>
4.1	Requirements baseline . . . . .	25
4.1.1	Requisiti Funzionali . . . . .	26
4.1.2	Requisiti di Qualità . . . . .	33
4.1.3	Requisiti di Vincolo . . . . .	35
4.2	Tracciamento dei requisiti . . . . .	36
<b>5</b>	<b>Progettazione e Codifica</b>	<b>38</b>
5.1	Autenticazione e Autorizzazione con Azure . . . . .	38
5.1.1	Registrazione dell'Applicazione . . . . .	39
5.1.2	Configurazione delle Credenziali e Dettagli dell'Applicazione . . . . .	40
5.1.3	Autorizzazioni per l'Accesso alle API . . . . .	41
5.2	Servizio di Autenticazione con OAuth 2.0 . . . . .	42
5.2.1	Panoramica del Protocollo OAuth 2.0 . . . . .	42
5.2.2	Processo di Ottenimento del Token Applicativo . . . . .	42
5.2.3	Uso del Token per le Richieste alle API di Microsoft . . . . .	45
5.2.4	Possibili Risposte di Errore delle API . . . . .	46
5.3	Autenticazione tramite API di Taiga . . . . .	47
5.3.1	Flusso di Autenticazione . . . . .	47
5.3.2	Uso del Token per le Chiamate API . . . . .	50
5.4	Approccio Standard per le Chiamate API . . . . .	50
5.5	Rilevamento delle Modifiche e Sincronizzazione dei Dati . . . . .	51
5.5.1	Configurazione della Funzione Periodica . . . . .	51
5.6	Recupero dei Progetti da Taiga . . . . .	53
5.6.1	Flusso di Recupero . . . . .	53
5.6.2	Recupero dell'ID Utente tecnico . . . . .	53
5.6.3	Recupero dei Progetti . . . . .	55
5.6.4	Ottenimento delle User Stories e delle Issues . . . . .	56
5.6.5	Estrazione delle Severity di un Progetto . . . . .	58
5.7	Determinazione della Modalità di Aggiornamento . . . . .	60
5.7.1	Associazione tra Piani e Gruppi in Microsoft Planner . . . . .	60
5.7.2	Modalità di Aggiornamento Completa . . . . .	60
5.7.3	Modalità di Aggiornamento Parziale . . . . .	60
5.8	Recupero dei Piani da Microsoft Planner . . . . .	62
5.8.1	Panoramica del Processo . . . . .	62
5.8.2	Recupero dei Piani per Tutti i Gruppi . . . . .	62
5.8.3	Recupero dei Piani di un Singolo Gruppo . . . . .	62
5.8.4	Estrazione dei Piani dalla Risposta API . . . . .	62
5.8.5	Dettagli Recuperati per ciascun Piano . . . . .	63
5.8.6	Recupero delle Categorie/Tag di un Piano . . . . .	64
5.8.7	Recupero dei Bucket di un Piano . . . . .	65
5.9	Confronto dei Dati . . . . .	66
5.9.1	Confronto delle <i>Regular Tasks</i> . . . . .	66
5.9.2	Confronto delle <i>Bug Tasks</i> . . . . .	69

5.9.3	Confronto delle User Stories . . . . .	71
5.10	Sistema di Log e Gestione dei File . . . . .	72
5.10.1	Creazione del Sito e della Cartella per i Log . . . . .	72
5.10.2	Generazione e Struttura dei File di Log . . . . .	74
5.10.3	Estrazione delle Informazioni dai Sistemi . . . . .	75
5.11	Gestione tramite interfaccia web . . . . .	78
5.11.1	Pagina di selezione piano e progetto . . . . .	78
5.11.2	Pagina Personalizza Gruppi . . . . .	80
5.11.3	Pagina Gestisci Associazioni . . . . .	82
5.11.4	Pagina Configurazione Collegamenti (UserStory) . . . . .	83
5.11.5	Pagina Configurazione Collegamenti (Issue) . . . . .	85
<b>6</b>	<b>Integrazione Aziendale</b>	<b>86</b>
6.1	Soluzione 1: Polling . . . . .	86
6.1.1	Descrizione della Soluzione . . . . .	86
6.1.2	Considerazioni Tecniche . . . . .	86
6.1.3	Analisi della Fattibilità . . . . .	87
6.2	Soluzione 2: Utilizzo dei Webhook . . . . .	87
6.2.1	Descrizione della Soluzione . . . . .	87
6.2.2	Problemi Operativi . . . . .	87
6.3	Possibile Alternativa: Migrazione a SharePoint . . . . .	88
6.3.1	Descrizione della Soluzione . . . . .	88
6.3.2	Conclusioni sulla Fattibilità . . . . .	88
6.4	Soluzione Finale . . . . .	88
6.4.1	Descrizione della Soluzione . . . . .	88
6.4.2	Realizzazione della Soluzione . . . . .	89
6.4.3	Sincronizzazione da Planner a Taiga . . . . .	89
6.4.4	Sincronizzazione da Taiga a Planner . . . . .	91
6.4.5	Conclusioni sulla Sincronizzazione . . . . .	92
<b>7</b>	<b>Conclusioni</b>	<b>93</b>
7.1	Raggiungimento degli Obiettivi . . . . .	93
7.2	Conoscenze Acquisite . . . . .	94
7.3	Esperienza Professionale e Formazione Universitaria . . . . .	94
7.4	Analisi Temporale dello Stage . . . . .	94
7.5	Conclusioni Personali . . . . .	94
	<b>Acronimi e abbreviazioni</b>	<b>96</b>
	<b>Glossario</b>	<b>97</b>
	<b>Bibliografia</b>	<b>102</b>

# Elenco delle figure

1.1	Logo ufficiale di Wintech S.p.A. . . . . .	2
2.1	Prima parte dei <i>bucket</i> del ciclo di vita delle <i>user story</i> in <i>Taiga</i> . . . . .	6
2.2	Seconda parte dei <i>bucket</i> del ciclo di vita delle <i>user story</i> in <i>Taiga</i> . . . . .	6
2.3	Esempio di creazione di una <i>user story</i> Analizzata e Stimata in <i>Taiga</i> . . . . .	7
2.4	Stati di una <i>user story</i> in <i>Taiga</i> . . . . .	8
2.5	Stati di una <i>Issue</i> in <i>Taiga</i> . . . . .	9
3.1	Logo <i>Microsoft Office 365</i> . . . . .	13
3.2	Logo <i>Microsoft Teams</i> . . . . .	14
3.3	Logo di <i>Microsoft Outlook</i> . . . . .	14
3.4	Logo <i>Microsoft OneDrive</i> . . . . .	15
3.5	Logo <i>Microsoft Planner</i> . . . . .	15
3.6	Caratteristiche specifiche di una <i>task</i> . . . . .	16
3.7	Logo <i>Microsoft SharePoint</i> . . . . .	17
3.8	Logo <i>Microsoft Power Automate</i> . . . . .	18
3.9	Logo <i>Microsoft Azure</i> . . . . .	19
3.10	Logo <i>Microsoft Graph</i> . . . . .	19
3.11	Logo <i>Taiga</i> . . . . .	20
3.12	Logo <i>Postman</i> . . . . .	21
3.13	Logo <i>IntelliJ IDEA</i> . . . . .	22
3.14	Logo <i>Apache Maven</i> . . . . .	23
3.15	Logo <i>Git</i> . . . . .	24
5.1	Creazione di un'applicazione personalizzata su <i>Azure Active Directory</i> . . . . .	39
5.2	Panoramica dell'applicazione registrata su <i>Azure</i> . . . . .	40
5.3	Scelta tra autorizzazioni delegate e autorizzazioni applicazione . . . . .	41
5.4	Autorizzazioni scelte per l'applicazione registrata . . . . .	41
5.5	Richiesta di <i>token microsoft</i> . . . . .	43
5.6	Risposta alla richiesta di <i>token</i> . . . . .	43
5.7	Esempio di risposta da un'API <i>Microsoft</i> in caso di <i>token</i> non valido o scaduto . . . . .	45
5.8	Esempio di chiamata POST per ottenere il token di autenticazione . . . . .	47
5.9	Esempio di risposta alla richiesta di autenticazione, contenente il <code>auth_token</code> . . . . .	48
5.10	Risposta dell' <i>endpoint</i> per ottenere l' <i>ID</i> utente dalla documentazione di <i>Taiga</i> . . . . .	54
5.11	Classe <i>template</i> condivisa per <i>User Story</i> e <i>Issue</i> in <i>Taiga</i> . . . . .	56
5.12	Struttura completa della risposta JSON contenente le <i>severity</i> del progetto. . . . .	59



5.13	Classe <i>Task</i> . . . . .	63
5.14	Esempio di una <i>Regular Task</i> spostata nel <i>bucket</i> "In Analisi [Sviluppatori]". . . . .	67
5.15	Creazione della <i>user story</i> corrispondente alla <i>RegularTask</i> nel progetto Taiga. . . . .	67
5.16	Scelta del tipo di sito: Sito del Team. . . . .	73
5.17	Creazione della cartella <i>logfiles</i> . . . . .	73
5.18	Esempio di file di log archiviato in SharePoint. . . . .	74
5.19	Discussione sulla mancanza di una cronologia delle <i>Task</i> nelle API di Planner. . . . .	76
5.20	Interfaccia grafica per la cronologia delle <i>Task</i> su Planner. . . . .	77
5.21	Pagina di selezione Piano e Progetto. . . . .	79
5.22	Riepilogo delle associazioni configurate tra Planner e Taiga. . . . .	79
5.23	Pagina di Personalizzazione dei <i>GroupID</i> . . . . .	80
5.24	Tabella dei <i>GroupID</i> esistenti con opzione di eliminazione. . . . .	81
5.25	Pagina Gestisci Associazioni. . . . .	82
5.26	Pagina di configurazione per collegamenti personalizzati tra bucket e stati delle <i>user story</i> . . . . .	83
6.1	Pagina di configurazione per collegamenti personalizzati tra bucket e stati delle <i>user story</i> . . . . .	88
6.2	Trigger Planner disponibili in <i>Power Automate</i> . . . . .	89
6.3	Schema del flusso Planner-Taiga in <i>Power Automate</i> . . . . .	90
6.4	<i>URL</i> generato dal <i>trigger HTTP</i> in <i>Power Automate</i> . . . . .	91
6.5	Errore di <i>Taiga</i> durante l'uso di un <i>URL</i> abbreviato. . . . .	92

## Elenco delle tabelle

2.1	Scala di conversione per le stime di tempo basata sui punti. . . . .	7
2.2	Competenze previste durante il progetto formativo . . . . .	11
4.1	Requisiti funzionali . . . . .	37
4.2	Requisiti qualitativi . . . . .	37
4.3	Requisiti di vincolo . . . . .	37

# Capitolo 1

## Introduzione

### 1.1 Descrizione del progetto

Il progetto da me scelto si propone di analizzare e integrare i sistemi di pianificazione di progetto utilizzati da Wintech, in particolare *Microsoft Planner* e *Taiga*.

*Microsoft Planner* è uno strumento di gestione delle attività che si distingue per un'interfaccia intuitiva e la sua integrazione con la *suite Microsoft 365*, rendendolo ideale per la gestione collaborativa delle attività aziendali.

La struttura di *Planner* si basa sull'organizzazione di attività in *task* e *bucket* che semplificano la gestione e il monitoraggio di progetti interni.

Tuttavia, *Planner* non è pensato per scenari complessi di [Project Management](#), dove spesso è necessario un sistema di tracciamento avanzato per il ciclo di vita dei progetti.

*Taiga*, invece, è una piattaforma [open-source](#) orientata al *projectmanagement* per *team* di sviluppo, che supporta metodologie [agile](#) come [Scrum](#) e [Kanban](#).

Rispetto a *Planner*, offre funzionalità avanzate, come la gestione del *backlog*, la pianificazione degli *sprint* e l'assegnazione di *user story*.

Questi strumenti, pur essendo efficaci singolarmente, presentano delle limitazioni quando utilizzati in modo congiunto, specialmente in termini di sincronizzazione e coerenza delle informazioni.

La mancanza di integrazione diretta tra *Planner* e *Taiga* costringe i *team* a inserire manualmente le stesse informazioni nei due sistemi, aumentando così il rischio di errore e la possibilità di discrepanze nei dati.

L'obiettivo principale del progetto è verificare l'utilizzo delle *API* disponibili e esplorare la possibilità di implementarle nell'infrastruttura aziendale, al fine di realizzare una [Proof of Concept \(PoC\)](#) che consenta di mantenere i dati sincronizzati tra i due strumenti.

Nello specifico, la mia *PoC* permette di rimanere in ascolto delle modifiche effettuate su *Planner* e ricevere aggiornamenti in tempo reale, elaborando i dati per replicare le stesse modifiche su *Taiga*, e viceversa.

Questo consente di sincronizzare le attività tra i due sistemi, assicurando che determinate modifiche effettuate in uno siano automaticamente riflesse anche nell'altro.

Per garantire una gestione chiara e ordinata, la *PoC* consente l'associazione 1:1 tra un piano di *Planner* e un progetto di *Taiga*.

Questa integrazione consentirebbe non solo di ottimizzare il processo di pianificazione, ma anche di ridurre significativamente gli errori legati all'inserimento manuale

dei dati, che possono compromettere l'efficacia del progetto.

Inoltre, il progetto mira a garantire una comunicazione più fluida tra i vari *team* coinvolti, facilitando il passaggio delle informazioni e migliorando la trasparenza nel processo di gestione dei progetti.

## 1.2 L'azienda



**Figura 1.1:** Logo ufficiale di Wintech S.p.A.

Winning Technologies, nota anche come Wintech S.p.A., è una società fondata nel 1987 da Massimo Gallotta, attuale [Chief Executive Officer \(CEO\)](#), concepita come [System Integrator](#) con sede principale a Padova, oltre a filiali a Milano, Bassano del Grappa e Pordenone.

La figura 1.1 mostra il logo ufficiale dell'azienda, che rappresenta l'identità visiva e professionale di Wintech.

Da oltre tre decenni, Wintech opera nel settore [Information and Communication Technology \(ICT\)](#), offrendo consulenza e soluzioni tecnologiche su misura per ottimizzare e innovare i processi aziendali.

Wintech si caratterizza per un ampio portafoglio di servizi che include consulenza, progettazione e implementazione di soluzioni tecnologiche personalizzate, e distribuzione di prodotti *ICT*. L'azienda si rivolge a una clientela diversificata, composta da [Piccole e Medie Imprese \(PMI\)](#), grandi aziende, studi professionali, istituti bancari e assicurativi, offrendo competenze mirate e risposte specifiche alle esigenze di ciascun settore.

Le aree di eccellenza di Wintech comprendono la *Social Business Collaboration*, la *Business Intelligence*, il *Cash & Credit Management*, la Gestione Documentale, i Sistemi [Enterprise Resource Planning \(ERP\)](#), e le Infrastrutture di Rete. Inoltre, Wintech fornisce soluzioni di [Outsourcing](#) e *System Integrator*, adattate a contesti aziendali variegati, garantendo costantemente qualità e innovazione.

Wintech si avvale inoltre di collaborazioni tecniche di alto profilo con nomi di rilievo come *IBM*, *Microsoft*, *HP*, *VMware*, *Nutanix*, *Veeam* e *HPE Aruba Networking*.

### 1.2.1 Certificazioni Aziendali

Wintech si impegna costantemente per garantire elevati *standard* di qualità e sicurezza, come dimostrato dalle certificazioni riconosciute a livello internazionale che ha ottenuto. Queste certificazioni riflettono l'attenzione dell'azienda nel rispettare i più rigorosi requisiti di efficienza, gestione del rischio e sicurezza delle informazioni, offrendo ai clienti un elevato grado di affidabilità.

#### Certificazione UNI EN ISO 9001:2015

Wintech ha conseguito la certificazione ISO 9001 il 24 agosto 2007 per il proprio sistema di gestione della qualità, aggiornandola successivamente alla versione ISO 9001:2015. Tale *standard* internazionale assicura che i processi aziendali siano strutturati per raggiungere un'efficace soddisfazione del cliente.

#### Certificazione UNI CEI EN ISO/IEC 27001:2017

A partire dal 22 maggio 2017, Wintech ha ottenuto la certificazione ISO/IEC 27001:2017, specifica per la gestione della sicurezza delle informazioni. Tale *standard* rappresenta un punto di riferimento fondamentale per le organizzazioni che desiderano garantire la protezione dei dati attraverso un sistema di gestione certificabile, consolidando la fiducia dei clienti e degli *stakeholder*.

### 1.2.2 Sviluppo Sicuro del Software

Wintech si distingue per l'elevata competenza dei suoi collaboratori, supportata dalle certificazioni ottenute, che testimoniano l'impegno dell'azienda nel garantire la massima qualità professionale e una gestione rigorosa della sicurezza.

L'approccio alla sicurezza è integrato in ogni fase di sviluppo e gestione dei progetti, con linee guida interne e pratiche sicure accessibili a tutto il personale.

Le linee guida aziendali comprendono direttive specifiche per uno sviluppo sicuro e sono organizzate per assicurare che ogni fase del ciclo di vita del *software* ([Secure Software Development Life Cycle \(SSDLC\)](#)) integri misure di sicurezza adeguate.

Tali direttive riguardano aspetti fondamentali quali l'analisi e l'implementazione delle migliori pratiche per la sicurezza, il controllo qualità del codice e la gestione degli strumenti di sicurezza più avanzati. La sicurezza viene quindi inserita sin dalle prime fasi di progettazione, per garantire che ogni progetto rispetti rigorosi *standard* di affidabilità e protezione dei dati.

Per prevenire vulnerabilità, Wintech promuove una formazione continua del personale. Questo approccio è rafforzato da un sistema di aggiornamenti interni basato su *SharePoint*, dove vengono condivise notizie, innovazioni tecnologiche e risorse per supportare un costante allineamento con le nuove tecnologie.

## 1.3 Organizzazione del documento

Il documento è strutturato come segue:

**Capitolo 1** Introduce il progetto realizzato e successivamente presenta l'azienda, descrivendone i prodotti, i servizi, le certificazioni e la metodologia di sviluppo adottata.

**Capitolo 2** descrive il progetto di *stage*, trattandone origine, obiettivi e vincoli.

**Capitolo 3** analizza lo studio delle tecnologie adottate durante lo *stage*.

**Capitolo 4** esamina i requisiti del progetto, con particolare attenzione alla *PoC* realizzata.

**Capitolo 5** illustra la progettazione, lo sviluppo e le attività di verifica e validazione delle soluzioni.

**Capitolo 6** descrive i metodi di integrazione aziendale dell'applicazione implementata.

**Capitolo 7** presenta le conclusioni e le prospettive future, includendo un'analisi personale sull'esperienza professionale rispetto alla formazione universitaria.

Nel testo sono adottate le seguenti convenzioni tipografiche:

- I termini tecnici sono definiti nel Glossario a fine documento.
- Gli acronimi e le abbreviazioni sono elencati nella sezione Acronimi e Abbreviazioni.
- I termini in lingua straniera sono scritti in *corsivo*.
- Immagini e tabelle includono didascalie e numerazioni che richiamano il capitolo di appartenenza.

## Capitolo 2

# Il progetto di stage

### 2.1 La necessità

In un contesto aziendale altamente competitivo e in rapida evoluzione, la gestione ottimale dei progetti è fondamentale per garantire l'efficienza operativa e il successo a lungo termine.

La pianificazione accurata delle attività e la sincronizzazione delle informazioni tra i vari *team* rappresentano requisiti essenziali per evitare ritardi, errori e disallineamenti nella comunicazione.

Per soddisfare queste esigenze, *Wintech* si è posta l'obiettivo di individuare nuove soluzioni per migliorare il proprio processo di pianificazione, sfruttando tecnologie che possano ridurre la dipendenza da processi manuali, soggetti ad errori.

In questo contesto, il secondo progetto proposto da *Wintech* nasce dall'esigenza di verificare la possibilità di ottimizzare e automatizzare il processo di pianificazione aziendale, riducendo al minimo il rischio di errori legati all'inserimento manuale dei dati o eventuali omissioni.

Attualmente, nella fase di pianificazione, *Wintech* utilizza due strumenti distinti per gestire ogni progetto: *Microsoft Planner* e *Taiga*.

Ogni progetto aziendale è associato sia a un piano in *Planner* sia a un progetto in *Taiga*.

L'uso di entrambi i sistemi risponde a esigenze specifiche di comunicazione: *Planner* viene utilizzato per la condivisione delle informazioni con tutti gli *stakeholder*, interni ed esterni all'azienda, inclusi i clienti, offrendo una visione ad alto livello del progetto.

Questo strumento permette a tutti i membri di accedere facilmente a una panoramica delle attività, grazie a un linguaggio chiaro e accessibile.

Quando le attività di *Planner* raggiungono una fase più avanzata e vengono assegnate al *team* di sviluppo, esse vengono trasferite su *Taiga*, uno strumento destinato esclusivamente all'uso interno per i soli sviluppatori.

In *Taiga*, le attività vengono dettagliate tramite *user story* che corrispondono alle singole *task* inserite in *Planner*, ma con una strutturazione molto più tecnica e approfondita. Ogni *user story* in *Taiga* può contenere, a sua volta, numerose attività specifiche, come la gestione di funzionalità dettagliate (ad esempio, *login*, *logout*) e descrizioni tecniche su come implementarle. Questo passaggio consente al *team* di sviluppo di operare con una documentazione completa e accurata, basata su un linguaggio specialistico, necessario per affrontare le esigenze tecniche del progetto in modo efficace.

## 2.2. PROCESSO AZIENDALE DI UTILIZZO DI MICROSOFT PLANNER E TAIGA6

Tuttavia, l'attuale processo di trasferimento delle informazioni da *Planner* a *Taiga* è manuale e suscettibile a errori, come trascrizioni errate e omissioni, che possono compromettere l'efficacia della pianificazione e la qualità complessiva della gestione del progetto.

Per questo motivo, Wintech ha sentito la necessità di esplorare una soluzione automatizzata, basata su un'integrazione *server-to-server* tra *Planner* e *Taiga*, che garantisca la sincronizzazione automatica delle informazioni e riduca l'intervento umano.

Il progetto, quindi, mira a sviluppare un sistema in grado di gestire automaticamente la creazione, l'aggiornamento e la sincronizzazione dei dati tra i due strumenti, rendendo il flusso di lavoro più efficiente e preciso. Tale automazione consentirebbe a Wintech di risparmiare tempo e ridurre significativamente gli errori, migliorando l'affidabilità e la qualità del processo di pianificazione.

## 2.2 Processo Aziendale di Utilizzo di Microsoft Planner e Taiga

Per gestire i progetti in modo organizzato ed efficiente, Wintech ha adottato un flusso di lavoro che integra *Microsoft Planner* e *Taiga*, seguendo ogni progetto dall'ideazione fino alla fase di produzione.

Questo processo l'ho personalizzato dopo diverse riunioni e discussioni iniziali con il mio *tutor* aziendale, con l'obiettivo di strutturare un sistema che permettesse di sincronizzare aggiornamenti tra il *team* di sviluppo e *stakeholder*, garantendo monitoraggio continuo e trasparenza lungo tutto il ciclo di sviluppo.

Il flusso di lavoro aziendale è quindi organizzato in fasi ben definite, ciascuna delle quali ha una funzione specifica in *Planner* e in *Taiga*, come descritto nei paragrafi seguenti.

### 2.2.1 Strutturazione dei progetti su Microsoft Planner

In *Microsoft Planner*, ogni progetto è organizzato in *bucket*, ognuno dei quali rappresenta una fase chiave del ciclo di vita del progetto.

La struttura dei *bucket* utilizzata per rappresentare il ciclo di vita è illustrata nella Figura 2.1 e nella Figura 2.2.

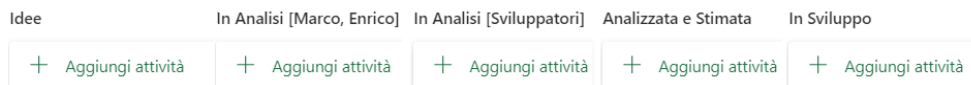


Figura 2.1: Prima parte dei *bucket* del ciclo di vita delle *user story* in *Taiga*.

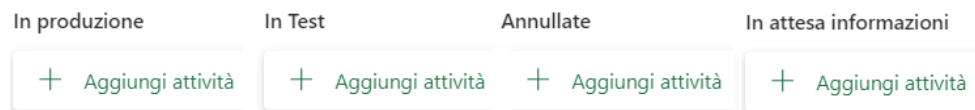


Figura 2.2: Seconda parte dei *bucket* del ciclo di vita delle *user story* in *Taiga*.

## 2.2. PROCESSO AZIENDALE DI UTILIZZO DI MICROSOFT PLANNER E TAIGA7

Questa struttura a *bucket* garantisce che tutte le attività siano ordinate e facilmente tracciabili nelle diverse fasi di sviluppo:

- **Idee** - Raccoglie nuove proposte di requisiti e funzionalità, provenienti sia da *stakeholder* interni che esterni, e rappresenta l'inizio del ciclo di sviluppo.
- **In Analisi [Stakeholder]** - Qui le idee vengono prese in carico da *stakeholder* interni come Marco ed Enrico, che avviano una discussione collaborativa per verificarne la fattibilità, scambiandosi opinioni e dettagli tramite la *chat* interna.
- **In Analisi [Sviluppatori]** - Le idee approvate dagli *stakeholder* vengono trasferite al *team* di sviluppo, che realizza una revisione tecnica dettagliata, integrando risorse e documenti tecnici per facilitare l'implementazione.

### 2.2.2 Transizione delle attività da Planner a Taiga

Quando una *task* viene spostata nel *bucket* "In Analisi [Sviluppatori]" in *Planner*, essa viene trasferita in *Taiga* come *User Story*.

Questo trasferimento consiste nel copiare la *task* da *Planner* al *backlog* del progetto corrispondente in *Taiga* (cioè il progetto collegato al piano di *Planner* in cui si è verificato lo spostamento della *task* nel *bucket* "In Analisi [Sviluppatori]").

Una volta creata la *User Story* in *Taiga* (come illustrato in Figura 2.3), viene stimato il tempo necessario per il completamento utilizzando un sistema di punti.

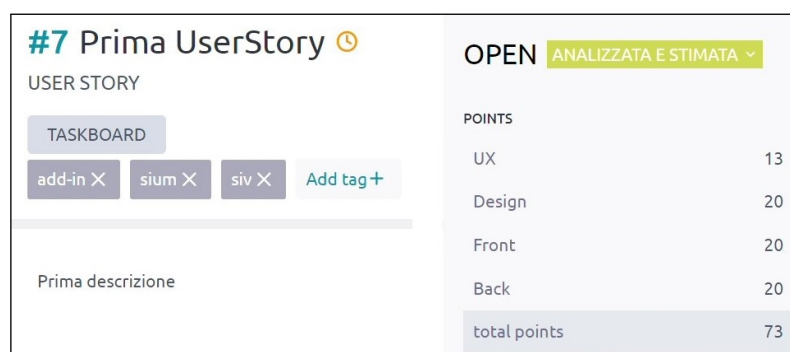


Figura 2.3: Esempio di creazione di una *user story* Analizzata e Stimata in *Taiga*.

Ogni punto rappresenta un'unità di lavoro specifica, come descritto nella Tabella 2.1.

Punti	Tempo di lavoro	Unità
1 punto	1 ora	<i>h</i> (hour)
8 ore	1 giorno	<i>d</i> (day)
40 ore	1 settimana	<i>w</i> (week)

Tabella 2.1: Scala di conversione per le stime di tempo basata sui punti.

Dopo la stima, la *user story* cambia stato da "New" a "Analizzata e Stimata". Parallelamente, la *task* corrispondente in *Planner* viene spostata nel *bucket* "Analizzata e Stimata" e rinominata con un prefisso che indica la stima del tempo convertita, come indicato nella Tabella 2.1.



## 2.2. PROCESSO AZIENDALE DI UTILIZZO DI MICROSOFT PLANNER E TAIGAS

Ad esempio, se la *user story* ha una stima di 50 punti, il prefisso sarà *[2h 1d 1w]*, e il titolo originale della *task* “Titolo 1” diventerà *[2h 1d 1w] Titolo 1*.

### 2.2.3 Ciclo di Vita delle user story in Taiga

Ogni *user story* passa attraverso una serie di stati che ne riflettono l’avanzamento, come illustrato in Figura 2.4.











USER STORY STATUSES	
Color	Name
	New
	Analizzata e Stimata
	Ready
	In progress
	Ready for test
	Failed Test
	Done
	Archived
	Needs info
	Rejected

Figura 2.4: Stati di una *user story* in Taiga.

Ogni modifica di stato in *Taiga* viene replicata in *Planner* spostando la corrispondente *task* tra i *bucket*, garantendo coerenza e tracciabilità.

- **New** - *user story* appena creata in *Taiga*.
- **Analizzata e Stimata** - Completata la fase di analisi e stimata la durata, l’attività è spostata nel *bucket* “Analizzata e Stimata” in *Planner*.
- **Ready** - *user story* pronta per l’inizio del lavoro, che avverrà all’interno di uno *sprint*.
- **In Progress** - Con l’inizio dello sviluppo, la *user story* passa a “*In Progress*” in *Taiga*, e l’attività corrispondente è spostata nel *bucket* “In Sviluppo” in *Planner*.
- **Ready for Test** - Completata la fase di sviluppo, la *user story* entra in fase di *test*, e l’attività corrispondente passa al *bucket* “In Test” in *Planner*.
- **Done** - Dopo il completamento della fase di *test*, la *User Story* è considerata completata e spostata nel *bucket* “Produzione” in *Planner*.

## 2.2. PROCESSO AZIENDALE DI UTILIZZO DI MICROSOFT PLANNER E TAIGA9

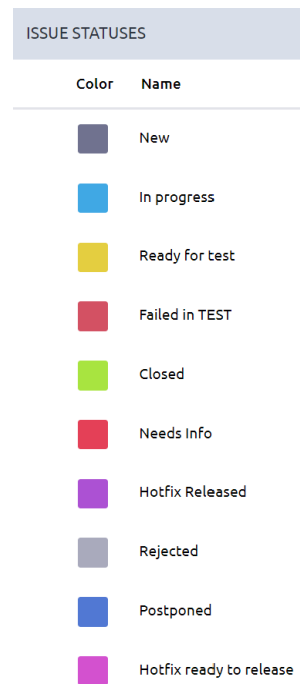
Stati aggiuntivi includono "Needs Info" (informazioni aggiuntive richieste) e "Rejected" (annullamento), che si riflettono rispettivamente nei *bucket* "In Attesa Informazioni" e "Annullate" su *Planner*.

### 2.2.4 Gestione delle issue in Taiga

A differenza delle *user story*, le *issue* rappresentano problemi o errori e non vengono stimate con il sistema a punti.

Le *issue* vengono create manualmente in *Taiga* solo quando in *Planner* viene creata una *task* con il tag "Bug".

Anche le *issue* seguono uno specifico ciclo di avanzamento (illustrato in Figura 2.5):



ISSUE STATUSES	
Color	Name
■	New
■	In progress
■	Ready for test
■	Failed in TEST
■	Closed
■	Needs Info
■	Hotfix Released
■	Rejected
■	Postponed
■	Hotfix ready to release

Figura 2.5: Stati di una *Issue* in Taiga.

- **New** - *issue* appena creata in *Taiga*.
- **In Progress** - Con l'inizio dello sviluppo, lo stato dell'*issue* cambia a "In Progress" e la *task* corrispondente viene spostata nel *bucket* "In Sviluppo" su *Planner*.
- **Ready for Test** - Completato lo sviluppo, l'*issue* passa in fase di *test* e l'attività corrispondente viene spostata nel *bucket* "In Test" su *Planner*.
- **Closed** - L'*issue* viene chiusa una volta completati i *test*, con il passaggio al *bucket* "Produzione" in *Planner*.

Anche qui, stati aggiuntivi come "Needs Info" e "Rejected" si riflettono rispettivamente nei *bucket* "In Attesa Informazioni" e "Annullate".

### 2.2.5 Organizzazione dei bucket in Planner e Stati Associati in Taiga

La gestione del processo di progetto su *Planner* e *Taiga* implica una chiara distinzione tra i vari *bucket* e stati.

Di seguito, una panoramica completa dei *bucket* su *Planner* e degli stati in *Taiga*:

#### Bucket su Planner

- Idee
- In Analisi [*Stakeholder*]
- In Analisi [Sviluppatori]
- Analizzata e Stimata
- In Sviluppo
- In *Test*
- Produzione
- Annullate
- In Attesa Informazioni

#### Stati in Taiga

- **User story:** *New, Analizzata e Stimata, Ready, In Progress, Ready for Test, Failed in Test, Done, Archived, Needs Info, Rejected*
- **Issue:** *New, In Progress, Ready for Test, Failed in Test, Closed, Needs Info, Hotfix Released, Rejected, Postponed, Hotfix Ready to Release*
- **Task all'interno delle user story:** *New, In Progress, Ready for Test, Closed, Needs Info, Rejected in Test*

## 2.3 Competenze previste

Sono stato inserito in un contesto collaborativo di ricerca e sviluppo ([Ricerca e Sviluppo \(R&D\)](#)), dove ho avuto l'opportunità di apprendere le linee guida attualmente in uso e sviluppare una proposta di integrazione completa. Questo ambiente di lavoro stimolante offre la possibilità di interagire con professionisti esperti e di contribuire attivamente alla creazione di una soluzione che possa portare valore aggiunto all'azienda.

Prima dell'inizio del mio percorso formativo, era stato concordato che avrei acquisito competenze tecniche in ambito Ricerca e Sviluppo, analisi dei requisiti. Inoltre, avrei sviluppato un certo grado di professionalità adeguato agli *standard* richiesti nel contesto lavorativo.

Questo processo formativo non solo ha migliorato le mie abilità tecniche, ma ha rafforzato anche le mie capacità di lavoro in *team*, collaborazione e *problem solving*.

Le competenze specifiche che ci si attendeva acquisirsi durante il progetto formativo sono illustrate nella Tabella [2.2](#).

Competenza	Descrizione
Ricerca e sviluppo	Capacità di condurre analisi tecniche e di implementare soluzioni innovative.
Analisi dei requisiti	Competenze nella raccolta e valutazione dei requisiti per l'integrazione dei sistemi.
Standard professionali	Conoscenza delle <i>best practice</i> e degli <i>standard</i> di settore nel contesto lavorativo.
Lavoro di <i>team</i>	Esperienza nella collaborazione con diversi professionisti per raggiungere obiettivi comuni.
<i>Problem solving</i>	Capacità di affrontare e risolvere in modo efficace le problematiche emerse durante il progetto.

**Tabella 2.2:** Competenze previste durante il progetto formativo

## 2.4 Obiettivi

Per l'attività di *stage*, sono stati stabiliti, in accordo con il *tutor* aziendale, degli obiettivi chiari e distinti, suddivisi in obbligatori, desiderabili e facoltativi.

Questi obiettivi hanno guidato il progetto e hanno garantito che le esigenze aziendali siano state adeguatamente soddisfatte.

Di seguito si illustrano le sigle utilizzate per identificare i requisiti:

- «O» per i requisiti obbligatori, che sono essenziali e rappresentano gli obiettivi primari del progetto;
- «D» per i requisiti desiderabili, che, pur non essendo vincolanti, apportano un valore aggiunto significativo e contribuiscono al miglioramento complessivo del progetto;
- «F» per i requisiti facoltativi, che rappresentano opportunità aggiuntive e miglioramenti non strettamente necessari, ma che possono arricchire l'*output* finale.

Le sigle indicate saranno seguite da una numerazione sequenziale, che permetterà di identificare facilmente ciascun requisito.

### 2.4.1 Obiettivi Obbligatori

- O1: Mappatura delle funzionalità disponibili nei due applicativi. Questo obiettivo include un'analisi approfondita dei due sistemi di pianificazione, l'osservazione delle modalità operative degli utenti e la creazione di una documentazione dettagliata di analisi dei requisiti.
- O2: Personalizzazione e integrazione dei sistemi. Si mira a individuare modalità di personalizzazione e integrazione tra *Microsoft Planner* e *Taiga*, per ottimizzare l'interazione tra i due strumenti.

### 2.4.2 Obiettivi Desiderabili

- D1: Analisi dei requisiti per l'integrazione aziendale, che contribuirà a delineare un quadro chiaro delle necessità operative dell'azienda.
- D2: Produzione di una documentazione progettuale completa, utile per facilitare la comprensione e l'adozione del progetto da parte di tutti gli *stakeholder* coinvolti.

### 2.4.3 Obiettivi Facoltativi

- F1: Realizzazione di un *PoC* che dimostri alcune delle funzionalità e dei requisiti individuati, evidenziando il potenziale dell'integrazione.
- F2: Presentazione interna agli *stakeholder* aziendali, per illustrare i risultati raggiunti e il valore aggiunto apportato dal progetto.
- F3: Predisposizione di un sistema strutturato ed efficace per garantire la manutenibilità dell'applicativo, affinché possa essere facilmente aggiornato e adattato nel tempo.

## Capitolo 3

# Tecnologie utilizzate

### 3.1 Microsoft Office 365



**Figura 3.1:** Logo *Microsoft Office 365*

*Microsoft 365* (vedi figura 3.1) è una *suite* integrata di strumenti per la produttività e la collaborazione, progettata per supportare le attività aziendali con una vasta gamma di applicazioni interconnesse, come *Outlook*, *Teams*, *Planner*, *SharePoint* e molte altre.

All'interno del mio progetto di *stage*, alcuni componenti di *Microsoft 365*, in particolare *Planner* e *SharePoint*, sono stati fondamentali per lo sviluppo di un'applicazione che integrasse e sincronizzasse le attività aziendali tra *Planner* e *Taiga*.

L'obiettivo era creare un sistema che permettesse una sincronizzazione *server-to-server* tra *Planner* e *Taiga*: determinati cambiamenti effettuati in un piano su *Planner* venivano aggiornati automaticamente nel progetto su *Taiga*, e viceversa.

Ho deciso di utilizzare *SharePoint* come *repository* per i *file* di *log* e i *file* di configurazione necessari al corretto funzionamento dell'applicazione.

Questa scelta si è rivelata particolarmente vantaggiosa in quanto *SharePoint*, essendo accessibile internamente a tutti i membri dell'azienda, evitava la necessità di esporre un [endpoint](#) della rete aziendale per gestire tali dati.

In questo modo, si sono ridotti sensibilmente i rischi di sicurezza legati a un accesso esterno e si garantiva al contempo una disponibilità centralizzata e sicura della documentazione, dei *log* operativi e delle configurazioni chiave.

### 3.1.1 Teams



**Figura 3.2:** Logo *Microsoft Teams*

*Microsoft Teams* (vedi figura 3.2) è la piattaforma di comunicazione unificata di *Microsoft 365*, che supporta *chat*, chiamate audio e video, riunioni e condivisione di *file*.

Nel mio progetto, *Teams* è stato il principale strumento di comunicazione per le interazioni quotidiane e settimanali con il *tutor* aziendale, Enrico Merigliano, e altri membri del *team* di Ricerca e Sviluppo, tra cui Alice Sasso.

*Teams* è stato essenziale per la programmazione delle riunioni e per coordinare le attività di progetto.

La funzione di calendario mi ha permesso di visualizzare la disponibilità dei partecipanti in tempo reale, facilitando l'organizzazione di riunioni in orari compatibili con gli impegni dei colleghi.

Nella sezione *Team*, a cui accedevo con permessi limitati, ero integrato nel *Team* di *DevOps - Pianificazione*, per inviare i *file* e documenti tecnici del *PoC* che ho realizzato.

Un'altra caratteristica vantaggiosa di *Teams* è stata l'integrazione con *OneDrive*. Grazie a questa funzionalità, ho potuto archiviare documenti e *file* in un ambiente centralizzato e sicuro, accedendo alle risorse necessarie in qualsiasi momento e da qualsiasi dispositivo, riducendo il rischio di perdita di dati e facilitando la sincronizzazione delle modifiche in tempo reale.

### 3.1.2 Outlook



**Figura 3.3:** Logo di *Microsoft Outlook*

Oltre a *Teams*, un altro strumento fondamentale di *Office 365* è stato *Outlook* (vedi figura 3.3), utilizzato per la gestione della posta elettronica.

La posta elettronica si è rivelata particolarmente utile per tracciare le comunicazioni e per avere uno storico consultabile di tutte le informazioni rilevanti.

*Outlook* mi ha permesso di gestire appuntamenti e scadenze grazie al calendario integrato, sincronizzato con *Teams*.

Questa integrazione ha assicurato una gestione ordinata degli impegni e delle attività, grazie anche alla funzione di promemoria, che segnalava le riunioni imminenti.

### 3.1.3 OneDrive



**Figura 3.4:** Logo *Microsoft OneDrive*

*OneDrive* (vedi figura 3.4) ha rappresentato uno strumento chiave per la gestione e l'archiviazione dei *file* di progetto in modo sicuro e accessibile.

Grazie all'integrazione diretta con *Teams* e agli accessi rapidi, *OneDrive* ha facilitato l'archiviazione e la condivisione dei documenti, mantenendo tutti i *file* aggiornati e accessibili.

Uno degli svantaggi di *OneDrive* riguarda la configurazione dei permessi di condivisione che richiede attenzione, in quanto può rallentare il processo di accesso se non gestita in modo corretto.

L'utilizzo del pacchetto *Office 365*, con particolare riferimento a *Teams*, *Outlook* e *OneDrive*, ha svolto un ruolo centrale nel facilitare la comunicazione, l'organizzazione e la gestione documentale durante lo *stage*.



Microsoft Planner

**Figura 3.5:** Logo *Microsoft Planner*

*Microsoft Planner* (vedi figura 3.5) è stato il fulcro operativo del mio progetto di *stage*.

Questo strumento consente di creare e visualizzare attività tramite una struttura basata su piani e *task*, facilitando il coordinamento e il monitoraggio delle attività aziendali in modo sistematico e visivo.

#### Struttura del Piano

In *Planner*, ogni progetto aziendale viene rappresentato come un piano, un contenitore di *task* correlati, che rappresentano i vari requisiti del progetto.

All'interno di ciascun piano, le attività possono essere organizzate in *bucket*, sezioni personalizzabili che permettono di suddividere le attività in categorie specifiche o fasi di sviluppo.

Ogni piano è associato a un gruppo (*Microsoft 365 Group*), che consente di gestire gli utenti che possono accedere e collaborare nel piano.

Durante la creazione, *Planner* offre la possibilità di associare un nuovo piano a un gruppo esistente o di crearne uno nuovo, garantendo così una flessibilità di utilizzo e di condivisione.



## Gestione dei Task

Test my app

○ **Prima Task**  
Ultima modifica effettuata in data qualche istante fa da te

Assegna

Aggiungi etichetta

Contenitore: Idee

Stato: ○ Non iniziata

Priorità: ● Media

Data di inizio: Qualsiasi data

Scadenza: Qualsiasi scadenza

Ripeti: ↻ Non si ripete

Note  Mostra nella scheda

Digita una descrizione o aggiungi note qui

**Figura 3.6:** Caratteristiche specifiche di una *task*

Ogni *task* rappresenta un requisito da analizzare e da completare e può essere configurato per adattarsi alle esigenze specifiche del progetto (vedi figura 3.6).

Le principali funzionalità dei *task* in *Planner* includono:

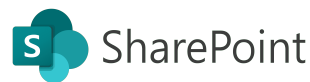
- **Titolo e Descrizione:** Ogni attività ha un titolo distintivo e una descrizione dettagliata, che ne esplicita obiettivi e specifiche.
- **Assegnazione e Scadenza:** È possibile assegnare uno o più membri del *team* a un *task* e impostare una data di scadenza, facilitando così la gestione delle responsabilità e delle tempistiche.
- **Checklist:** Per monitorare meglio le attività più complesse, *Planner* consente di suddividere ogni *task* in sotto-attività tramite una *checklist*.
- **Tag e Label:** Ogni *task* può essere associato a *label* colorate, utilizzabili come *tag* visivi per categorizzare le attività in base a criteri specifici (es. utenza, tipologia), migliorando la leggibilità e l'organizzazione complessiva.
- **Commenti e Allegati:** *Planner* permette di tracciare le comunicazioni relative al *task* tramite commenti, creando uno spazio di discussione per il progresso dell'attività.

### Visualizzazione Kanban e Monitoraggio

Una delle caratteristiche più apprezzate di *Planner* è la visualizzazione in stile *Kanban*, che dispone i *task* come schede ordinate in colonne all'interno dei vari *bucket*.

Questa struttura visiva facilita la comprensione del flusso di lavoro, evidenziando lo stato di avanzamento.

### 3.1.4 SharePoint



**Figura 3.7:** Logo *Microsoft SharePoint*

*SharePoint* (vedi figura 3.7) è stato utilizzato come archivio centralizzato e sicuro per i *file* di configurazione e i *file* di *log* generati dalla *PoC* realizzata durante lo *stage*.

Ho optato per *SharePoint* come *repository* sicuro per i file della *PoC*, evitando così la necessità di ospitare un *database* dedicato sul *server* aziendale, soluzione che avrebbe potuto presentare criticità in termini di sovraccarico e di sicurezza.

#### Gestione Documentale in SharePoint

*SharePoint* offre un ambiente di archiviazione strutturato e consultabile tramite interfaccia *web*, rendendolo ideale per la memorizzazione e la condivisione di documenti all'interno dell'organizzazione.

Nello specifico, ho utilizzato *SharePoint* per creare e gestire un *sito* dedicato, all'interno del quale ho organizzato in una cartella specifica tutti i file di *log* e i *file* di configurazione della *PoC*.

Questa impostazione ha facilitato l'accesso e il recupero delle informazioni necessarie, garantendo al contempo che i *file* fossero centralizzati e accessibili dall'azienda.

#### Motivazioni per la Scelta di SharePoint

L'azienda ha deciso di evitare l'uso di un *database* per i *file* della *PoC*, principalmente per limitare il rischio di esposizione della rete aziendale, quindi ho optato per una soluzione *cloud-based* interna come *SharePoint*.

L'infrastruttura aziendale non è attualmente configurata per esporre i propri dati a sistemi esterni, e un *database* esterno avrebbe comportato sfide aggiuntive legate alla sicurezza della rete e al monitoraggio degli accessi.

Nel complesso, *SharePoint* ha soddisfatto le esigenze di memorizzazione, accessibilità e sicurezza per i documenti della *PoC*, offrendo una piattaforma stabile e integrata che non richiedesse risorse aggiuntive sul *server* aziendale e mantenendo il controllo completo sugli accessi e sulla distribuzione dei *file* di progetto.

Si è quindi deciso di utilizzare questa piattaforma soprattutto per gestire i *file* di *log* e di configurazione, per il corretto funzionamento della *PoC*.

I *file* di *log* e i file di configurazione verranno spiegati in dettaglio nei prossimi capitoli, offrendo una visione più chiara di come vengono utilizzati.

I file di *log* sono *file* generati automaticamente che registrano le azioni svolte dalla *PoC* in relazione a ciascuna coppia di *piano* di *Planner* e *progetto* di *Taiga*.

Questi *file* documentano quando una modifica è stata eseguita, da chi, cosa è stato fatto e quali saranno le prossime operazioni della *PoC* in seguito a questa modifica.

I *file* di configurazione, invece, vengono generati dopo aver configurato una relazione tra un *piano* di *Planner* e un *progetto* di *Taiga* sulla base delle personalizzazioni aziendali.

### 3.1.5 Microsoft Power Automate



**Figura 3.8:** Logo *Microsoft Power Automate*

*Microsoft Power Automate* (vedi figura 3.8) è una piattaforma di automazione dei flussi di lavoro inclusa nella *suite Microsoft 365*, che consente di automatizzare processi aziendali e operazioni ripetitive.

Utilizzando un'interfaccia grafica intuitiva basata su **drag-and-drop**, *Power Automate* permette di collegare applicazioni e servizi diversi, creando flussi di lavoro, chiamati *flow*, che si attivano in risposta a specifiche condizioni o eventi.

#### Caratteristiche Principali di Power Automate

Le funzionalità principali di *Power Automate* includono:

- **Automazione dei processi:** consente di automatizzare attività ripetitive, come invio di notifiche, sincronizzazione di dati tra sistemi e operazioni di approvazione.
- **Integrazione con servizi cloud:** supporta l'integrazione con un vasto numero di applicazioni di terze parti e servizi *cloud* sempre all'interno del pacchetto *office 365*, come *SharePoint*, *Planner*, *Teams*, e molte altre, facilitando il collegamento e il trasferimento dati tra sistemi.
- **Flussi di lavoro basati su trigger:** ogni *flow* può essere attivato da *trigger* specifici, come la creazione di un *file*, la ricezione di un'*email* e tanto altro, rendendo i processi reattivi e adattabili.
- **Condizioni e ramificazioni:** *Power Automate* supporta flussi di lavoro complessi che includono condizioni, cicli e ramificazioni logiche, permettendo di costruire processi automatizzati che rispondono a scenari aziendali articolati.

#### Utilizzo di Power Automate per la PoC

All'interno del mio progetto di *stage*, *Power Automate* è stato utilizzato durante la fase finale della *PoC* per tentare l'integrazione della soluzione sviluppata all'interno dell'infrastruttura aziendale.

Per dettagli sui risultati dell'integrazione della *PoC* tramite *Power Automate* e le soluzioni finali adottate, si rimanda al Capitolo 6, che descrive i *test* di integrazione eseguiti e l'efficacia dei flussi di lavoro implementati.

### 3.1.6 Microsoft Azure



**Figura 3.9:** Logo *Microsoft Azure*

*Microsoft Azure* (vedi figura 3.9) è una piattaforma *cloud* di *Microsoft* che offre una vasta gamma di servizi per lo sviluppo, la gestione e la distribuzione di applicazioni in ambienti altamente scalabili e sicuri.

Tra i vari servizi disponibili, *Azure Active Directory* fornisce un sistema di gestione delle identità e degli accessi, fondamentale per l'autenticazione delle applicazioni e per la gestione delle autorizzazioni in ambienti aziendali.

Nel contesto del mio progetto di *stage*, *Azure* è stato utilizzato per registrare l'applicazione necessaria alla comunicazione *server-to-server* tra *Planner* e *Taiga*.

La registrazione ha permesso di generare un'identità applicativa sicura, indispensabile per la comunicazione *cloud-based* tra i due sistemi e per la gestione degli accessi alle risorse aziendali.

Tramite *Azure Active Directory*, è stata assegnata all'applicazione una serie di permessi configurati per limitare l'accesso ai soli ambiti essenziali, garantendo una comunicazione sicura e conforme agli *standard* aziendali.

Per maggiori informazioni sulla configurazione dei permessi e sull'impostazione dell'applicazione in *Azure*, si rimanda alla Sezione 5.1.2.

### 3.1.7 Microsoft Graph



**Figura 3.10:** Logo *Microsoft Graph*

*Microsoft Graph* (vedi figura 3.10) è il fulcro dell'ecosistema *API* di *Microsoft 365*, offrendo un'interfaccia unica e centralizzata per accedere a tutte le applicazioni e i servizi all'interno della *suite*.

Qualsiasi applicazione *Microsoft 365* che necessiti di accedere ai dati o alle funzionalità di altri strumenti della piattaforma, come *Planner*, *Outlook* o *SharePoint*, deve passare attraverso *Graph*, rendendolo il punto di accesso privilegiato e obbligato per la comunicazione tra i servizi *cloud* aziendali.

Nel contesto della *PoC*, *Microsoft Graph* è stato fondamentale per interfacciarsi con le *API* di *Planner*, dei *Groups* e di *SharePoint*.

Grazie alle *API* di *Planner* e dei *Groups*, l'applicazione ha potuto gestire e sincronizzare le attività tra *Planner* e *Taiga*.

L'interazione con *SharePoint* tramite *Graph* ha inoltre garantito l'accesso sicuro e centralizzato ai *file di log* e di configurazione, fondamentali per il funzionamento dell'applicazione.

## 3.2 Taiga



Figura 3.11: Logo Taiga

*Taiga* (vedi figura 3.11) è una piattaforma *opensource* per la gestione *agile* dei progetti, pensata per supportare i *team* di sviluppo e adattarsi a metodologie come *kanban* e *scrum*.

Durante il mio progetto, *Taiga*, come *Planner*, è stato il fulcro del mio *stage*.

### 3.2.1 Caratteristiche principali di Taiga

Tra le funzionalità chiave di *Taiga*, si trovano:

- **Gestione delle User Story:** Taiga permette di definire le *user story*, ovvero descrizioni delle funzionalità richieste dal punto di vista dell'utente. Ogni *user story* viene ulteriormente suddivisa in attività specifiche, facilitando l'assegnazione e il controllo delle responsabilità. In azienda una *user story* corrisponde a una *task* in *Planner* e quindi a un requisito del progetto che va implementato.
- **Gestione delle Issue:** *Taiga* permette di gestire le *issue*, ovvero problemi o impedimenti riscontrati durante il processo di sviluppo. Ogni *issue* può essere assegnata, classificata per priorità e monitorata fino alla sua risoluzione, facilitando il controllo dei problemi in corso.
- **Kanban e Scrum Board:** La piattaforma offre sia una visualizzazione *Kanban*, che consente di organizzare le attività in colonne, sia una visualizzazione *Scrum* basata su *sprint*.
- **Sprint Planning e Backlog:** Per ogni progetto, Taiga consente di pianificare *sprint* e di gestire un *backlog*, ovvero l'elenco delle attività e delle priorità. Questa struttura è particolarmente utile per organizzare il lavoro in cicli settimanali o mensili, mantenendo traccia delle attività completate e di quelle ancora in corso.
- **Epiche e Task:** Taiga permette di raggruppare le *user story* in epiche più ampie e di suddividerle in task specifici, creando così una gerarchia di obiettivi e compiti. Questa organizzazione gerarchica permette al *team* di focalizzarsi sia sugli obiettivi di alto livello sia sui dettagli operativi.
- **Tracciamento del Tempo:** La piattaforma include strumenti per il monitoraggio del tempo impiegato, permettendo di avere una panoramica precisa della produttività e di individuare eventuali aree di miglioramento.

## 3.3 Postman



Figura 3.12: Logo *Postman*

*Postman* (vedi figura 3.12) è uno strumento avanzato per il *testing* e l'automazione delle *API*, largamente utilizzato per eseguire richieste [HyperText Transfer Protocol \(HTTP\)](#), analizzare risposte e monitorare le interazioni con *endpoint RESTful*.

Durante lo *stage*, *Postman* è stato essenziale per l'integrazione di *Microsoft Planner*, *Taiga* e *SharePoint* nel mio *PoC*, permettendomi di esplorare e ottimizzare le chiamate *api* e di verificare la correttezza dei dati ricevuti.

### 3.3.1 Analisi delle Risposte JSON

Una delle funzionalità principali di *Postman* è la capacità di mostrare in tempo reale le risposte in formato [JavaScript Object Notation \(JSON\)](#), rendendo possibile un'analisi dettagliata dei dati restituiti da ciascuna piattaforma.

Attraverso *Postman*, ho potuto visualizzare e comprendere le informazioni contenute nelle risposte, come le strutture di dati, i valori restituiti e i possibili errori.

Questo processo ha aiutato a:

- **Identificare i Valori Necessari:** Ho utilizzato *Postman* per decidere quali valori delle risposte *JSON* fossero rilevanti per la mia *PoC*, come *ID*, nomi, descrizioni o stati delle *task*, mentre altri dati meno utili per l'applicazione sono stati esclusi. Questa selezione è stata cruciale per individuare i dati specifici da utilizzare all'interno della *PoC*, assicurando che venissero inclusi solo i valori effettivamente necessari per il funzionamento dell'integrazione tra *Planner*, *Taiga* e *SharePoint*.
- **Comprendere le Specifiche di Output:** Grazie alla visualizzazione chiara delle risposte *JSON*, è stato possibile determinare quali valori dovessero essere trattati, modificati o riutilizzati nei processi dell'integrazione.
- **Verifica delle Risposte:** *Postman* mi ha permesso di verificare se le risposte *JSON* contenessero i dati attesi e di identificare eventuali discrepanze o errori, guidandomi nell'ottimizzazione delle successive richieste.

## 3.4 Java

Per lo sviluppo della *PoC*, in conformità con le richieste aziendali, ho utilizzato *Java JDK 21*.

Questa versione offre miglioramenti significativi, soprattutto in termini di prestazioni e funzionalità avanzate, come le classi di *record*, i miglioramenti alla gestione delle eccezioni, e le nuove funzionalità per le espressioni *lambda*.

*Java 21* si è dimostrato particolarmente adatto per lo sviluppo di applicazioni aziendali grazie alla sua affidabilità e alla vasta gamma di librerie disponibili.

## 3.5 IDE: IntelliJ IDEA (versione Community)



Figura 3.13: Logo *IntelliJ IDEA*

Lo sviluppo è stato realizzato utilizzando *IntelliJ IDEA* (vedi figura 3.13), un ambiente di sviluppo integrato ([Integrated Development Environment \(IDE\)](#)) potente e intuitivo, noto per la sua efficienza e l'integrazione avanzata con *Java*.

Ho utilizzato la versione *Community*, gratuita e accessibile, che offre comunque tutte le funzionalità essenziali per un *workflow* completo e produttivo.

*IntelliJ IDEA* ha facilitato il processo di sviluppo grazie a strumenti integrati come l'autocompletamento intelligente del codice, la gestione avanzata dei progetti *Maven*, il *debugging* e la visualizzazione chiara della struttura del progetto.

In aggiunta, il supporto per l'integrazione di *framework* come *Spring Boot* ha reso agevole la configurazione e il controllo dell'applicazione durante l'intero ciclo di sviluppo.

## 3.6 Spring Boot e Thymeleaf

*Spring Boot* è stato scelto come *framework* principale per lo sviluppo dell'applicazione, grazie alla sua flessibilità e alle sue capacità di gestione delle dipendenze, che semplificano l'implementazione di applicazioni *web* complesse.

Con *Spring Boot* è stato possibile configurare rapidamente il progetto, utilizzando le convenzioni *standard* senza dover configurare manualmente ogni singola dipendenza.

### 3.6.1 Moduli di Spring Boot Utilizzati

- **Spring Web:** Ha permesso di costruire *endpoint RESTful* per la comunicazione tra i diversi moduli dell'applicazione e l'integrazione delle *API* esterne come quelle di *Planner*, *Taiga* e *SharePoint*.
- **Thymeleaf:** Utilizzato come motore di *template* per il *rendering* delle pagine [HyperText Markup Language \(HTML\)](#), facilitando la gestione di contenuti dinamici e l'integrazione con il *backend*.

- **Spring WebFlux:** Integrato per migliorare il supporto alle comunicazioni asincrone e reattive, particolarmente utile per le interazioni con le *API* esterne.
- **Spring Boot DevTools:** Strumento opzionale per semplificare il ciclo di sviluppo grazie al caricamento automatico delle modifiche.
- **Spring Boot Starter Test e Reactor Test:** Forniscono, rispettivamente, strumenti per i *test* unitari e reattivi, migliorando l'affidabilità del codice.
- **Reactor Netty:** Gestisce le comunicazioni di rete non bloccanti, facilitando la gestione delle connessioni asincrone.

## 3.7 Apache Maven



Figura 3.14: Logo Apache Maven

*Apache Maven* (vedi figura 3.14) è stato utilizzato come strumento di gestione delle dipendenze e di costruzione del progetto.

Grazie alla sua capacità di automatizzare il processo di compilazione e gestione delle librerie, *Maven* ha semplificato notevolmente l'organizzazione del progetto e l'integrazione di diverse tecnologie.

### 3.7.1 Gestione delle Dipendenze

Uno dei principali vantaggi di *Maven* è la gestione delle dipendenze, che ha permesso di includere rapidamente le librerie necessarie per il progetto.

Grazie al *file* di configurazione `pom.xml`, tutte le dipendenze del progetto sono centralizzate e organizzate, permettendo a *Maven* di risolvere automaticamente le versioni appropriate e gestire le eventuali dipendenze.

### 3.7.2 Configurazione e Automazione

Oltre alla gestione delle dipendenze, *Maven* è stato utilizzato per automatizzare diverse fasi del ciclo di vita del progetto, tra cui:

- **Compilazione:** *Maven* ha automatizzato la compilazione del codice *Java*, riducendo il rischio di errori manuali e garantendo una compilazione coerente.
- **Testing:** Grazie alla sua integrazione con librerie di *testing* come *JUnit*, *Maven* permette di eseguire i *test* in modo automatico ad ogni *build*, verificando che tutte le funzionalità implementate rispondessero ai requisiti.
- **Packaging:** Con *Maven*, il progetto è stato impacchettato in un *file* `.jar` o `.war`, pronto per il *deployment* in ambienti di produzione o di *testing*.
- **Deployment:** Sebbene il progetto non fosse ancora stato distribuito in un ambiente di produzione, *Maven* ha reso il processo di *deployment* più agevole, preparandolo per un eventuale rilascio.



## 3.8 Git



**Figura 3.15:** Logo *Git*

*Git* (vedi figura 3.15) è uno strumento fondamentale per il controllo versione, che consente di tracciare tutte le modifiche al codice, facilitando il lavoro di squadra e la gestione dei conflitti.

*Git* è stato utilizzato come sistema di controllo versione per gestire il codice sorgente del progetto, mentre *GitHub* ha rappresentato il *repository* remoto per il *backup*.

L'integrazione di *Git* all'interno di *IntelliJ IDEA* ha semplificato notevolmente il flusso di lavoro, permettendo di effettuare operazioni di versionamento e gestione del codice direttamente dall'ambiente di sviluppo.

### 3.8.1 Gestione del Codice con Git

L'uso di *Git* ha permesso di:

- **Creare rami (branch):** Ogni nuovo sviluppo o funzionalità è stato realizzato su un ramo separato, garantendo che la versione principale restasse stabile.
- **Gestire le modifiche:** Le modifiche sono state salvate tramite *commit*, consentendo di tornare facilmente a versioni precedenti del progetto se necessario.
- **Unire i rami (merge):** I rami sono stati uniti tramite richieste di *pull* (*pull request*) su *GitHub* dopo che le modifiche erano state testate, facilitando l'integrazione delle funzionalità senza compromettere la stabilità del codice.

## Capitolo 4

# Analisi dei requisiti

L'analisi dei requisiti è stata una fase cruciale per il successo della *PoC*, poiché fornisce una visione chiara delle funzionalità, delle condizioni e dei vincoli necessari per poterla sviluppare in modo tale che risponda pienamente agli obiettivi aziendali.

La definizione e la catalogazione dei requisiti, inoltre, rappresentano un requisito obbligatorio per la realizzazione della *PoC*, essenziale per strutturare uno sviluppo conforme alle aspettative degli *stakeholder* e del *team* di sviluppo (*R&D*).

Questa analisi preliminare è stata condotta attraverso una serie di presentazioni e discussioni sia con Enrico Merigliano, sia con il *team* di sviluppo.

I requisiti definiti nel corso dello *stage* definiscono le caratteristiche del sistema e costituiscono la base per tutte le successive fasi di sviluppo.

### 4.1 Requirements baseline

In questa sezione vengono elencati i requisiti emersi durante la fase di analisi del progetto.

I requisiti rappresentano le funzionalità, le condizioni e i vincoli necessari per lo sviluppo del sistema, e sono stati organizzati e catalogati secondo un codice univoco, utilizzando un formalismo che ne facilita la tracciabilità e la gestione.

I requisiti sono stati suddivisi in tre macrocategorie principali:

1. **Requisiti Funzionali (RF)**: descrivono le funzionalità che il sistema dovrà implementare, specificando le interazioni tra il sistema e l'utente.
2. **Requisiti di Qualità (RQ)**: definiscono le metriche di qualità, gli *standard* da rispettare e la documentazione da fornire per garantire un prodotto robusto e ben documentato.
3. **Requisiti di Vincolo (RV)**: indicano i limiti tecnici e le tecnologie specifiche da utilizzare, oltre agli ambienti operativi in cui il sistema dovrà funzionare.

### 4.1.1 Requisiti Funzionali

I requisiti funzionali descrivono le operazioni principali che il sistema dovrà essere in grado di eseguire, specificando le azioni disponibili per l'utente e i servizi che il sistema fornirà in risposta alle richieste.

Ogni requisito funzionale è identificato da un codice univoco, che ne indica il tipo e la priorità:

- **RFO (Requisito Funzionale Obbligatorio)**: requisiti essenziali per il funzionamento di base del sistema, necessari per considerare il sistema completo.
- **RFF (Requisito Funzionale Facoltativo)**: funzionalità aggiuntive non essenziali ma che possono migliorare l'esperienza utente o l'efficienza operativa.
- **RFD (Requisito Funzionale Desiderabile)**: funzionalità desiderate che, se presenti, arricchirebbero il sistema, ma la cui assenza non compromette l'usabilità di base.

#### RFO1 – Autenticazione con Taiga

Il sistema deve essere in grado di autenticarsi automaticamente con *Taiga* utilizzando credenziali preconfigurate di un [utente tecnico](#).

Durante il processo di autenticazione, il sistema deve ottenere un [token delegato](#) indispensabile per l'interazione con le *API* di *Taiga*. Questo [token](#) sarà utilizzato per eseguire operazioni relative alla gestione delle *user story*, delle *issue* e dei progetti.

#### RFO2 – Autenticazione con Planner

Il sistema deve autenticarsi automaticamente con *Microsoft Planner* sfruttando il servizio di *Microsoft Azure* per ottenere un *token* applicativo che consenta di gestire e automatizzare le operazioni sui *task* all'interno di *Microsoft Planner*.

#### RFO3 – Gestione del rinnovo dei token di accesso

Il sistema deve monitorare la scadenza dei *token* di accesso utilizzati per interfacciarsi con *Taiga* e *Microsoft Planner*, gestendone il rinnovo in modo automatico.

Poiché la durata dei *token* è limitata a un'ora, il sistema deve richiederne di nuovi proattivamente prima della scadenza, assicurando continuità nelle operazioni e minimizzando eventuali interruzioni.

#### RFO4 – Sincronizzazione dei progetti

Il sistema deve garantire la sincronizzazione periodica dell'elenco dei progetti presenti in *Taiga*, aggiornandoli automaticamente in base alle modifiche effettuate, come la creazione di nuovi progetti o l'eliminazione di quelli esistenti.

La sincronizzazione deve assicurare che tutti i nuovi progetti creati in *Taiga* siano visibili nel sistema e che i progetti eliminati siano rimossi.

#### RFO5 – Sincronizzazione dei piani

Il sistema deve sincronizzare automaticamente l'elenco dei piani associati a uno o più [group-id](#) in *Microsoft Planner*. Questa sincronizzazione deve rilevare periodicamente modifiche come la creazione di nuovi piani o l'eliminazione di quelli esistenti, garantendo che i dati rimangano coerenti.

**RFO6 – Definizione della sincronizzazione tra piani e progetti**

Il sistema deve permettere la configurazione di una sincronizzazione tra un piano di *Microsoft Planner* e un progetto di *Taiga*, utilizzando i rispettivi identificativi.

Una volta configurata, la sincronizzazione deve propagare automaticamente le modifiche effettuate su uno degli elementi (piano di *Microsoft Planner* o progetto di *Taiga*) all'altro, mantenendo i dati costantemente allineati.

**RFO7 – Esecuzione periodica delle funzioni di sincronizzazione**

Il sistema deve eseguire automaticamente, a intervalli regolari, funzioni di sincronizzazione per aggiornare *user story*, *issue* e *task* tra *Taiga* e *Microsoft Planner*.

Questa operazione deve assicurare che i dati siano sempre aggiornati, eliminando la necessità di interventi manuali da parte degli utenti.

**RFO8 – Recupero delle informazioni di dettaglio di un piano**

Il sistema deve recuperare periodicamente le informazioni strutturali associate a ciascun piano presente in *Microsoft Planner*.

**RFO9 – Recupero delle informazioni di dettaglio di un progetto**

Il sistema deve recuperare periodicamente le informazioni strutturali di ciascun progetto presente in *Taiga*, nei quali appartiene l'utente tecnico.

**RFO10 – Identificazione delle task relative a user story o issue**

Il sistema deve identificare automaticamente, all'interno di *Microsoft Planner*, quali *task* siano associate a *user story* e quali a *issue*.

Questa identificazione avviene utilizzando specifici *tag*:

- **BugTask**: sono tutte le *task* aventi il *tag Bug* e fanno riferimento a *issue*;
- **RegularTask**: sono tutte le *task* prive del *tag Bug* e fanno riferimento a *user story*.

**RFO11 – Monitoraggio delle modifiche alle BugTask**

Il sistema deve monitorare costantemente le modifiche apportate alle *BugTask* e alle *RegularTask* in *Microsoft Planner*.

Le modifiche da rilevare includono:

- Titolo della *task*;
- *Category/tag* associati;
- Priorità della *task*;
- Data di scadenza;
- Descrizione.

**RFO12 – Monitoraggio delle modifiche a user story e issue**

Il sistema deve monitorare in tempo reale le modifiche apportate alle *user story* e alle *issue* nei progetti presenti in *Taiga*.

Deve essere in grado di rilevare qualsiasi cambiamento di stato, come ad esempio il passaggio di una *user story* o di una *issue* dallo stato “*In Progress*” a quello “*Done*”.

**RFO13 – Gestione delle modifiche alle BugTask**

Il sistema deve rilevare quali delle modifiche effettuate siano significative. Inoltre, deve essere in grado di gestire automaticamente le modifiche significative apportate alle *BugTask* in *Microsoft Planner*, sincronizzandole con le *issue* corrispondenti in *Taiga*, se esiste una relazione tra gli elementi.

Le modifiche rilevanti includono:

1. Titolo;
2. Data di scadenza;
3. *Category/Tag*;
4. Priorità;
5. Descrizione.

Il sistema deve verificare se una *BugTask* modificata è associata a una *issue* in *Taiga* e, in caso affermativo, riflettere tali modifiche. Se non è presente alcuna associazione, il sistema deve segnalare l'incoerenza tramite un sistema di *log*.

**RFO14 – Gestione delle modifiche alle Regulartask**

Il sistema deve essere in grado di rilevare quali sono le modifiche significative apportate a una *RegularTask* in *Microsoft Planner* e sincronizzarle con la *user story* corrispondente in *Taiga*.

Le modifiche rilevanti includono:

1. Titolo;
2. Data di scadenza;
3. *Category/Tag*;
4. Priorità;
5. Descrizione.

Il sistema deve verificare se la *RegularTask* modificata è associata a una *user story* esistente e, in caso affermativo, aggiornare la *user story* corrispondente in *Taiga*.

Se la *RegularTask* non è collegata a una *user story* (ad esempio, perché si trova in uno stato ignorabile), il sistema non deve procedere con l'aggiornamento.

**Gestione dei bucket:** Se una *RegularTask* viene spostata nel *bucket* specifico “*In Analisi [Sviluppatori]*”, il sistema deve creare automaticamente la *user story* corrispondente nel *backlog* di *Taiga* (vedi Sezione 4.1.1).

Se invece si trova in *bucket* ignorabili come “*Idee*” o “*In Analisi [Stakeholder]*”, non verrà sincronizzata finché non cambierà *bucket*, in quanto tali modifiche sono considerate irrilevanti.

#### **RFO15 – Gestione delle modifiche alle *user story***

Il sistema deve rilevare i cambiamenti di stato delle *user story* in *Taiga* e sincronizzare tali modifiche con la *RegularTask* corrispondente in *Microsoft Planner*.

I seguenti scenari devono essere gestiti:

1. Se la *user story* passa allo stato “*In Progress*”, la *task* corrispondente deve essere spostata nel *bucket* “*In Sviluppo*”;
2. Se passa a “*Ready for test*”, la *task* deve essere spostata nel *bucket* “*In Test*”;
3. Se passa a “*Done*”, la *task* deve essere spostata nel *bucket* “*In Produzione*”;
4. Se passa a “*Need info*”, la *task* deve essere spostata nel *bucket* “*In attesa informazioni*”;
5. Se passa a “*Rejected*”, la *task* deve essere spostata nel *bucket* “*Annullate*”.

**Gestione delle incongruenze:** Se il sistema non trova la *task* corrispondente o il *bucket* di destinazione, deve notificare l’errore tramite un sistema di *log*.

**Calcolo dei punti e aggiornamento del titolo della *user story*** Quando una *user story* cambia stato in “*Analizzata e Stimata*”, il sistema deve calcolare i punti della *user story* necessari per completarla, utilizzando la conversione presente nella Tabella 2.1.

Una volta completato il calcolo, il sistema deve aggiornare il titolo della *task* associata in *Microsoft Planner*, e di conseguenza della *user story* come descritto nella Sezione 4.1.1.

#### **RFO16 – Gestione delle modifiche alle *issue***

Il sistema deve sincronizzare i cambiamenti di stato delle *issue* in *Taiga* con le *BugTask* corrispondenti in *Microsoft Planner*.

I seguenti scenari devono essere gestiti:

1. Se l’*issue* passa a “*In Progress*”, la *task* corrispondente deve essere spostata nel *bucket* “*In Sviluppo*”;
2. Se passa a “*Ready for test*”, la *task* deve essere spostata nel *bucket* “*In Test*”;
3. Se passa a “*Closed*”, la *task* deve essere spostata nel *bucket* “*In Produzione*”;
4. Se passa a “*Need info*”, la *task* deve essere spostata nel *bucket* “*In attesa informazioni*”;
5. Se passa a “*Rejected*”, la *task* deve essere spostata nel *bucket* “*Annullate*”.

**Gestione delle incongruenze:** Il sistema deve notificare l'errore se la *task* corrispondente non viene trovata o se il *bucket* di destinazione non è disponibile attraverso un sistema di *log*

#### **RFO17 – Creazione di *user story***

Il sistema deve rilevare quando una *RegularTask* viene spostata nel *bucket* “*In Analisi [Sviluppatori]*”.

In tal caso, il sistema deve generare automaticamente una *user story* corrispondente, trasferendo i seguenti valori dalla *task*:

- Titolo;
- Category/*Tag*;
- Priorità;
- Scadenza;
- Descrizione.

#### **RFO18 – Creazione di *issue***

Il sistema deve identificare quando una *BugTask* viene creata.

In questo caso, il sistema deve generare automaticamente un'*issue* corrispondente, trasferendo i seguenti valori dalla *task*:

- Titolo;
- Categoria/*Tag*;
- Priorità;
- Scadenza;
- Descrizione.

#### **RFO19 – Scrittura di azioni su un File di Log**

Il sistema deve registrare tutte le azioni intraprese e le motivazioni dietro queste azioni in un file di *log*.

Questo file sarà essenziale per la ricostruzione delle dinamiche in caso di problemi e per l'attribuzione di responsabilità.

#### **Specifiche del File di Log:**

- Un file di *log* deve essere creato per ciascuna connessione tra un piano e un progetto.
- Il file avrà il formato di denominazione: TitoloPiano\_TitoloProgetto.txt.
- Ogni notifica registrata dovrà seguire il formato:

[GG/MM/AAAA e HH:MM:SS]: Ho fatto questa azione (es. spostato un *task* dal *bucket* X al *bucket* Y) perché [Nome Utente] ha cambiato lo stato della *user story* dallo stato W allo stato Z.

Le informazioni da registrare includono:

- Data e orario.
- Nome dell'utente che ha effettuato la modifica.
- Modifiche effettuate
- Azione intrapresa dal sistema.

#### **RFF1 – Scrittura di errore su File di Log**

Il sistema deve essere in grado di interpretare un errore e registrarlo nel file di *log*.

#### **RFO20 – Estrazione nome utente che ha modificato lo stato della Issue**

Il sistema deve essere in grado di identificare l'utente che ha effettuato una modifica significativa a un'*issue*, estraendo il suo *full name*.

#### **RFO21 – Estrazione nome utente che ha modificato lo stato della User Story**

Il sistema deve essere in grado di identificare l'utente che ha effettuato una modifica significativa a una *User Story*, estraendo il suo *full name*.

#### **RFO22 – Estrazione del nome utente che ha modificato una Task**

Il sistema dovrebbe essere in grado di estrarre e registrare il nome utente che ha effettuato modifiche a una *RegularTask* o a una *BugTask* all'interno di *Microsoft Planner*.

#### **RFO23 – Gestione tramite interfaccia web**

Il sistema deve fornire una pagina *web* dedicata per la gestione delle associazioni tra piani di *Microsoft Planner* e progetti di *Taiga*. In questa pagina, gli utenti devono essere in grado di eseguire le seguenti operazioni:

- Selezionare un piano di *Microsoft Planner* e un progetto di *Taiga* da mettere in associazione;
- Visualizzare le associazioni già esistenti;
- Eliminare associazioni esistenti;
- Configurare associazioni esistenti.

#### **RFF2 – Personalizzazione dei gruppi**

Il sistema deve consentire agli utenti di gestire i *group-id*, che rappresentano i gruppi di piani di *Microsoft Planner*.

Le seguenti funzionalità devono essere supportate:

- Inserimento di un nuovo *group-id*;
- Visualizzazione dei *group-id*.



**RFF3 – Modalità di aggiornamento completa**

Oltre alla modalità predefinita, il sistema deve supportare una modalità di aggiornamento completa che permette di recuperare tutti i piani di *Microsoft Planner* associati al [tenant](#), indipendentemente da quelli ottenuti dai *group-id* forniti.

**RFO24 – Salvataggio delle associazioni**

Il sistema deve consentire il salvataggio delle associazioni tra i piani di *Microsoft Planner* e i progetti di *Taiga*.

Una volta salvata un'associazione la sincronizzazione automatica deve iniziare, e quindi:

- Determinate modifiche apportate alle *task* in *Planner* saranno sincronizzate con le *user story* o le *issue* corrispondenti in *Taiga*;
- Determinate modifiche apportate in *Taiga*, saranno sincronizzate con le *task* corrispondenti in *Microsoft Planner*.

**RFF4 – Gestione delle associazioni Piano-Progetto esistenti**

Gli utenti devono poter gestire le associazioni esistenti tra piani di *Microsoft Planner* e progetti di *Taiga*. Le operazioni disponibili includono:

- Visualizzazione delle associazioni già create;
- Eliminazione delle associazioni non più necessarie;
- Configurazione personalizzata per le relazioni *bucket-user story*;
- Configurazione personalizzata per le *bucket-issue*.

**Configurazione delle associazioni per le *bucket-user story*** Il sistema deve permettere di personalizzare il collegamento tra i *bucket* di *Planner* e gli stati delle *user story* in *Taiga*.

- Gli utenti possono specificare manualmente quali *bucket* corrispondono a quali stati delle *user story*.
- Gli utenti possono eliminare associazioni create in precedenza.

**Configurazione delle associazioni per le *bucket-issue*** Analogamente alle *user story*, il sistema deve consentire di configurare manualmente i collegamenti tra *bucket* e stati delle *issue*.

- Gli utenti possono definire associazioni personalizzate per i *bucket* e gli stati delle *issue*.
- Gli utenti possono eliminare configurazioni obsolete o non più necessarie.

### 4.1.2 Requisiti di Qualità

I requisiti di qualità definiscono i criteri che il sistema deve soddisfare per assicurare prestazioni, affidabilità e conformità agli standard aziendali.

Questi requisiti sono fondamentali per assicurare che il prodotto finale non solo funzioni correttamente, ma sia anche robusto, ben documentato e facilmente estendibile o manutenibile nel tempo.

Ogni requisito di qualità è identificato da un codice univoco, che ne facilita la tracciabilità e la gestione.

I requisiti sono classificati come segue:

- **RQO (Requisito di Qualità Obbligatorio)**: definisce aspetti essenziali che devono essere rispettati per garantire il successo della *PoC*.
- **RQF (Requisito di Qualità Facoltativo)**: descrive caratteristiche non essenziali, ma che migliorano l'esperienza complessiva del sistema.
- **RQD (Requisito di Qualità Desiderabile)**: requisiti auspicabili che aggiungerebbero valore al sistema senza rappresentare vincoli imprescindibili.

#### RQO1 – Documentazione Tecnica

Il sistema deve includere una documentazione tecnica dettagliata, denominata *Specifiche Tecnica*, che illustri le scelte implementative e progettuali adottate.

Questa documentazione è fondamentale per garantire la trasparenza e la comprensione delle soluzioni sviluppate, oltre a fornire un riferimento per future evoluzioni del sistema e per una possibile reimplementazione.

La documentazione deve includere:

- **Descrizione dell'Architettura**: un'illustrazione chiara della struttura generale del sistema, evidenziando i componenti principali e le loro interazioni.
- **Motivazione delle Scelte**: una spiegazione dettagliata delle decisioni architettoniche e delle tecnologie selezionate.
- **Metodi di Personalizzazione e Integrazione**: una descrizione delle personalizzazioni che ho adottato.

#### RQO2 – Manuale Utente

Deve essere fornita una documentazione denominata *Manuale Utente*, progettata per assistere gli utenti nell'utilizzo del sistema.

Questo manuale deve essere chiaro, intuitivo e dettagliato, in modo da facilitare la comprensione e l'uso delle funzionalità offerte.

Il *Manuale Utente* deve includere:

- **Guida all'Installazione e Configurazione**: una descrizione passo-passo delle operazioni necessarie per installare e configurare il sistema.
- **Descrizione delle Funzionalità**: spiegazioni dettagliate delle funzionalità disponibili, accompagnate da esempi pratici per un utilizzo ottimale.
- **Procedure Risolutive**: una sezione dedicata alla risoluzione di problemi comuni e alle [Frequently Asked Questions \(FAQ\)](#), per consentire agli utenti di risolvere autonomamente eventuali difficoltà.

**RQO3 – Manuale Admin**

Il sistema deve includere un *Manuale Admin*, progettato per fornire agli amministratori una guida completa alla gestione e configurazione della parte *web* del sistema.

Questo manuale deve essere chiaro e dettagliato, in modo da consentire una gestione efficiente e sicura delle funzionalità avanzate.

**RQF1 – Struttura e Manutenibilità del Codice**

Il codice sorgente del sistema deve essere ben strutturato, leggibile e facilmente manutenibile, al fine di agevolare futuri interventi di sviluppo o correzione.

Per raggiungere questo obiettivo, devono essere adottate le seguenti pratiche:

- **Standard di Codifica:** adozione di convenzioni di codifica riconosciute, per garantire coerenza e leggibilità.
- **Commenti e Documentazione:** inserimento di commenti chiari e concisi per spiegare le funzioni e le logiche principali del codice.

**RQD5 – Gestione delle Eccezioni nelle Richieste API**

Il sistema deve implementare un meccanismo robusto per la gestione degli errori che possono verificarsi durante le richieste *API*, riducendo al minimo l'impatto sugli utenti e garantendo la continuità operativa.

Gli aspetti chiave di questo requisito includono:

- **Messaggi di Errore Informativi:** restituzione di messaggi di errore chiari, che spieghino la causa del problema e forniscano suggerimenti su come risolverlo.
- **Continuità Operativa:** implementazione di meccanismi di resilienza per garantire che il sistema continui a funzionare anche in presenza di errori *API*, minimizzando l'interruzione dei servizi.

**RQF2 – Pagine Web Autoesplicative**

Il sistema deve fornire pagine *web* autoesplicative che illustrino chiaramente il funzionamento e le funzionalità principali dell'applicazione.

Questa soluzione consente agli utenti di comprendere rapidamente come utilizzare il sistema senza dover consultare ogni volta il manuale utente.

### 4.1.3 Requisiti di Vincolo

I requisiti di vincolo definiscono le restrizioni e i limiti che il sistema deve rispettare per operare in modo conforme alle aspettative aziendali, normative e tecnologiche.

Ogni requisito di vincolo è identificato da un codice univoco per facilitarne la tracciabilità e la gestione.

I requisiti sono classificati come segue:

- **RVO (Requisito di Vincolo Obbligatorio)**: restrizioni fondamentali che il sistema deve rispettare senza eccezioni.
- **RVF (Requisito di Vincolo Facoltativo)**: vincoli opzionali che possono migliorare il progetto ma non sono obbligatori.
- **RVD (Requisito di Vincolo Desiderabile)**: vincoli auspicabili che, se rispettati, arricchiscono la qualità del sistema senza essere indispensabili.

#### RVO1 – Sistema Operativo

Il sistema deve essere progettato per funzionare su *Windows 10*, garantendo piena compatibilità con l'ambiente operativo standard dell'azienda. Questo requisito è obbligatorio per assicurare l'integrazione con le infrastrutture esistenti e rispettare le politiche aziendali.

#### RVO2 – Linguaggio di Programmazione

Il sistema deve essere sviluppato utilizzando *Java* con il *framework Spring Boot*, in modo da favorire il riutilizzo di competenze e risorse aziendali già esistenti.

## 4.2 Tracciamento dei requisiti

A seguito dell'analisi dei requisiti, è stata redatta una tabella che traccia ogni requisito associandolo alla tipologia e alla priorità.

La codifica utilizzata per ogni requisito segue il formato R(F/Q/V)(O/D/Fa), dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

O = obbligatorio

D = desiderabile

Fa = facoltativo

Le informazioni relative ai requisiti e alla loro classificazione sono illustrate nelle seguenti tabelle:

- Tabella 4.1: Requisiti funzionali
- Tabella 4.2: Requisiti qualitativi
- Tabella 4.3: Requisiti di vincolo

Requisito	Descrizione
RFO1	Autenticazione con Taiga.
RFO2	Autenticazione con Planner.
RFO3	Gestione del rinnovo dei token di accesso.
RFO4	Sincronizzazione dei progetti.
RFO5	Sincronizzazione dei piani.
RFO6	Definizione della sincronizzazione tra piani e progetti.
RFO7	Esecuzione periodica delle funzioni di sincronizzazione.
RFO8	Recupero delle informazioni di dettaglio di un piano.
RFO9	Recupero delle informazioni di dettaglio di un progetto.
RFO10	Identificazione delle task relative a user story o issue.
RFO11	Monitoraggio delle modifiche alle BugTask.
RFO12	Monitoraggio delle modifiche a user story e issue.
RFO13	Gestione delle modifiche alle BugTask.
RFO14	Gestione delle modifiche alle Regulartask.
RFO15	Gestione delle modifiche alle user story.
RFO16	Gestione delle modifiche alle issue.
RFO17	Creazione di user story.
RFO18	Creazione di issue.
RFO19	Scrittura di azioni su un File di Log.
RFFa1	Scrittura di errore su File di Log.
RFO21	Estrazione nome utente che ha modificato lo stato della User Story.
RFO22	Estrazione del nome utente che ha modificato una Task.
RFD2	Gestione tramite interfaccia web.
RFD3	Personalizzazione dei gruppi.
RFD4	Modalità di aggiornamento completa.
RFO24	Salvataggio delle associazioni.
RFFa4	Gestione delle associazioni Piano-Progetto esistenti.

Tabella 4.1: Requisiti funzionali

Requisito	Descrizione
RQO1	Documentazione Tecnica.
RQO2	Manuale Utente.
RQO3	Manuale Admin.
RQFA1	Struttura e Manutenibilità del Codice.
RQD5	Gestione delle Eccezioni nelle Richieste API.
RQF2	Pagine Web Autoesplicative.

Tabella 4.2: Requisiti qualitativi

Requisito	Descrizione
RVO1	Sistema Operativo.
RVO2	Linguaggio di Programmazione.

Tabella 4.3: Requisiti di vincolo

## Capitolo 5

# Progettazione e Codifica

### 5.1 Autenticazione e Autorizzazione con Azure

Per consentire alla *PoC* di interagire in modo sicuro con le *API* di *Microsoft Planner* e *SharePoint*, è stato necessario configurare un meccanismo di autenticazione e autorizzazione basato su *Azure Active Directory*.

Questo approccio ha permesso di ottenere un *token* applicativo per eseguire operazioni in *backend* senza intervento umano diretto.

L'intero processo è stato gestito attraverso il portale di *Azure* per garantire sicurezza e conformità agli *standard* aziendali.

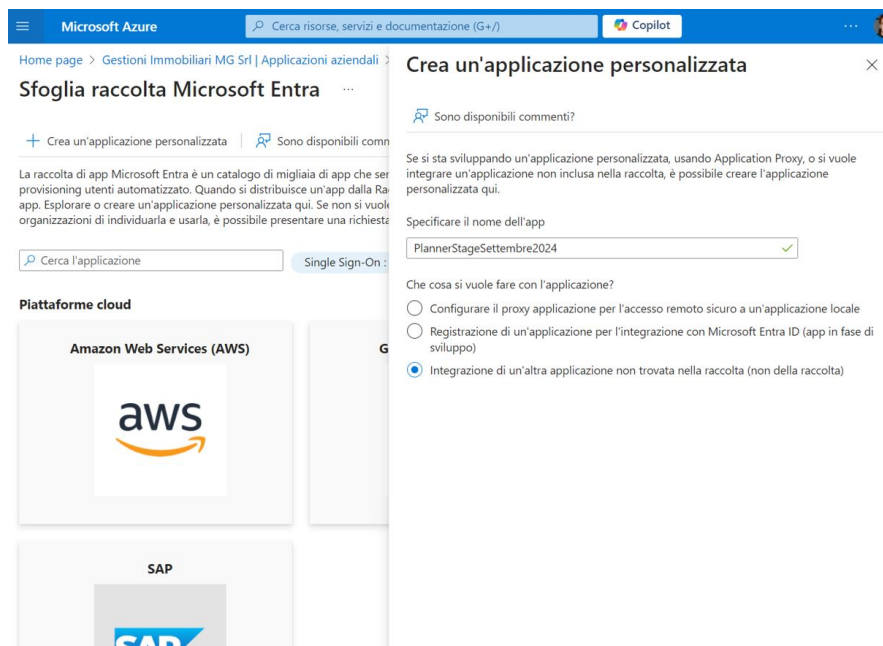
Il processo si è articolato nei seguenti passi:

1. Registrazione di un'applicazione su *Azure Active Directory*.
2. Configurazione delle credenziali di accesso, inclusa la generazione del [Client Secret](#).
3. Assegnazione delle autorizzazioni richieste per l'accesso alle risorse *Planner* e *SharePoint*.

Per tutte le attività di sviluppo, è stato utilizzato un *tenant* aziendale dedicato al *testing* ([@wintechsviluppo](#)), separato dal *tenant* di produzione ([@wintech](#)).

Questa separazione garantisce un ambiente sicuro per *test* e verifiche senza rischio di impatti sui dati reali.

### 5.1.1 Registrazione dell'Applicazione



**Figura 5.1:** Creazione di un'applicazione personalizzata su *Azure Active Directory*

La registrazione dell'applicazione, mostrata in figura 5.1, è stata eseguita accedendo come amministratore alla sezione *Applicazioni Aziendali* del portale di *Azure*.

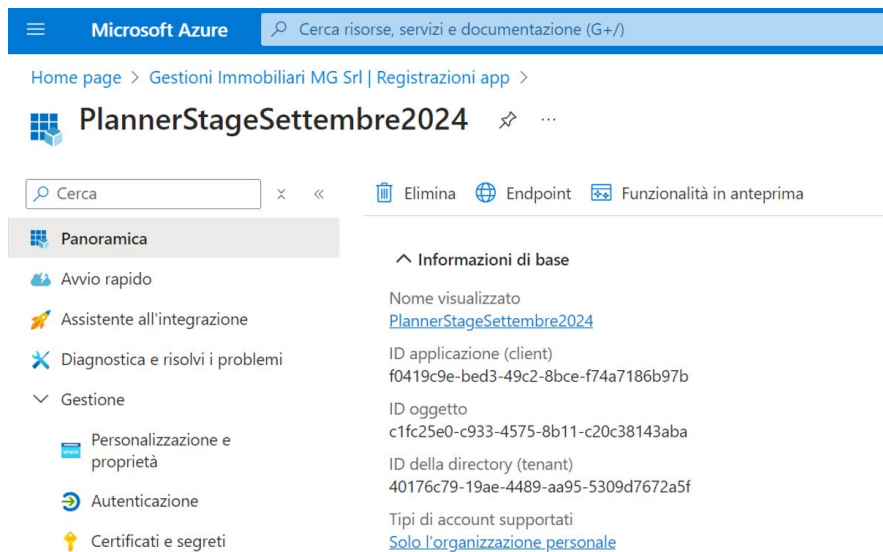
Durante la procedura, sono state richieste le seguenti informazioni principali:

- **Tipo di applicazione:** È stata selezionata l'opzione per un'applicazione non inclusa nella raccolta.
- **Nome dell'applicazione:** Impostato su `PlannerStageSettembre2024`.

Una volta registrata, l'applicazione è risultata visibile nella sezione *Registrazioni App* del portale di *Azure*.



### 5.1.2 Configurazione delle Credenziali e Dettagli dell'Applicazione



**Figura 5.2:** Panoramica dell'applicazione registrata su *Azure*

Dopo la registrazione, i dettagli principali dell'applicazione, mostrati in figura 5.2, sono visibili nella sezione *Panoramica* del portale di *Azure*.

Tra questi si trovano il **Client ID**, il **Tenant ID** e l'**Object ID**, informazioni chiave per l'integrazione con le *API* di *Microsoft Graph*.

#### Creazione del Client Secret

Il *client secret*, talvolta definito "*password applicativa*", è una stringa utilizzata per identificare l'applicazione in sostituzione di un certificato, facilitando l'autenticazione *server-to-server* senza intervento umano.

Sebbene i *client secret* siano meno sicuri rispetto ai certificati, rappresentano una soluzione efficace per lo sviluppo locale o per ambienti non di produzione.

La procedura di creazione del *client secret* è stata la seguente:

- Nella sezione *Certificates & Secrets* dell'applicazione registrata, è stato selezionato il pulsante *New Client Secret*.
- È stata aggiunta una descrizione esplicativa per il *client secret*, specificando la sua funzione.
- È stata impostata una scadenza di 12 mesi, come raccomandato da *Microsoft*. Si sottolinea che la durata massima consentita per un *client secret* è di 24 mesi.
- Dopo la conferma, il valore del *client secret* è stato salvato in modo sicuro, poiché non è più recuperabile in seguito.

Il *client secret* è stato generato con validità fino al 6 settembre 2025.

Questa credenziale è essenziale per ottenere i *token* di accesso richiesti per autenticare le chiamate alle *API* di *Microsoft Graph*.

### 5.1.3 Autorizzazioni per l'Accesso alle API



**Figura 5.3:** Scelta tra autorizzazioni delegate e autorizzazioni applicazione

Le operazioni previste dalla *PoC* richiedono autorizzazioni specifiche per l'accesso alle risorse di *Microsoft Planner* e *SharePoint*.

La configurazione è avvenuta nella sezione *Autorizzazioni API* del portale di *Azure*. Sono state configurate autorizzazioni a livello applicativo, che consentono operazioni *server-to-server* senza intervento umano, come mostrato in figura 5.3.

Tra le autorizzazioni assegnate (vedi figura 5.4):

- `Group.Create`
- `Group.Read.All`
- `Group.ReadWrite.All`
- `Task.Read.All`
- `Task.ReadWrite.All`

Autorizzazioni configurate

Le applicazioni sono autorizzate a chiamare le API quando ottengono autorizzazioni da utenti/amministratori come parte del processo di consenso. L'elenco di autorizzazioni configurate deve includere tutte le autorizzazioni necessarie per l'applicazione. [Altre informazioni sulle autorizzazioni e sul consenso](#)

+ Aggiungi un'autorizzazione    ✓ Concedi consenso amministratore per Gestioni Immobiliari MG Srl

Nome dell'API/autorizzazio...	Tipo	Descrizione	Consenso amministr...	Stato
Microsoft Graph (5)				
Group.Create	Applicazio...	Create groups	Si	✓ Concesso per Gestioni i... ...
Group.Read.All	Applicazio...	Read all groups	Si	✓ Concesso per Gestioni i... ...
Group.ReadWrite.All	Applicazio...	Read and write all groups	Si	✓ Concesso per Gestioni i... ...
Tasks.Read.All	Applicazio...	Read all users' tasks and tasklist	Si	✓ Concesso per Gestioni i... ...
Tasks.ReadWrite.All	Applicazio...	Read and write all users' tasks and tasklists	Si	✓ Concesso per Gestioni i... ...

**Figura 5.4:** Autorizzazioni scelte per l'applicazione registrata

Queste autorizzazioni sono necessarie per la gestione di gruppi (*group-id*), piani (*plans*) e attività (*tasks*) all'interno di *Microsoft Planner*.

## 5.2 Servizio di Autenticazione con OAuth 2.0

L'autenticazione per accedere alle *API* di *Microsoft Graph* è stata implementata utilizzando il protocollo *OAuth 2.0*.

Questo protocollo consente alle applicazioni di ottenere un *token* applicativo che autentica e autorizza le richieste alle risorse protette, senza necessità di un intervento diretto dell'utente.

### 5.2.1 Panoramica del Protocollo OAuth 2.0

*OAuth 2.0* è uno *standard* aperto per la delega sicura dell'accesso alle risorse.

È progettato per applicazioni distribuite e include diversi *grant types* (tipologie di flusso di autorizzazione).

Nella mia implementazione, è stato utilizzato il **Client Credentials Grant**, ideale per scenari *server-to-server*, in cui non è richiesta l'interazione con un utente finale.

Questo flusso prevede:

- L'applicazione richiede un *access token* direttamente al *server* di autorizzazione, autenticandosi con il proprio *client-id* e *client secret*.
- Il *server* di autorizzazione restituisce un *token* che rappresenta i permessi dell'applicazione (vedi figura 5.4) e consente l'accesso alle risorse configurate.

### 5.2.2 Processo di Ottenimento del Token Applicativo

Per accedere alle *API* di *Microsoft Graph*, è stato necessario seguire una procedura precisa per ottenere un *access token* tramite l'*endpoint* di *Azure Active Directory*.

#### Identificazione dell'Endpoint

L'*endpoint* per richiedere il *token* è specifico del *tenant* e si ottiene dal *tenant-id*.

Il formato è il seguente:

```
https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token
```

#### Configurazione della Richiesta

Per ottenere il *token*, è stata inviata una richiesta *HTTP POST* (vedi figura 5.5) con i seguenti parametri nella *body*:

- **grant\_type**: `client_credentials`. Specifica il tipo di autorizzazione richiesta.
- **client\_id**: Identificativo unico dell'applicazione.
- **client\_secret**: Credenziale segreta generata durante la registrazione dell'applicazione.
- **scope**: Impostato su `https://graph.microsoft.com/.default`, che indica le autorizzazioni configurate nel portale *Azure*.

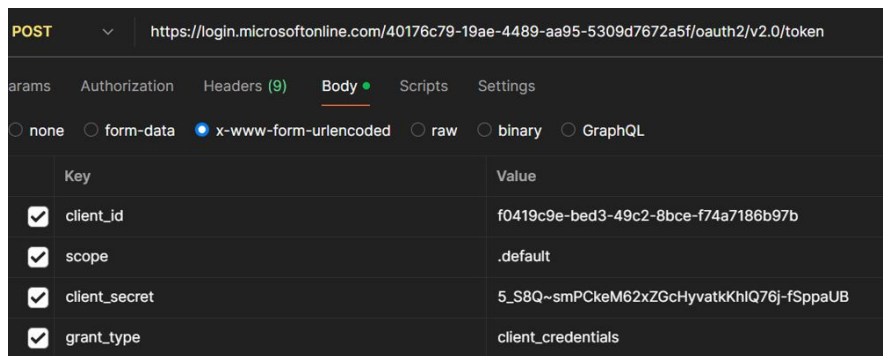


Figura 5.5: Richiesta di *token microsoft*

### Risposta del Server

In caso di successo, il *server* di autorizzazione restituisce una risposta JSON (vedi figura 5.6) contenente:

- `token_type`: Tipologia di *token*, solitamente [Bearer](#).
- `expires_in`: Tempo in secondi prima della scadenza del *token*.
- `access_token`: Il *token* effettivo, utilizzato per autorizzare le richieste alle *API*.

Esempio di risposta:

```
{
  "token_type": "Bearer",
  "expires_in": 3599,
  "ext_expires_in": 3599,
  "access_token":
    "eyJ0eXAiOiJKV1QiLCJub25jZSI6ImwxR2xlcjBwOU5TTGR0eGpobDIzWVREOVQ2UXNGeDdqYVYtbEi5dyIsImtpZCI6Ikg5bmo1OU9Tc3dNcGhnMVNGeDdqYVYtbEi5dyJ9."
}
```

Figura 5.6: Risposta alla richiesta di *token*

### Esempio di Codice per l'ottenimento del Token Microsoft

La chiamata *API* per ottenere il *token applicativo* di *Microsoft* è stata implementata nel seguente metodo:

**Listing 5.1:** Metodo per l'ottenimento del token di autenticazione

```
public Mono<String> getMicrosoftToken(String tenantId, String
    clientId, String clientSecret) {

    WebClient authWebClient = WebClient.builder()
        .baseUrl("https://login.microsoftonline.com/" +
            tenantId + "/oauth2/v2.0")
        .build();

    return authWebClient.post()
        .uri("/token")
        .contentType(MediaType.APPLICATION_FORM_URLENCODED)
        .body(BodyInserters.fromFormData("client_id",
            clientId)
            .with("scope", ".default")
            .with("client_secret", clientSecret)
            .with("grant_type", "client_credentials"))
        .retrieve()
        .onStatus(status -> status.is4xxClientError() ||
            status.is5xxServerError(),
            clientResponse -> Mono.error(new
                RuntimeException("Errore durante la
                    richiesta del token di Microsoft")))
        .bodyToMono(String.class)
        .map(response -> {
            try {
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonNode = mapper.readTree(response)
                    ;
                return jsonNode.get("access_token").asText();
            } catch (Exception e) {
                throw new RuntimeException("Errore durante il
                    parsing del token di Microsoft", e);
            }
        });
}
```

### Descrizione dei Metodi di Autenticazione

Il processo di autenticazione è gestito dal metodo `getMicrosoftToken()`, che invia una richiesta *POST* all'endpoint di *Microsoft* per ottenere il *token*.

I parametri `client_id`, `client_secret`, `scope`, e `grant_type` vengono passati nel corpo della richiesta.

La risposta del server, in formato *JSON*, contiene il *token* di autenticazione.

Il valore del *token* viene estratto dal metodo `getMicrosoftToken()` tramite il *parsing* del *JSON*.

In caso di errore nella richiesta, come credenziali non valide, viene lanciata un'eccezione con un messaggio descrittivo.

### Uso di WebClient e Mono<String>

Il `WebClient` di *Spring WebFlux* è stato scelto per effettuare le chiamate *HTTP* alle *API* di *Taiga* in combinazione con il paradigma reattivo offerto da `Mono`.

Il risultato di una chiamata effettuata tramite `WebClient` viene incapsulato in un oggetto `Mono`, che rappresenta un dato singolo prodotto in un momento successivo.

Questo paradigma consente di gestire in modo efficace sia scenari asincroni sia scenari sincroni:

- **Esecuzione Asincrona:** In contesti dove la risposta non è necessaria immediatamente, il `Mono` permette di continuare il flusso di esecuzione senza bloccare il *thread*, migliorando l'efficienza del sistema.
- **Esecuzione Sincrona su Richiesta:** In situazioni dove il dato è immediatamente necessario, come nel caso del *token* di autenticazione, il `Mono` può essere convertito in un dato sincrono tramite il metodo `block()`. Questa funzionalità rende l'approccio flessibile, consentendo l'integrazione di flussi asincroni e sincroni nello stesso sistema.

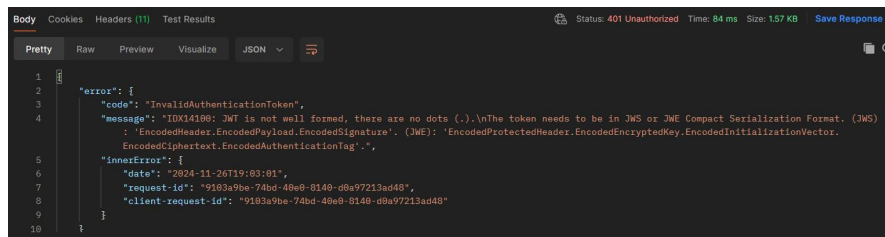
### 5.2.3 Uso del Token per le Richieste alle API di Microsoft

Il *token* viene inviato nell'intestazione *Authorization* delle richieste *HTTP* alle *API* di *Microsoft*, nel formato:

```
Authorization: Bearer <access-token>
```

Le *API* verificano il *token* prima di consentire l'accesso alle risorse richieste.

In caso di *token* non valido o scaduto, viene restituito un errore `401 Unauthorized`, come mostrato in figura 5.7.



```
Body Cookies Headers (11) Test Results Status: 401 Unauthorized Time: 84 ms Size: 1.57 KB Save Response
Pretty Raw Preview Visualize JSON
1
2
3
4
5
6
7
8
9
10
{"error": {
  "code": "InvalidAuthenticationToken",
  "message": "IDX10109: JWT is not well formed, there are no dots (.)\n\nThe token needs to be in JWS or JWE Compact Serialization Format. (JWS) : 'EncodedHeader.EncodedPayload.EncodedSignature'. (JWE): 'EncodedProtectedHeader.EncodedEncryptedKey.EncodedInitializationVector.EncodedCiphertext.EncodedAuthenticationTag'.",
  "innerError": {
    "date": "2024-11-26T19:03:01",
    "request-id": "9183a9be-74bd-48e0-8148-d8a97213ad48",
    "client-request-id": "9183a9be-74bd-48e0-8148-d8a97213ad48"
  }
}}
```

Figura 5.7: Esempio di risposta da un'API Microsoft in caso di *token* non valido o scaduto

### 5.2.4 Possibili Risposte di Errore delle API

Durante l'interazione con le API di *Microsoft Graph* o *Taiga*, ho riscontrato e compreso una varietà di errori.

Di seguito sono elencati i principali codici di stato *HTTP* che possono essere restituiti, con una spiegazione delle cause e suggerimenti per la risoluzione:

- **400 Bad Request:** La richiesta inviata al server è malformata o contiene parametri errati o mancanti. *Cause comuni:*
  - Formato non valido della richiesta (es. *JSON* errato).
  - Mancanza di parametri obbligatori.

*Soluzione:* Verificare la correttezza della struttura della richiesta e la presenza di tutti i parametri richiesti.

- **401 Unauthorized:** Il client non è autorizzato ad accedere alla risorsa richiesta. Questo può verificarsi in caso di:
  - *token* non valido, scaduto o mancante.
  - Credenziali di autenticazione errate.

*Soluzione:* Assicurarsi che il *access token* sia valido e aggiornato, e che le credenziali di autenticazione siano corrette.

- **403 Forbidden:** L'accesso alla risorsa è vietato, nonostante il *client* sia autenticato. *Cause comuni:*
  - Permessi insufficienti (*scope* non configurato correttamente).
  - Configurazione delle autorizzazioni nel portale *Azure* non conforme ai requisiti della chiamata.

*Soluzione:* Controllare e aggiornare le autorizzazioni necessarie per la risorsa richiesta.

- **500 Internal Server Error:** Un errore generico lato server ha impedito l'elaborazione della richiesta. *Cause comuni:*
  - Problemi temporanei o malfunzionamenti del *server*.
  - *Bug* lato *server* nell'elaborazione della richiesta.

*Soluzione:* Riprovare dopo alcuni minuti. Se il problema persiste, contattare il supporto tecnico o consultare la documentazione ufficiale.

## 5.3 Autenticazione tramite API di Taiga

Per interagire con le *API* di *Taiga*, è stato implementato un sistema di autenticazione che utilizza un utente tecnico.

Questo utente, configurato appositamente con credenziali predefinite, consente all'applicazione di ottenere un *token* delegato per eseguire azioni a nome dell'applicazione stessa.

L'utente tecnico deve essere presente in ogni progetto aziendale per garantire l'accesso e la visibilità sia ai progetti stessi sia a tutte le informazioni e risorse in essi contenute tramite le *API*.

### 5.3.1 Flusso di Autenticazione

Il flusso di autenticazione con *Taiga* si basa su una chiamata *POST* alle *API*, fornendo *email*, *password* e tipo di utente.

Il risultato è un *access token*, utilizzabile per autenticare le successive richieste.

#### Endpoint di Autenticazione

L'endpoint per l'autenticazione è il seguente:

```
POST https://api.taiga.io/api/v1/auth
```

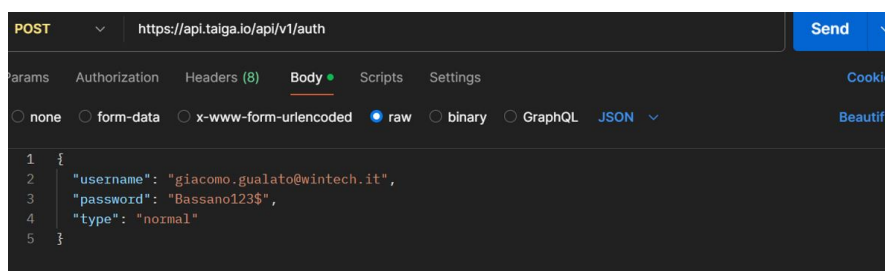
#### Parametri della Richiesta

La richiesta inviata al *server* include i seguenti parametri nel *body* in formato *JSON*:

- **username**: L'indirizzo *email* dell'utente tecnico.
- **password**: La *password* associata all'utente tecnico.
- **type**: Impostato su **normal**, per indicare che l'utente è un normale *account* di accesso.

#### Esempio di Richiesta con Postman

Nella figura 5.8 viene illustrata la chiamata all'endpoint di autenticazione tramite Postman, con i parametri necessari inclusi nel *body* della richiesta in formato *JSON*.



**Figura 5.8:** Esempio di chiamata POST per ottenere il token di autenticazione



### Esempio di Risposta del Server

La figura 5.9 mostra la risposta restituita dal *server* di *Taiga* a seguito della richiesta di autenticazione.

Nel risultato *JSON* è presente il campo `auth_token`, che rappresenta il token di accesso necessario per autenticare le successive richieste alle *API*.

```
{
  "accepted_terms": true,
  "auth_token": "eyJ1c2VyX2F1dGh1bnRyY2F0aw9uX2lkIjoxNn0:1jrHFG:QR38NU7tHvPz1_s7BvgwEgnoxIc",
  "big_photo": null,
  "bio": "",
  "color": "#c9f5fe",
  "date_joined": "2020-07-03T08:40:29.738Z",
  "email": "test-register@email.com",
  "full_name": "test",
  "full_name_display": "test",
  "gravatar_id": "1ec29e4d0732b571e9a975e258a7e9b5",
  "id": 16,
  "is_active": true,
  "lang": "",
  "max_memberships_private_projects": null,
  "max_memberships_public_projects": null,
  "max_private_projects": null,
  "max_public_projects": null,
  "photo": null,
  "read_new_terms": false,
  "refresh":
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1b190eXB1IjoicmVmcmVzaCIsImV4cCI6MTYyNzI5OTQzMiwiaWZkZkZkZD1mMD1mNWYwNzAwMTkiLCJ1c2VyX2lkIjo1fQ.vez_-n6y9yQo2uFgXTPB5YdJHFKUIAsCrNVJ29_T3wM",
  "roles": [
    "Front"
  ],
  "theme": "",
  "timezone": "",
  "total_private_projects": 0,
  "total_public_projects": 0,
  "username": "test-username",
  "uuid": "c30015cc735e4b33b008139b58f13791"
}
```

Figura 5.9: Esempio di risposta alla richiesta di autenticazione, contenente il `auth_token`

### Esempio di Codice per la Chiamata API

La chiamata API per ottenere il *token* è stata implementata nel seguente metodo:

Listing 5.2: Metodo per l'ottenimento del token di autenticazione

```
public ApiTaigaService(WebClient.Builder webClientBuilder) {
    String baseUrl = "https://api.taiga.io/api/v1";
    this.authWebClient = webClientBuilder.baseUrl(baseUrl).build
        ();
}

public Mono<String> getAuthToken(String username, String password
    , String type) {
```

```

    return authWebClient.post()
        .uri("/auth")
        .bodyValue(new AuthRequest(username, password, type))
        .retrieve()
        .onStatus(HttpStatusCode::isError, clientResponse ->
            Mono.error(new RuntimeException("Credenziali
                non valide, riprovare")))
        .bodyToMono(String.class)
        .map(this::extractAuthToken);
}

private String extractAuthToken(String json) {
    try {
        JSONObject jsonObject = new JSONObject(json);
        return jsonObject.optString("auth_token", null);
    } catch (Exception e) {
        throw new RuntimeException("Errore durante il parsing del
            JSON", e);
    }
}
}

```

### Descrizione dei Metodi di Autenticazione

Il sistema di autenticazione implementato sfrutta due metodi principali per ottenere e gestire il *token* necessario alle successive richieste *API*.

Il metodo `getAuthToken()` invia una richiesta POST all'endpoint `/auth` del server *Taiga*, includendo le credenziali di autenticazione (`username`, `password`, `type`) nel corpo della richiesta.

La risposta del *server*, come mostrato in Figura 5.9, restituita in formato *JSON*, contiene il *token* di autenticazione.

Per elaborare questa risposta, il metodo privato `extractAuthToken()` effettua il *parsing* del *JSON* ricevuto, estraendo il valore del campo `auth_token`.

### Gestione dei Parametri di Configurazione

I parametri come *email*, *password* e *tipo* vengono recuperati dal *file* di configurazione `application.properties` tramite l'annotazione `@Value` nel codice.

Ad esempio:

**Listing 5.3:** Recupero dei parametri dal file di configurazione

```

@Value("${email}")
private String email;

@Value("${password}")
private String password;

@Value("${type}")
private String type;

```

Esempio di configurazione nel file `application.properties`:

**Listing 5.4:** Esempio di configurazione

```
email=mario.rossi@wtcsviluppo.onmicrosoft.com
password>Password123
type=normal
```

L'utilizzo di un file di configurazione consente di separare le credenziali sensibili e i parametri statici dal codice sorgente, garantendo:

- **Flessibilità:** la possibilità di modificare facilmente i valori senza dover intervenire sul codice.
- **Sicurezza:** i file di configurazione possono essere protetti e criptati, riducendo il rischio di esposizione accidentale delle credenziali.

### 5.3.2 Uso del Token per le Chiamate API

Una volta ottenuto, il `auth_token` viene utilizzato come intestazione di autorizzazione (**Authorization**) per autenticare tutte le richieste successive alle *API* di *Taiga*.

Il formato dell'intestazione è:

```
Authorization: Bearer <auth_token>
```

## 5.4 Approccio Standard per le Chiamate API

A partire da questo punto, tutte le interazioni con le *API* di *Microsoft Planner*, *Microsoft SharePoint* e *Taiga* seguiranno un paradigma uniforme, progettato per garantire efficienza, riutilizzabilità e chiarezza.

L'approccio adottato è strutturato in due fasi principali:

1. **Chiamata API asincrona con WebClient:** Le richieste alle *API* vengono effettuate utilizzando istanze dedicate di `WebClient`, configurate in modo specifico per ciascun sistema.  
Sono stati implementati due `WebClient`: uno per le *API* di *Taiga* e uno per quelle di *Microsoft*. Nel caso delle *API* di *Microsoft*, esiste un'eccezione per le richieste di autenticazione, che utilizzano una *base URL* diversa rispetto alle altre chiamate. Questa separazione consente una gestione chiara e scalabile delle connessioni, mentre l'adozione di un'architettura asincrona riduce i tempi di attesa e ottimizza l'uso delle risorse.
2. **Elaborazione della risposta:** Una funzione dedicata si occupa di analizzare le risposte ricevute, estraendo solo i dati necessari. I campi di interesse, come `id`, `name` o altre informazioni rilevanti, vengono estratti in base al contesto applicativo. Questo approccio facilita il riutilizzo del codice, riduce la complessità e mantiene una logica modulare.

Questa metodologia presenta molteplici vantaggi. Adottare uno stile di implementazione uniforme semplifica la gestione del codice e migliora la modularità, separando la logica di richiesta da quella di elaborazione. Inoltre, l'uso di `WebClient` garantisce prestazioni elevate grazie all'adozione di chiamate asincrone, che riducono i tempi di attesa e favoriscono un'architettura reattiva ed efficiente.

Le sezioni successive illustreranno le interazioni specifiche con le *API*, applicando sempre questo schema generale. Il codice dettagliato per le chiamate *API* non sarà ripetuto, poiché l'approccio rimane invariato, con eventuali adattamenti che verranno descritti in base alle necessità delle risorse o delle operazioni richieste.

## 5.5 Rilevamento delle Modifiche e Sincronizzazione dei Dati

Nel PoC implementato, è stato utilizzato un meccanismo di esecuzione automatica di funzioni periodiche tramite l'annotazione `@Scheduled` di *Spring*. Questa annotazione consente di eseguire automaticamente un metodo a intervalli regolari, eliminando la necessità di invocazioni manuali.

Una funzione periodica è stata definita per raccogliere e aggiornare periodicamente le informazioni relative a progetti, piani, *user story*, *issue*, *RegularTask* e *BugTask*, interagendo con le *API* di *Taiga* e *Microsoft Planner*.

### 5.5.1 Configurazione della Funzione Periodica

La funzione periodica è configurata per iniziare l'esecuzione con un ritardo di 5 secondi dalla prima invocazione e per ripetersi ogni 10 secondi. Il ritardo iniziale garantisce il tempo necessario per recuperare i *token* di accesso alle *API* di *Microsoft Planner* e *Taiga*. L'intervallo di 10 secondi, invece, è stato scelto per consentire un controllo efficace delle modifiche nei dati.

La configurazione è rappresentata nel seguente frammento di codice, dove l'annotazione `@Scheduled` gestisce l'esecuzione automatica della funzione:

**Listing 5.5:** Configurazione della funzione periodica con `@Scheduled`

```
@Scheduled(initialDelay = 5000, fixedRate = 10000)
public void executePeriodicTasks() throws JSONException {
    log.info("Esecuzione del task periodico per ottenere le user
        story, le issue e i task aggiornati");
}
```

#### Descrizione della Funzione `executePeriodicTasks`

La funzione `executePeriodicTasks()` è incaricata di raccogliere, aggiornare e sincronizzare i dati provenienti da *Taiga* e *Microsoft Planner*.

Ogni ciclo di esecuzione segue un flusso strutturato:

- **Recupero dei progetti:** tramite il servizio `apiTaigaService`, la funzione recupera la lista dei progetti disponibili su *Taiga* e ne estrae i dettagli specifici.
- **Determinazione della modalità di aggiornamento:** in base alla configurazione, la funzione decide se eseguire un aggiornamento completo o parziale.
  - In modalità completa, vengono recuperati tutti i *group-id* direttamente da *Microsoft Planner*.
  - In modalità parziale, i *group-id* vengono estratti da un file di configurazione predefinito.

## 5.5. RILEVAMENTO DELLE MODIFICHE E SINCRONIZZAZIONE DEI DATI<sup>52</sup>

- **Recupero dei piani:** per ciascun *group-id*, la funzione ottiene i relativi piani utilizzando le *API* di *Microsoft Planner*.
- **Confronto dei dati:** i dati attuali vengono confrontati con quelli salvati in precedenza (*dati precedenti*) per identificare eventuali modifiche. Questo confronto avviene per ciascun tipo di dato:
  - *RegularTask*: confronto dei *RegularTask* salvati con quelli recuperati.
  - *BugTask*: confronto analogo per i *task* di tipo "bug".
  - *User Story*: confronto tra le *user story* salvate in precedenza e quelle attuali.
  - *Issue*: confronto tra le *issue* precedenti e quelle nuove.

In caso di modifiche, viene eseguita una funzione di analisi per aggiornare i dati e riflettere le variazioni rilevate. Ogni tipo di dato (*task*, *user story*, *issue*) viene gestito separatamente per garantire una logica specifica e scalabile.

- **Aggiornamento dei dati precedenti:** al termine della sincronizzazione, i dati precedenti vengono aggiornati con quelli più recenti. Questo consente di avere una base di confronto accurata per il successivo ciclo di esecuzione.

Nelle sezioni successive verranno analizzati più nel dettaglio i singoli aspetti e le tecniche utilizzate per ciascun tipo di dato gestito durante il processo di sincronizzazione.

## 5.6 Recupero dei Progetti da Taiga

Il recupero dei progetti da *Taiga* rappresenta uno step fondamentale per la sincronizzazione con *Microsoft Planner*. Attraverso un flusso ben definito, vengono ottenuti i progetti associati all'utente tecnico autenticato.

L'interazione con le *API* di *Taiga* segue un approccio *standard*, garantendo chiarezza, modularità e riutilizzabilità.

### 5.6.1 Flusso di Recupero

Il processo per ottenere i progetti prevede i seguenti passaggi principali:

- Autenticazione tramite *token*, necessaria per accedere in modo sicuro alle *API* di *Taiga*.
- Recupero dell'**ID utente tecnico**, che consente di identificare i progetti di cui è membro.
- Chiamata all'*endpoint* dedicato per ottenere la lista dei progetti.
- Analisi della risposta *JSON* per estrarre i campi rilevanti, come **id**, **name** e **slug**.

### 5.6.2 Recupero dell'ID Utente tecnico

Il recupero dell'**ID utente tecnico** è il primo passo per accedere ai progetti associati all'utente tecnico. L'*ID* dell'utente tecnico è un identificativo univoco utilizzato per filtrare i progetti di cui è membro e accedere ai relativi dettagli.

#### Endpoint Utilizzato

L'*endpoint* per ottenere l'*ID* dell'utente tecnico è il seguente:

```
GET https://api.taiga.io/api/v1/users/me
```

La chiamata richiede l'inclusione dell'intestazione **Authorization**, che deve contenere il *token* di autenticazione preceduto dal prefisso **Bearer**.

### Risposta del Server

La risposta del *server* restituisce un oggetto *JSON* contenente informazioni sull'utente.

Nella documentazione ufficiale di *Taiga* viene riportato un esempio di risposta come riportato in Figura 5.10.

## 47.5. User detail

```
{
  "accepted_terms": true,
  "big_photo": null,
  "bio": "",
  "color": "#40826D",
  "date_joined": "2020-07-02T11:56:19.209Z",
  "email": "user2114747470430251528@taigaio.demo",
  "full_name": "Vanesa Torres",
  "full_name_display": "Vanesa Torres",
  "gravatar_id": "b579f05d7d36f4588b11887093e4ce44",
  "id": 6,
  "is_active": true,
  "lang": "",
  "max_memberships_private_projects": null,
  "max_memberships_public_projects": null,
  "max_private_projects": null,
  "max_public_projects": null,
  "photo": null,
  "read_new_terms": false,
  "roles": [
    "Design",
    "Front",
    "Product Owner",
    "UX"
  ],
  "theme": "",
  "timezone": "",
  "total_private_projects": 4,
  "total_public_projects": 4,
  "username": "patchedusername",
  "uuid": "e9296d489b6a42d79048df2f3789b396"
}
```

Figura 5.10: Risposta dell'*endpoint* per ottenere l'*ID* utente dalla documentazione di *Taiga*.

### Analisi della Risposta

Dalla risposta *JSON*, viene estratto esclusivamente il campo *id*, che rappresenta l'identificativo univoco dell'utente tecnico.

- **Motivazione:** L'*ID* è indispensabile per tutte le chiamate successive, poiché permette di identificare i progetti associati all'utente tecnico. Gli altri campi (*full\_name*, *email*, ecc.) non sono necessari in questo contesto specifico e vengono ignorati per mantenere il processo semplice e focalizzato.
- **Modularità:** L'estrazione limitata a questo campo riduce la complessità della gestione dei dati e garantisce che il flusso rimanga scalabile.

L'analisi della risposta viene effettuata seguendo l'approccio *standard* per le chiamate *API*, descritto in precedenza, utilizzando una funzione dedicata per estrarre i dati rilevanti dalla risposta.

### 5.6.3 Recupero dei Progetti

Una volta ottenuto l'*ID* utente tecnico, viene eseguita la chiamata per ottenere i progetti associati a tale utente.

#### Endpoint per il Recupero dei Progetti

L'*endpoint* per ottenere i progetti associati all'utente è il seguente:

```
GET https://api.taiga.io/api/v1/projects?member=<userId>
```

#### Parametri della Richiesta

La richiesta richiede l'inclusione dell'intestazione **Authorization**, che deve contenere il *token* di autenticazione preceduto dal prefisso **Bearer**.

Inoltre, il parametro **member** deve essere settato con l'*ID* utente ottenuto in precedenza.

#### Analisi della Risposta

La risposta del server restituisce un oggetto *JSON* contenente informazioni sui progetti associati all'utente.

Tra tutti i dati contenuti in ciascun progetto vengono utilizzati i seguenti campi rilevanti:

- **id:** L'identificativo univoco del progetto.
- **name:** Il nome del progetto.
- **slug:** Un identificativo leggibile utilizzato nelle *URL*. Lo *slug* è particolarmente utile per il collegamento dei progetti all'interno della pagina di amministrazione, ma questa funzionalità verrà approfondita nelle sezioni successive.



### Recupero dei Dettagli del Progetto

Per ciascun progetto, vengono recuperati i seguenti dettagli rilevanti:

- **User Stories:** Rappresentano le attività specifiche legate al progetto.
- **Issues:** Problematiche o difetti riscontrati nel progetto.
- **Severity:** Riferito alle priorità delle *issue* del progetto.

Questi dettagli vengono estratti tramite *endpoint* specifici, che sono descritti nelle sottosezioni seguenti.

#### 5.6.4 Ottenimento delle User Stories e delle Issues

Le *User Story* e le *Issue* rappresentano due delle principali entità gestite in un progetto su *Taiga*. Entrambe consentono di tracciare attività, problemi e funzionalità, ma con finalità e dettagli operativi differenti.

In questo contesto, si è reso necessario sviluppare un meccanismo per il recupero di queste entità tramite le *API* di *Taiga*, garantendo un'integrazione coerente con il sistema di *Planner*. La progettazione si è concentrata sulla creazione di una classe *template* condivisa che potesse rappresentare entrambe le entità, semplificando così la gestione delle informazioni e riducendo la complessità del codice.

#### Definizione della Classe Template per User Stories e Issues

La classe *template*, utilizzata per rappresentare sia le *User Stories* sia le *Issues*, include i seguenti campi principali, come illustrato in Figura 5.11.

```
private Long id; 4 usages
private String version; 4 usages
private String status; 4 usages
private String subject; 4 usages
private String dueDate; 4 usages
private String description; 4 usages
private Double totalPoints; 5 usages
private List<Tag> tags; 4 usages
private int severityid; //Id che s
```

Figura 5.11: Classe *template* condivisa per *User Story* e *Issue* in *Taiga*

### Differenze tra User Stories e Issues

Sebbene le *User Story* e le *Issue* condividano diversi attributi, esistono alcune differenze fondamentali:

- Le **Issues** utilizzano il campo `severityId` della classe per rappresentare la gravità (*severity*), che si allinea bene con le priorità di *Planner*.
- Le **User Stories** non dispongono di un campo nativo per la priorità. Per ovviare a questa mancanza, si è scelto di utilizzare un *tag* dinamico nel formato "priority: <valore\_priorità>", dove i valori possibili sono Low, Medium, Important, Urgent.
- Il campo `totalPoints` è utilizzato solo per le *User Stories* e rimane null per le *Issues*, poiché queste ultime non prevedono l'uso del sistema a punti.

### Motivazioni e Scelte di Progetto

#### Utilizzo del Campo `severityId` per le Issues

Le *Issues* di *Taiga* presentano un campo `severityId`, che consente di rappresentare la gravità delle *issue* in modo strutturato. Questo campo è stato scelto poiché i valori di `severity` in *Taiga* mappano bene con i livelli di priorità di *Planner*, come illustrato di seguito:

- **Taiga Severity:** Wishlist, Minor, Normal, Important, Critical (personalizzabili).
- **Planner Priority:** Low, Medium, Important, Urgent (fissi).

Ad esempio, se in *Planner* una *BugTask* cambia priorità da Low a Medium, è possibile determinare l'*ID* corrispondente alla Medium in *Taiga* consultando la lista delle *severity*, e aggiornare il campo `severityId` dell'*issue* di conseguenza, si veda la Sezione [5.6.5](#)

#### Utilizzo dei Tag Dinamici per le User Stories

Le *User Stories*, al contrario, non dispongono di un campo dedicato per la priorità. Per rappresentare questa informazione, si è scelto di utilizzare un *tag* dinamico nel formato "priority: <valore\_priorità>" (ad esempio, "priority: Medium"). Questa soluzione offre i seguenti vantaggi:

- Evita la necessità di modificare la struttura delle *User Stories* in *Taiga*.
- Permette di aggiungere informazioni personalizzate senza impattare sugli altri campi, sfruttando il fatto che *Taiga* non impone limiti sul numero di *tag* utilizzabili, a differenza di *Planner*, dove il limite è di 26.
- Garantisce una rappresentazione coerente delle priorità tra *Taiga* e *Planner*.

#### Considerazioni sui Punti Totali (`totalPoints`)

Il campo `totalPoints` viene utilizzato solo per le *User Stories* nel momento in cui passano agli stati **Analizzata** e **Stimata**. Per maggiori dettagli sul processo di stima, si veda la Sezione [5.9.3](#).

### Esempio di Richiesta per User Stories e Issues

L'ottenimento delle *user story* e delle *issue* avviene tramite richieste GET agli *endpoint* seguenti:

```
GET https://api.taiga.io/api/v1/userstories?project={projectId}
```

```
GET https://api.taiga.io/api/v1/issues?project={projectId}
```

Entrambe le richieste richiedono il parametro `projectId` e un *token* di autenticazione specificato nell'intestazione.

### Motivazione della Scelta

In alternativa, avrei potuto utilizzare il campo `priority` predefinito di *Taiga*. Tuttavia, i valori predefiniti di `priority` in *Taiga* (`Low`, `Medium`, `High`) non si adattano ai valori fissi di *Planner* (`Low`, `Medium`, `Important`, `Urgent`).

Modificare le priorità predefinite in ogni progetto di *Taiga* avrebbe comportato un rischio significativo di errore e un aumento della complessità di gestione.

Pertanto, ho preferito utilizzare il campo `severityId` per le *Issues*, mantenendo la coerenza con le priorità di *Planner*, e i *tag* dinamici per rappresentare la priorità nelle *User Stories*.

### 5.6.5 Estrazione delle Severity di un Progetto

Le *severity* rappresentano il livello di gravità associato alle *issue* in *Taiga*. Questa informazione è essenziale per garantire una gestione coerente delle priorità, specialmente quando si integra *Taiga* con strumenti come *Planner*.

Ad esempio, nel caso in cui la priorità di una *BugTask* venga modificata in *Planner* (e.g., da `Low` a `Medium`), è necessario aggiornare la *severity* corrispondente nell'*issue* di *Taiga*. Per effettuare questa modifica, è indispensabile sapere quale sia l'*ID* associato al valore `Medium`, in modo da sostituire il precedente *ID* della *severity* con quello corretto. Questa informazione può essere recuperata tramite la lista delle *severity* associate al progetto, utilizzando una ricerca per nome all'interno della lista.

### Endpoint per le Severity del Progetto

L'*endpoint* per ottenere le *severity* di un progetto è il seguente:

```
GET https://api.taiga.io/api/v1/severities?project={projectId}
```

### Parametri della Richiesta

Per effettuare la richiesta, è necessario fornire i seguenti parametri:

- `projectId`: L'identificativo univoco del progetto di cui si desidera recuperare le *severity*.
- `Authorization`: Il *token* di autenticazione.

### Analisi della Risposta

La risposta restituita dall'API è un array di oggetti *JSON*, ognuno dei quali rappresenta una *severity* del progetto. Ogni oggetto include vari campi, ma per le operazioni richieste vengono utilizzati solo i seguenti:

- **id**: L'identificativo numerico univoco della *severity*.
- **name**: Il nome descrittivo della *severity* (ad esempio, "Wishlist", "Medium", "High", "Critical").

La struttura completa della risposta contiene ulteriori dettagli, come mostrato nella seguente Figura 5.12:

```
{
  "color": "#666666",
  "id": 1,
  "name": "Patch name",
  "order": 1,
  "project": 1
}
```

**Figura 5.12:** Struttura completa della risposta JSON contenente le *severity* del progetto.

Per semplicità, vengono estratti solo i campi **id** e **name**, ottenendo una rappresentazione come illustrato in Figura ??:

### Motivazioni

L'estrazione delle *severity* di un progetto è cruciale per consentire modifiche coerenti e precise alle *issue* di *Taiga*. In *Taiga*, il campo *severity* delle *issue* contiene solo l'*ID*, che rappresenta il valore della gravità, ma senza una corrispondenza con il nome, questo valore è incomprensibile.

La lista delle *severity* permette di mappare ogni *ID* al relativo nome descrittivo. Ad esempio, se in *Planner* la priorità di una *task* cambia da **Low** a **Medium**, è possibile identificare, tramite la lista delle *severity*, che l'*ID* corrispondente a **Medium** è 31355. Questo valore può quindi essere utilizzato per aggiornare correttamente la *severity* della *issue* corrispondente in *Taiga*.

Grazie a questa mappatura, è possibile mantenere sincronizzate le priorità tra i due strumenti, riducendo il rischio di incongruenze e migliorando l'efficienza nella gestione delle attività.

## 5.7 Determinazione della Modalità di Aggiornamento

La sincronizzazione tra i gruppi (`group-id`) di *Microsoft Planner* e il sistema può avvenire attraverso due modalità principali: **completa** o **parziale**. Questa configurazione è regolabile tramite un *switch* nell'interfaccia amministrativa, che sarà approfondita nella sezione dedicata all'amministrazione.

### 5.7.1 Associazione tra Piani e Gruppi in Microsoft Planner

In *Planner*, ogni piano è associato a un unico gruppo (`group-id`), che funge da contenitore per le attività condivise.

È possibile assegnare un piano a un gruppo esistente o crearne uno nuovo durante la configurazione iniziale. È importante evidenziare che:

- Un piano può essere associato a un solo gruppo.
- Un gruppo può contenere uno o più piani.

### 5.7.2 Modalità di Aggiornamento Completa

La modalità completa permette di sincronizzare tutti i gruppi presenti nel tenant corrente, ad esempio `@wtcsviluppo.onmicrosoft.com`.

Questa modalità utilizza le *API* di *Microsoft Graph* per recuperare in modo automatico l'elenco di tutti i gruppi, garantendo una copertura totale.

È particolarmente utile in fase di avvio del sistema o per sincronizzazioni periodiche, in quanto assicura che nessun gruppo venga escluso.

Tuttavia, questa modalità può risultare più onerosa in termini di tempo e risorse, soprattutto in contesti con un numero elevato di gruppi.

### 5.7.3 Modalità di Aggiornamento Parziale

La modalità di aggiornamento parziale consente di sincronizzare un sottoinsieme specifico di gruppi (`group-id`) attraverso un *file* di configurazione predefinito. Questo approccio è particolarmente utile per scenari in cui non è necessario aggiornare l'intero *tenant*, permettendo di ridurre il carico di elaborazione e ottimizzare i tempi di sincronizzazione.

#### Recupero del File di Configurazione da SharePoint

Il *file* di configurazione, denominato tipicamente `groupIds.txt`, è archiviato in una cartella dedicata su *SharePoint*.

Per accedere a questo *file*, il sistema esegue una serie di operazioni sequenziali che sfruttano le *API* di *Microsoft Graph* per ottenere l'elenco dei *file* disponibili e identificare quello specificato.

**Endpoint per il Recupero dei File** Il recupero dei *file* presenti nella cartella *SharePoint* utilizza il seguente *endpoint* delle *API* di *Microsoft Graph*:

```
GET /drives/{driveId}/items/{folderId}/children
```

Dove:

- **driveId**: Identificativo del *drive* (unità) in cui è memorizzato il *file*. Questo valore viene determinato automaticamente dal sistema.
- **folderId**: Identificativo della cartella contenente i *file* di configurazione.
- **Authorization**: Token di accesso generato per autenticare la richiesta.

La risposta dell'*endpoint* è una lista di oggetti *JSON*, ciascuno dei quali rappresenta un *file* presente nella cartella.

Ogni oggetto include dettagli come l'*id* univoco e il nome del *file*.

### Identificazione del File Specifico

Una volta ottenuta la lista dei *file*, il sistema identifica quello desiderato (`groupIds.txt`) attraverso una ricerca basata sul nome. Se il *file* non viene trovato, il sistema registra un errore e la sincronizzazione parziale avviene ma con i valori del *group-id* precedenti.

Se il *file* è presente, viene utilizzato il suo *id* per recuperare il contenuto utilizzando il seguente *endpoint*:

```
GET /drives/{driveId}/items/{fileId}/content
```

Dove:

- **fileId**: Identificativo univoco del *file* individuato in precedenza.
- **Authorization**: *Token* di accesso generato per autenticare la richiesta.

Questo *endpoint* restituisce il contenuto del *file* come una stringa in formato testo.

### Estrazione dei Group-ID dal File

Il contenuto del *file* `groupIds.txt` è costituito da un elenco di *group-id*, ciascuno su una riga separata. Il sistema analizza il contenuto per estrarre gli identificativi, ignorando eventuali righe vuote o non valide.

Questo elenco di *group-id* è poi utilizzato per eseguire la sincronizzazione parziale dei gruppi.

## 5.8 Recupero dei Piani da Microsoft Planner

Come descritto nella sezione precedente, il sistema ottiene i `groupId` validi utilizzando un meccanismo automatizzato di aggiornamento basato sul *file* di configurazione `groupIds.txt`.

Il recupero dei piani da *Microsoft Planner* costituisce il passo successivo e permette di acquisire le informazioni essenziali per la sincronizzazione con altri strumenti.

Questa sezione descrive nel dettaglio il processo di recupero dei piani e delle relative informazioni associate, come i *task*, le categorie/*tag* e i *bucket* di ciascun piano.

### 5.8.1 Panoramica del Processo

Il processo di recupero dei piani da *Microsoft Planner* si articola in tre fasi principali:

1. Recupero dei piani associati a ciascun `groupId`.
2. Recupero dei *task* associati a ciascun piano.
3. Recupero delle categorie/*tag* e dei *bucket* associati a ciascun piano.

### 5.8.2 Recupero dei Piani per Tutti i Gruppi

Il recupero dei piani avviene iterando su tutti i `groupId` disponibili. Per ogni `groupId`, viene effettuata una richiesta *API* verso un *endpoint* dedicato per ottenere i piani associati a quel gruppo.

I piani recuperati vengono raccolti in un'unica lista che rappresenta l'insieme completo dei piani disponibili per la sincronizzazione.

### 5.8.3 Recupero dei Piani di un Singolo Gruppo

Per ogni gruppo identificato da un `groupId`, una richiesta *API* specifica consente di ottenere i dettagli dei piani associati.

L'endpoint per recuperare i piani di un singolo gruppo è il seguente:

```
GET /groups/{groupId}/planner/plans
```

La richiesta include il *token* di autenticazione nell'intestazione e restituisce, in risposta, una lista di oggetti *JSON* che rappresentano i piani.

Tra i dati restituiti, vengono utilizzati principalmente:

- **id**: L'identificativo univoco del piano.
- **title**: Il titolo del piano.

Questi dati vengono elaborati per costruire oggetti rappresentativi dei piani, successivamente utilizzati nelle fasi di sincronizzazione.

### 5.8.4 Estrazione dei Piani dalla Risposta API

La risposta restituita dall'*API*, in formato *JSON*, contiene una lista di oggetti rappresentanti i piani. Il sistema elabora questa risposta per estrarre i dati essenziali, come l'identificativo (`id`) e il titolo (`title`), oltre a collegare ogni piano al rispettivo `groupId`.

In caso di errori durante l'elaborazione, il sistema è progettato per generare eccezioni gestibili che ne permettono il *debug* e il monitoraggio.

### 5.8.5 Dettagli Recuperati per ciascun Piano

Per ciascun piano, vengono recuperati i seguenti dettagli rilevanti:

- **Task:** L'elenco completo dei *task* associati al piano. Questi includono sia i *RegularTask* che i *BugTask*.
- **Categorie/Tag:** *Microsoft Planner* supporta un massimo di 26 categorie per ciascun piano. Ogni categoria può essere configurata con un colore specifico o lasciata senza valore (`null`).
- **Bucket:** I *bucket* rappresentano una suddivisione logica dei *task* nel piano. Per ciascun *bucket* vengono recuperati l'identificativo (`id`) e il nome (`name`).

Questi dettagli vengono estratti tramite *endpoint API* dedicati, che saranno descritti nelle sottosezioni successive.

#### Recupero dei Task

Per ogni piano, una volta ottenuti i dettagli, è necessario recuperare tutti i *task* associati ad esso.

Ogni *task* contiene informazioni cruciali come l'identificativo, il titolo, la scadenza, le categorie applicate, e la priorità. La gestione della descrizione dei *task* è anch'essa inclusa, se disponibile.

L'endpoint per ottenere i *task* di un piano è il seguente:

```
GET /planner/plans/{planId}/tasks
```

La risposta dell'*API* fornisce una lista di *task* associati al piano. I dettagli dei *task* comprendono i campi principali rappresentati in Figura 5.13.

Questi campi sono stati discussi e dopo varie presentazioni è stato concordato di sincronizzare solo questi dati.

```
private String id; 3 usages
private String etag; 3 usages
private String bucketId; 3 usages
private String title; 4 usages
private String dueDateTime; 7 usages
private boolean hasDescription; 3 usages
private String description; 4 usages
private List<Category> appliedCategories; 5 usages
private int priority; //0,1 = urgent; 28 usages
                    //2,3,4 = important
                    //5,6,7 = medium
                    //8,9 = low
```

Figura 5.13: Classe *Task*

Questi dati vengono estratti e suddivisi in due categorie principali: *BugTask* e *RegularTask*.



### 5.8.6 Recupero delle Categorie/Tag di un Piano

Il recupero delle categorie (o *tag*) associate a un piano in *Microsoft Planner* è un passaggio cruciale per ottenere le informazioni relative alle etichette applicate ai *task* all'interno del piano.

Queste categorie permettono una migliore gestione e classificazione dei *task*, consentendo di distinguere tra diverse tipologie di attività.

#### Endpoint per le Categorie di un Piano

L'*endpoint* per ottenere le categorie di un piano è il seguente:

```
GET https://graph.microsoft.com/v1.0/planner/plans/{planId}/details
```

#### Parametri della Richiesta

La richiesta per recuperare le categorie richiede i seguenti parametri:

- **planId**: L'identificativo univoco del piano di cui si desidera recuperare le categorie.
- **Authorization**: Il *token* di autenticazione, necessario per autorizzare l'accesso ai dati del piano.

#### Analisi della Risposta

La risposta dell'*API* contiene un oggetto *JSON* che fornisce i dettagli del piano, inclusi i *tag* applicati.

Il campo di interesse principale per il recupero delle categorie è `categoryDescriptions`, che rappresenta le descrizioni delle categorie applicabili al piano.

Ogni categoria è rappresentata con un indice numerico e una descrizione associata.

Se la categoria è applicata, la descrizione è presente; altrimenti, il valore della descrizione sarà `null`.

La risposta *JSON* avrà una struttura simile alla seguente:

```
"categoryDescriptions": {  
  "category1": "Documenti",  
  "category2": "Bug",  
  "category3": null,  
  "category4": "Spanesi"  
}
```

Dove:

- Ogni chiave (ad esempio, `category1`) rappresenta un identificatore della categoria.
- Il valore associato rappresenta la **descrizione** della categoria, se disponibile.

Se una categoria non ha una descrizione applicata, il valore sarà `null`.

#### Gestione delle Categorie

Nel codice, le categorie vengono estratte dal campo `categoryDescriptions` e rappresentate come oggetti di tipo `Category`.

Ogni categoria viene identificata da una chiave (ad esempio, `category1`) e associata a un valore che indica se la categoria è stata applicata al piano.

```
Categoria: "Documenti", Applicata: true
Categoria: "Bug", Applicata: true
Categoria: "Spanesi", Applicata: false
```

### 5.8.7 Recupero dei Bucket di un Piano

Il recupero dei *bucket* associati a un piano in *Microsoft Planner* è essenziale per ottenere informazioni sui contenitori che raggruppano i *task* all'interno di un piano.

Ogni bucket rappresenta una sezione o categoria di attività, utile per organizzare e gestire i *task* in modo efficiente.

#### Endpoint per i Bucket di un Piano

L'*endpoint* per ottenere i *bucket* di un piano è il seguente:

```
GET https://graph.microsoft.com/v1.0/planner/plans/{planId}/buckets
```

#### Parametri della Richiesta

La richiesta per recuperare i *bucket* richiede i seguenti parametri:

- **planId**: L'identificativo univoco del piano di cui si desidera recuperare i *bucket*.
- **Authorization**: Il *token* di autenticazione, necessario per autorizzare l'accesso ai dati del piano.

#### Analisi della Risposta

La risposta dell'*API* restituisce una lista di oggetti *JSON*, ciascuno dei quali rappresenta un *bucket* del piano.

Ogni *bucket* è identificato da un **id** e un **name**, che ne definiscono rispettivamente l'identificativo e il nome.

La risposta *JSON* avrà una struttura simile alla seguente:

```
"value": [
  { "id": "54326", "name": "To Do" },
  { "id": "76738", "name": "In Progress" },
  { "id": "07965", "name": "Done" }
]
```

Ogni oggetto *bucket* contiene:

- **id**: L'identificativo univoco del *bucket*.
- **name**: Il nome del *bucket* (ad esempio, "To Do", "In Progress", "Done").

#### Gestione dei Bucket

Nel codice, i *bucket* vengono estratti dalla risposta *JSON* e rappresentati come oggetti di tipo `Bucket`.

I *bucket* sono fondamentali per gestire e organizzare le attività all'interno di un piano. Ogni *task* è associata a un *bucket*, e questa relazione è cruciale per il flusso di lavoro del progetto. In particolare, i *bucket* vengono utilizzati per spostare le *task* tra

le varie fasi del piano. Ad esempio, quando una *user story* o *issue* cambia stato, la *task* corrispondente viene spostata nel *bucket* appropriato.

Questo meccanismo di spostamento dei *task* tra i *bucket* sarà esplorato più dettagliatamente nelle prossime sezioni, ma è importante sottolineare che i *bucket* aiutano a mantenere organizzato il progresso del progetto.

Inoltre, i *bucket* sono utili anche per monitorare lo stato attuale di una *task*. Ad esempio, se una *task* è spostata nel *bucket* "In Analisi [Sviluppatori]", significa che la *task* ha raggiunto una fase specifica del ciclo di vita. Questa informazione è particolarmente rilevante quando si tratta di gestire le *RegularTask*, poiché consente di creare una *user story* corrispondente, come vedremo più avanti.

## 5.9 Confronto dei Dati

Il confronto dei dati è un'operazione fondamentale per monitorare le modifiche apportate agli elementi di un progetto. Questo processo avviene confrontando i dati correnti con quelli salvati in precedenza per individuare eventuali variazioni. Ogni tipo di dato (*task*, *user story*, *issue*) viene gestito separatamente, garantendo una logica specifica e scalabile.

Il confronto viene eseguito per le seguenti tipologie di dati:

- **Regular Task:** confronto dei *RegularTask* salvati con quelli recuperati.
- **Bug Task:** confronto analogo per i *BugTask*.
- **User Story:** confronto tra le *user story* salvate in precedenza e quelle attuali.
- **Issue:** confronto tra le *issue* precedenti e quelle nuove.

In caso di modifiche rilevate, viene avviata un'analisi per aggiornare i dati in modo da riflettere i cambiamenti identificati.

Ogni variazione significativa viene gestita in maniera separata per garantire coerenza nel sistema.

### 5.9.1 Confronto delle *Regular Tasks*

Il confronto delle *RegularTask* avviene verificando eventuali modifiche tra i *task* salvati e quelli recuperati.

Per effettuare il confronto in modo efficiente, viene utilizzato il campo `etag`, un identificatore univoco che rappresenta lo stato corrente di un *task*.

Se l'`etag` di un *task* è diverso rispetto alla versione precedente, significa che sono state apportate modifiche.

L'`etag` consente di verificare rapidamente se un elemento è stato aggiornato senza dover confrontare direttamente tutti i dettagli. Questo approccio migliora l'efficienza del confronto e riduce il rischio di errori.

### Azioni sui Task Modificati

Quando viene rilevata una modifica significativa in una *RegularTask*, il sistema esegue le seguenti operazioni:

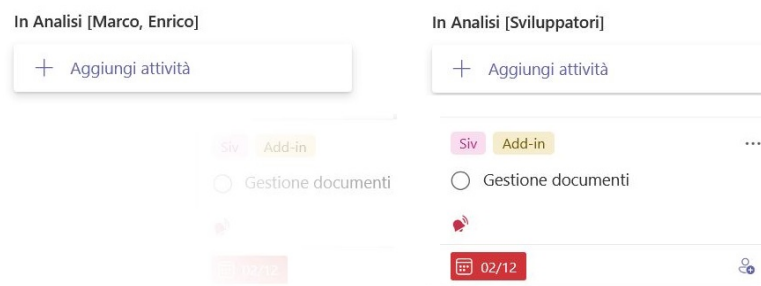
1. Verifica se il *task* è stato spostato in un bucket rilevante, come il bucket "In Analisi [Sviluppatori]".
2. Identifica se il *task* è associato a una *user story* esistente.
3. Se sono presenti modifiche importanti o uno spostamento rilevante, viene aggiornato il contenuto della *user story* associata o, se necessario, creata una nuova.

### Creazione della *User Story* per una *Regular Task*

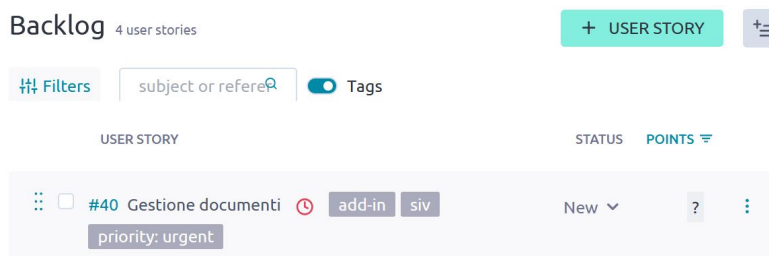
Quando una *RegularTask* viene spostata nel bucket "In Analisi [Sviluppatori]" (come mostrato in Figura 5.14) e non esiste ancora una *user story* associata, il sistema procede con la creazione di una nuova *user story* nel backlog del progetto *Taiga* associato al piano *Planner* (come mostrato in Figura 5.15).

Il sistema effettua i seguenti passaggi per creare una nuova *user story*:

1. Recupera l'elenco delle *user story* esistenti per il progetto.
2. Verifica che non esista una *user story* con lo stesso titolo del *task*.
3. Se non trovata, utilizza i dettagli del *task* per creare una nuova *user story*.



**Figura 5.14:** Esempio di una *Regular Task* spostata nel bucket "In Analisi [Sviluppatori]".



**Figura 5.15:** Creazione della *user story* corrispondente alla *RegularTask* nel progetto *Taiga*.

### Modifiche Rilevanti nelle *Regular Tasks*

Oltre alla creazione di una nuova *user story* per una *regular task*, il sistema tiene traccia e riporta eventuali modifiche significative apportate ai *task* da *Planner* nelle corrispondenti *user story* in *Taiga*. Questo garantisce che i dati siano sempre sincronizzati e che le *user story* riflettano accuratamente lo stato corrente dei *task*.

**Tipologie di Modifiche Rilevanti** Le modifiche rilevanti considerate dal sistema includono:

- **Cambiamento del titolo:** Se il titolo del *task* viene modificato, la variazione viene registrata e riportata nella *user story*, aggiornando il campo **subject**.
- **Cambiamento della data di scadenza:** Ogni aggiornamento alla data di scadenza del *task* viene riflesso nella *user story* corrispondente attraverso il campo **due\_date**.
- **Modifica della descrizione:** Una variazione nella descrizione del *task* viene sincronizzata per mantenere allineate le informazioni nel campo **description**.
- **Aggiornamento delle categorie applicate:** Se cambiano le categorie assegnate al *task*, queste vengono trasformate in *tag* e aggiornate nel campo **tags** della *user story*.
- **Cambiamento della priorità:** Ogni modifica alla priorità del *task* viene propagata alla *user story* come un *tag* aggiuntivo, poiché il campo **priority** non è direttamente supportato.

### Endpoint per l'Aggiornamento di una User Story

L'aggiornamento di una *user story* in *Taiga* avviene utilizzando il seguente *endpoint REST*:

```
PATCH https://{taiga_base_url}/api/v1/userstories/{userStoryId}
```

### Parametri della Richiesta

La richiesta richiede l'invio di specifici parametri nel *body* e negli *header*. Di seguito vengono descritti in dettaglio:

### Parametri del Body

- **subject:** Il nuovo titolo della *user story*, derivato dal titolo del *task*.
- **due\_date:** La data di scadenza aggiornata del *task* (se presente).
- **description:** La descrizione aggiornata del *task* (se disponibile).
- **tags:** Un *array* contenente i *tag* aggiornati, inclusi i nomi delle categorie applicate al *task* e la sua priorità.
- **version:** La versione attuale della *user story*, utilizzata per evitare conflitti durante l'aggiornamento. Ogni volta che una *user story* viene modificata in *Taiga*, il sistema incrementa il numero di versione associato a quella *user story*. Questo parametro garantisce che le modifiche vengano applicate solo se la versione

inviata nella richiesta corrisponde a quella attualmente salvata nel sistema. In caso di discrepanza, ad esempio quando un'altra modifica è stata apportata nel frattempo, l'aggiornamento viene rifiutato per evitare sovrascritture non intenzionali. Questo meccanismo è fondamentale per mantenere l'integrità dei dati in scenari con aggiornamenti concorrenti.

### Headers della Richiesta

- **Authorization:** *Bearer token* necessario per autenticare la richiesta.
- **Content-Type:** Specifica il formato della richiesta, impostato ad *application/json*.

### Analisi della Risposta

L'aggiornamento di una *user story* restituisce una risposta *HTTP* per indicare l'esito dell'operazione. In caso di successo, viene restituito uno stato *200 OK*, senza alcun corpo *JSON*.

**Gestione delle Modifiche** Quando una modifica rilevante viene identificata, il sistema:

1. Registra il cambiamento tramite un *log* dettagliato per garantire tracciabilità.
2. Aggiorna la *user story* in *Taiga*, mantenendo sincronizzati i dati con *Planner*.
3. Specifica le modifiche apportate nel *log* di sistema per una revisione successiva, se necessario.

**Sincronizzazione delle Categorie** Un caso particolare riguarda le categorie applicate ai *task*. Il sistema considera solo categorie valide, ignorando quelle non pertinenti o vuote. Le categorie vecchie e nuove vengono confrontate e, se diverse, la *user story* associata viene aggiornata per riflettere il cambiamento.

## 5.9.2 Confronto delle *Bug Tasks*

Il confronto delle *BugTasks* segue lo stesso approccio descritto per le *RegularTasks*, utilizzando il campo *etag* per individuare modifiche rispetto alla versione precedente. In presenza di variazioni, il sistema esegue le azioni necessarie per aggiornare i dati.

**Azioni sui Task Modificati** Le azioni relative alle *BugTasks* modificate includono:

1. Verifica delle modifiche in campi rilevanti (es. titolo, descrizione, data di scadenza, categorie, severità).
2. Controllo dell'esistenza di un'*issue* associata.
3. Aggiornamento o creazione dell'*issue* in *Taiga* con i dettagli aggiornati.
4. Registrazione delle modifiche nel *file* di *log*.

### Creazione dell'*Issue* per una *Bug Task*

Se una *BugTask* non ha un'*issue* associata, il sistema crea una nuova *issue* utilizzando le informazioni del *task* (titolo, descrizione, data di scadenza, categorie, severità).

La creazione avviene tramite una richiesta *POST* al server *Taiga*, assicurando che non vi siano duplicati.

### Modifiche Rilevanti nelle *Bug Tasks*

Le modifiche significative delle *bug tasks* seguono lo schema delle *RegularTasks*, con l'aggiunta della gestione del campo *priority*, che viene mappato direttamente al livello di *severity* dell'*issue* in *Taiga*.

### Endpoint per l'Aggiornamento di una *Issue*

L'aggiornamento delle *issues* in *Taiga* avviene tramite l'*endpoint REST*:

```
PATCH https://{taiga_base_url}/api/v1/issues/{issueId}
```

**Parametri della Richiesta** I parametri richiesti sono analoghi a quelli descritti per le *RegularTask*, con l'inclusione del campo *severity*, specifico per le *BugTasks*.

Di seguito i parametri principali:

- *subject*, *due\_date*, *description*, *tags*: derivati dalle informazioni del *task*.
- *severity*: *ID* della severità. Il sistema determina l'*ID* corrispondente alla severità della *BugTask* confrontandola con la lista delle severità del progetto *Taiga*, precedentemente recuperata. In base al valore della priorità della *BugTask*, viene assegnato l'*ID* corretto per la severità, indipendentemente dal suo livello (ad esempio, *low*, *medium*, *important*, *urgent*).
- *version*: utilizzato per evitare conflitti di aggiornamento.

**Headers della Richiesta** Analoghi a quelli descritti per le *RegularTask*, includono:

- *Authorization*: *Bearer token* per autenticazione.
- *Content-Type*: impostato a *application/json*.

### Analisi della Risposta

L'aggiornamento dell'*issue* restituisce una risposta HTTP con codice 200 OK in caso di successo. Non è previsto un corpo JSON nella risposta. Se l'aggiornamento va a buon fine, la modifica viene applicata senza ulteriori dettagli nel corpo della risposta. Qualora si verificassero problemi, il server restituirà un codice di errore (come 400, 401, 404) per indicare il tipo di errore, ma in caso di successo non è presente un JSON di conferma dettagliata.

**Gestione delle Modifiche** Il sistema registra ogni variazione rilevante nei *log* e sincronizza le *issues* con *Planner*, mantenendo la coerenza dei dati. Questo processo è analogo a quanto descritto per le *RegularTask*.

### 5.9.3 Confronto delle User Stories

Il confronto delle *user story* avviene analizzando eventuali modifiche tra le versioni precedentemente salvate e quelle recuperate dal sistema.

Questo processo si basa principalmente sul campo **version**, che rappresenta lo stato corrente di una *user story*.

**Gestione delle Modifiche** Quando viene rilevata una modifica significativa, come un cambiamento di stato (*status*), il sistema intraprende una serie di azioni per garantire l'allineamento tra le *user story* e i *task* associati nel *Planner*.

Il confronto tiene conto esclusivamente delle modifiche allo stato della *user story*.

#### Cambiamento di Stato in "Analizzata e Stimata"

Quando una *user story* passa allo stato "Analizzata e Stimata", il sistema esegue automaticamente le seguenti operazioni:

1. **Calcolo dei punti e tempo stimato:** Viene utilizzata la Tabella di conversione 2.1 che associa i punti di lavoro al tempo necessario.
2. **Aggiornamento del titolo della *RegularTask*:** Il sistema aggiorna il titolo della *task* associata aggiungendo un prefisso che indica il tempo stimato in formato leggibile. Ad esempio, un titolo iniziale come "Implementazione Login" diventa "[1w 2d 7h] Implementazione Login".
3. **Spostamento della task nel bucket corrispondente:** La *task* viene trasferita automaticamente nel bucket "Analizzata e Stimata" del *Planner*.
4. **Sincronizzazione del titolo della *User Story*:** Il nuovo titolo della *RegularTask* viene propagato alla *user story* associata, mantenendo i due elementi sincronizzati.

**Aggiornamento delle *RegularTask* Associate** Quando una *user story* passa allo stato "Analizzata e Stimata", il sistema effettua due aggiornamenti principali sulla *RegularTask* associata:

- **Cambio del titolo:** Il titolo della *task* viene aggiornato per includere il prefisso con il tempo stimato (ad esempio, "[1w 2d 7h] Implementazione Login"). Questo aggiornamento garantisce una rappresentazione chiara e coerente della stima di lavoro necessaria.
- **Cambio del bucket:** La *RegularTask* viene spostata automaticamente nel *bucket* "Analizzata e Stimata" del *Planner*, assicurando che il suo stato rifletta il nuovo stato della *user story*.

Questi aggiornamenti avvengono tramite specifici *endpoint REST*, utilizzando il metodo **PATCH** per modificare i campi **title** e **bucketId** della *task*. La chiamata include il parametro **etag** per garantire che l'aggiornamento sia effettuato sull'ultima versione del *task*, prevenendo conflitti con eventuali modifiche concorrenti.



**Endpoint REST Utilizzati** Gli *endpoint REST* usati per aggiornare le *RegularTask* sono:

- **Per il cambio del titolo:** PATCH /planner/tasks/{taskId}
- **Per il cambio del bucket:** PATCH /planner/tasks/{taskId}

In entrambi i casi, le richieste includono:

- **Autenticazione:** Un *token Bearer* per autorizzare l'operazione.
- **Intestazione If-Match:** Necessaria per specificare l'ultima versione nota del *task* (*etag*).
- **Corpo JSON:** I dettagli del cambiamento, ad esempio "title": "[1w 2d 7h] Implementazione Login" o "bucketId": "Analizzata e Stimata".

### Spostamenti nei Bucket per Altri Stati

Per ogni modifica di stato diversa da "Analizzata e Stimata", il sistema verifica il bucket appropriato in base alle configurazioni predefinite, come spiegato nel Requisito RFO15 4.1.1.

La *regular task* associata viene spostata automaticamente nel *bucket* corretto, mantenendo l'allineamento con lo stato della *user story*.

**Esempio di Calcolo del Tempo** Utilizzando la Tabella di conversione 2.1, una *user story* con un punteggio totale di 63 punti viene calcolata come segue:

- 63 punti equivalgono a 1 settimana, 2 giorni e 7 ore.
- Il prefisso risultante sarà [1w 2d 7h].

Questo prefisso viene aggiunto al titolo della *task* associata e sincronizzato automaticamente con la *user story*.

### Monitoraggio e Logging

Ogni modifica rilevante viene registrata in un *log* dettagliato. Questo include:

- Il titolo originale e quello aggiornato della *user story*.
- Lo stato iniziale e quello modificato.
- Il *bucket* di origine e quello di destinazione per la *regular task*.

## 5.10 Sistema di Log e Gestione dei File

Il sistema utilizza *SharePoint* per archiviare e gestire i *file* di *log* prodotti dall'applicazione, garantendo uno storico completo delle operazioni effettuate.

### 5.10.1 Creazione del Sito e della Cartella per i Log

Il sistema di *log* è stato configurato su *SharePoint* seguendo i passaggi descritti di seguito.

### Creazione del Sito

Per configurare l'ambiente di archiviazione su *SharePoint*, è stato necessario accedere all'*account* di sviluppo del *tenant* `@wtcsviluppo.onmicrosoft.com`, in quanto le operazioni richiedono l'accesso autenticato al contesto di sviluppo.

Una volta effettuato il *login*, è stato creato un nuovo sito su *SharePoint*. Durante la creazione, è stato selezionato il tipo *Sito del Team*, come mostrato in Figura 5.16. La scelta è stata motivata dalla necessità di garantire una visualizzazione completa dei *file* e delle risorse esclusivamente ai membri del *team*, limitandone l'accesso a livello aziendale per mantenere riservatezza e organizzazione.

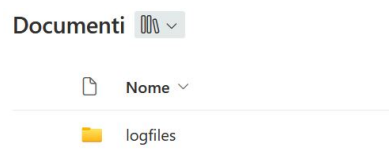


**Figura 5.16:** Scelta del tipo di sito: Sito del Team.

Successivamente, è stato selezionato il modello *Team standard* per garantire una struttura semplice e funzionale. Il sito è stato denominato *TestMyApp* e descritto come un repository per *file* di *log* e configurazioni generati durante la *PoC*. La sezione principale di interesse è quella denominata *Documenti*, dove vengono archiviati i *file*.

### Creazione della Cartella *logfiles*

All'interno della sezione *Documenti*, è stata creata una cartella denominata *logfiles*, destinata a ospitare i file di *log* generati dall'applicazione, come illustrato in Figura 5.17.



**Figura 5.17:** Creazione della cartella *logfiles*.

## 5.10.2 Generazione e Struttura dei File di Log

### Creazione Automatica dei File di Log

Il sistema genera automaticamente un *file di log* per ciascuna coppia *piano-progetto* sincronizzata tra *Microsoft Planner* e *Taiga*. La generazione avviene alla prima modifica rilevante che richiede un intervento del sistema, ad esempio la creazione di una *User Story* su *Taiga* corrispondente a una *Task* spostata nel *bucket* "In Analisi [Sviluppatori]" su *Planner*.

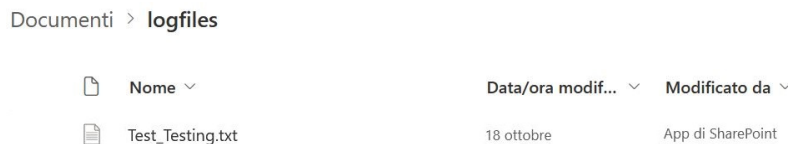
Il *file* viene denominato seguendo la convenzione:

NomePiano\_NomeProgetto.txt

Ad esempio, se il piano di *Planner* è *ImplementazioneWow* e il progetto *Taiga* è *Wow*, il *file di log* generato sarà:

ImplementazioneWow\_Wow.txt

Questo *file* può essere consultato nella cartella *logfiles* del sito *TestMyApp*, come illustrato in Figura 5.18.



**Figura 5.18:** Esempio di file di log archiviato in SharePoint.

### Struttura dei File di Log

Ogni *file di log* contiene una registrazione cronologica delle attività svolte, utilizzando il seguente formato standard:

[DD/MM/YY HH:mm:ss]: Azione descrittiva.

L'azione descrittiva comprende:

- L'oggetto modificato (*User Story*, *Task*, *Issue*) e il relativo titolo.
- Il piano o progetto coinvolto, specificato tramite nome.
- L'utente che ha effettuato la modifica.
- La modifica effettuata, indicando i valori precedenti e quelli aggiornati.
- L'azione intrapresa dal sistema in risposta alla modifica.

Un esempio di riga di *log* è il seguente:

```
[15/10/24 12:48:59]: User Story '[4d 7h] Ciao000' nel progetto
'Test my app': 'Giacomo Gualato' ha cambiato lo stato da 'New' a
'Ready for test'. Ora sposterò la task corrispondente nel bucket 'In Test'.
```

### 5.10.3 Estrazione delle Informazioni dai Sistemi

#### Identificazione dell'Oggetto e del Progetto

Per determinare il nome del progetto o piano associato a una modifica, il sistema utilizza i dati ottenuti in fase di sincronizzazione. Ad esempio, per una *User Story* con un determinato *Project ID*, il nome del progetto viene ricavato confrontando l'*ID* con l'elenco dei progetti recuperati nella Sezione 5.6.3.

#### Estrazione del Nome dell'Utente su Taiga

Per identificare l'ultimo utente che ha modificato lo stato di una *User Story*, viene effettuata una chiamata all'endpoint:

```
GET /history/userstory/{userStoryId}
```

Questa richiesta restituisce un *array JSON* con la cronologia delle modifiche associate alla *User Story*. Tra le informazioni presenti, vengono analizzate le modifiche di stato ("status"), estrapolando il nome dell'utente che ha effettuato la modifica effettuata.

Un esempio di risposta *JSON* è il seguente:

```
[
  { "diff": { "status": { "from": "New", "to": "Ready for test"
  } },
  "user": { "name": "Mario Rossi" } },
  { "diff": { "subject": { "from": "Old Title", "to": "New Title"
  } },
  "user": { "name": "Anna Bianchi" } }
]
```

In questo esempio, il cambiamento di stato è stato effettuato da "Mario Rossi". Mentre, il cambiamento del titolo è stato effettuato da "Anna Bianchi". Il sistema analizza queste informazioni per aggiornare correttamente il *file* di *log*.

### Estrazione del Nome dell'Utente su Planner

L'estrazione del nome dell'utente che ha effettuato modifiche su *Planner* non è stata possibile implementarla.

Ciò è dovuto al fatto che, a differenza delle *User Story* su *Taiga*, che forniscono uno storico dettagliato attraverso le *API*, le *API* di *Planner* (parte di *Microsoft Graph*) attualmente non supportano l'accesso alla cronologia delle modifiche delle *Task*.

Dopo numerose ricerche, è emerso che la mancanza di un'*API* per la cronologia delle *Task* è una limitazione nota e discussa da anni, come evidenziato nella Figura 5.19.

Nonostante l'interesse della community, questa funzionalità non è ancora stata rilasciata.



**Figura 5.19:** Discussione sulla mancanza di una cronologia delle *Task* nelle *API* di *Planner*.

Un altro aspetto da considerare è che, anche con una licenza di *Project Plan 3* o superiore, che consente di accedere a funzionalità avanzate di *Planner*, la cronologia delle attività è disponibile solo come parte dell'interfaccia grafica.

Non esiste alcuna *API* che permetta di accedere a questi dati, rendendo quindi impossibile la loro integrazione automatica nella *PoC*.

Nello specifico, la cronologia grafica permette di visualizzare i dettagli delle modifiche cliccando sull'icona dedicata, situata nell'angolo superiore del pannello dei dettagli della *Task*, come mostrato in Figura 5.20.

Tuttavia, questa funzionalità è limitata alla consultazione manuale e non fornisce un supporto *API* per l'automazione.

The image shows a screenshot of the Microsoft Planner interface. On the left, a task titled "Review local marketing ..." is displayed. The task is assigned to Alex Wilber and has a "Requires legal signoff" label. The task details include a start date of 07/17/2024, a finish date of 07/17/2024, a duration of 1 day, 0% completion, and a priority of "Important". The task is currently in the "Create Localization Strategy" bucket. On the right, a "Changes" panel shows a list of recent updates:

- Diego Siciliani (Several minutes ago) edited the earlier task "Review global marketing strategy, Depends on", impacting the start date from 06/27/2024 to 07/17/2024.
- Alex Wilber (Several minutes ago) edited the bucket to "Create Localization Strategy", which includes the task "Review Business Strategy".
- Alex Wilber (Several minutes ago) added a label "Requires legal signoff".
- Alex Wilber (Several minutes ago) edited the name to "Review local marketing strategy", with a placeholder for Alex's strategy work.
- Diego Siciliani (Several minutes ago) added an attachment "Review local marketing strategy...".

**Figura 5.20:** Interfaccia grafica per la cronologia delle *Task* su Planner.

Pertanto, anche con licenze *premium*, questa limitazione rende impossibile ottenere in modo automatico il nome dell'utente che ha effettuato le modifiche su *Planner*.

Posso quindi concludere che il requisito 4.1.1 risulta non realizzabile.

## 5.11 Gestione tramite interfaccia web

Per agevolare gli utenti autorizzati, come amministratori o personale tecnico, è stata implementata un'interfaccia *web* dedicata alla gestione delle associazioni tra *Microsoft Planner* e *Taiga*.

Questa interfaccia permette di effettuare operazioni di configurazione e monitoraggio in maniera intuitiva e autoesplicativa, senza necessità di consultare continuamente il manuale amministratore. Tale approccio è stato condiviso e approvato dal *tutor* aziendale, Enrico Merigliano.

L'interfaccia è stata sviluppata utilizzando il *framework Thymeleaf* per il *rendering* dinamico delle pagine *web*. Tale scelta è stata motivata dalla semplicità di integrazione con il linguaggio *Java* e con il *framework Spring Boot*, utilizzato per il *backend*.

Per quanto riguarda l'aspetto grafico, è stato utilizzato il *framework Bootstrap*, che offre componenti predefiniti e responsivi per la progettazione dell'interfaccia utente.

**Accesso e struttura delle pagine** L'interfaccia *web* è accessibile tramite l'indirizzo locale `localhost:8080`.

Se il sistema fosse implementato su un *server* aziendale con endpoint pubblico, gli utenti potrebbero accedervi tramite un *URL* personalizzato con l'indirizzo *IP* del *server* e la porta configurata.

La struttura delle pagine è stata progettata per facilitare la navigazione e l'utilizzo.

Una volta raggiunto l'indirizzo principale, il sistema reindirizza automaticamente alla pagina iniziale (`http://localhost:8080/select`), dove è possibile selezionare un piano di *Microsoft Planner* e un progetto di *Taiga* da associare.

### 5.11.1 Pagina di selezione piano e progetto

La pagina iniziale consente agli utenti di configurare una sincronizzazione tra un piano di *Microsoft Planner* e un progetto di *Taiga*.

Tale funzionalità rappresenta il punto di ingresso principale per gestire le associazioni necessarie a una sincronizzazione automatica e continua tra i due strumenti.

Un esempio dell'aspetto della pagina è mostrato in Figura 5.21.

#### Funzionalità principali

La pagina offre diversi strumenti interattivi:

- **Selezione del piano Microsoft Planner:** tramite un *menù* a tendina, l'utente può scegliere un piano basato su quelli disponibili e recuperati dal *file* di configurazione tramite i *group-id*. È inoltre presente un pulsante che permette di personalizzare i *group-id*, aprendo la pagina `http://localhost:8080/customize-groups`.
- **Selezione del progetto Taiga:** anche i progetti vengono presentati in un *menù* a tendina. I dati sono recuperati tramite un utente tecnico *bot* che deve essere associato a tutti i progetti desiderati su *Taiga* per essere qui visibili.
- **Modalità Aggiornamento Completa:** un interruttore attiva questa modalità, che consente di ampliare la ricerca dei piani *Microsoft Planner* includendo tutti quelli presenti nel *tenant*, indipendentemente dai *group-id* configurati. Attivando questa opzione, il caricamento dei piani può richiedere fino a 30 secondi. Disattivandola, l'applicazione utilizza solo i *group-id* definiti nel *file* di configurazione.

## Seleziona Piano e Progetto

### Seleziona Piano Microsoft Planner



I piani disponibili sono quelli recuperati dal file di configurazione tramite group-id.

### Seleziona Progetto Taiga

I progetti mostrati sono quelli definiti in Taiga e recuperati tramite un utente tecnico bot. Questo utente deve essere inserito in tutti i progetti disponibili in Taiga, per poter essere visibili e selezionabili.

### Modalità Aggiornamento Completa

**Modalità Aggiornamento Completa:** Quando attivata, l'applicazione recupererà **tutti i piani Microsoft Planner** associati al tenant, indipendentemente da quelli definiti nel file di configurazione. Questo potrebbe includere un numero elevato di dati e il processo potrebbe richiedere fino a **30 secondi**. Dopo aver attivato questa modalità, **aggiorna la pagina** per vedere i risultati completi. Se disattivata, l'applicazione utilizzerà solo i piani definiti nel file di configurazione tramite i group-id.

Dopo aver salvato l'associazione, la sincronizzazione automatica tra il piano di Microsoft Planner e il progetto Taiga selezionato inizierà entro **10 secondi**. Le modifiche apportate ai campi delle task in Planner verranno sincronizzate automaticamente alla user story o issue corrispondente. Le modifiche apportate ai campi delle task in Taiga (tranne i punti) non verranno considerate, si prega di non farlo.

Non è necessario spostare manualmente le task nelle bucket di Planner: l'applicazione rileva automaticamente i cambiamenti di stato in Taiga e aggiorna la posizione delle task in Planner in modo coerente.

Prima di iniziare a effettuare modifiche bisogna configurare i rapporti tra gli stati delle user story e delle issue. Si prega di andare in 'Gestisci Associazione', poi in 'Configura UserStory' e poi in 'Configura Issue'.

**Figura 5.21:** Pagina di selezione Piano e Progetto.

## Generazione di link dinamici

- Dopo la selezione di un piano, il sistema genera un *link* diretto a *Microsoft Planner* nella seguente forma:  
`https://planner.cloud.microsoft/webui/plan/{planId}`, dove `planId` è l'identificativo univoco del piano selezionato.
- Analogamente, per il progetto *Taiga*, viene fornito un *link* nella forma:  
`https://tree.taiga.io/project/{slug}`, dove `slug` è una stringa che rappresenta l'identificativo leggibile del progetto. Esso combina il nome utente associato al progetto e il nome del progetto stesso.  
Ad esempio: `https://tree.taiga.io/project/guala_test-my-app/`.

Nella parte inferiore della pagina, vi è una tabella che riepiloga tutte le associazioni esistenti, con *link* diretti a ogni piano e progetto configurato (Figura 5.22).

## Associazioni Esistenti

Qui di seguito puoi vedere le associazioni esistenti tra i progetti di Taiga e i piani di Planner.

Piano	Progetto
<a href="#">Test my app</a>	<a href="#">Test my app</a>
<a href="#">prova</a>	<a href="#">Testing</a>

**Figura 5.22:** Riepilogo delle associazioni configurate tra Planner e Taiga.



### Funzionamento della sincronizzazione

Dopo aver salvato l'associazione, la sincronizzazione automatica tra il piano di *Microsoft Planner* e il progetto *Taiga* selezionato viene attivata entro 10 secondi.

La sincronizzazione, come già detto, prevede:

- Modifiche ai campi delle *task* in *Planner*, sincronizzate automaticamente con le corrispondenti *user story* o *issue* in *Taiga*.
- Stato delle *user story* e delle *issue* in *Taiga* che, una volta aggiornati, spostano automaticamente le *task* nei *bucket* appropriati di *Planner*.

### Frequenza di aggiornamento delle associazioni

La sincronizzazione tra *Planner* e *Taiga* avviene automaticamente ogni 10 secondi. Tuttavia, all'aumentare del numero di *task*, *issue*, *user story* o associazioni, il tempo di aggiornamento potrebbe estendersi.

### 5.11.2 Pagina Personalizza Gruppi

La pagina di personalizzazione dei *GroupID* consente agli utenti di aggiungere, modificare o eliminare i *GroupID* associati ai piani di *Microsoft Planner*. Questa funzionalità è fondamentale per ampliare o restringere i piani disponibili per la sincronizzazione con *Taiga*.

Un esempio dell'aspetto della pagina è mostrato in Figura 5.23.



Figura 5.23: Pagina di Personalizzazione dei *GroupID*.

### Funzionalità principali

Le funzionalità offerte da questa pagina includono:

- **Aggiunta di nuovi GroupID:** tramite un modulo, gli utenti possono inserire un nuovo *GroupID*. Dopo aver salvato, il nuovo gruppo sarà disponibile per la selezione dei piani.
- **Visualizzazione dei GroupID esistenti:** una tabella riepiloga i *GroupID* attualmente configurati, permettendo di monitorare quali piani sono già associati.

- **Eliminazione di GroupID:** ogni riga della tabella include un pulsante per eliminare il corrispondente *GroupID*. L'eliminazione rimuove il piano o i piani associati dal *GroupID* eliminato.

### Cosa sono i GroupID e come recuperarli

I *GroupID* rappresentano l'identificativo unico di un piano di *Microsoft Planner*. Quando si accede a un piano tramite l'interfaccia *web* di *Planner*, il *GroupID* può essere recuperato facilmente dall'*URL* della pagina.

Ad esempio, accedendo al seguente URL:

`https://tasks.office.com/tenant-id/en-US/Home/Planner#/plantaskboard?groupId=2jQOGmK7D0G0uMV1JfjodpcAAoow`

Il *GroupID* è la stringa presente dopo `groupId=`, in questo caso:

`2jQOGmK7D0G0uMV1JfjodpcAAoow`

Questo identificativo può essere copiato e incollato nel modulo della pagina *Personalizza Gruppi* per associare il piano corrispondente al sistema di sincronizzazione.

### Esempio pratico

Nella Figura 5.23 viene mostrato l'aspetto della pagina in uso. Gli utenti possono aggiungere un nuovo *GroupID*, ad esempio `2jQOGmK7D0G0uMV1JfjodpcAAoow`, tramite il modulo dedicato. I *GroupID* già configurati sono visibili nella tabella sottostante. Un pulsante accanto a ciascun *GroupID* consente di rimuoverlo con un semplice *click*.

#### Gruppi Esistenti

GroupID	Azioni
0c7ecafe-7fdb-4f3d-aef1-4c459a58cdb4	Elimina
6f6e2b25-0e4a-42c2-9ac7-f5727c551089	Elimina
beb64b0a-e049-4cad-ba38-32ee85d28e5e	Elimina
95ccd6fb-d910-4e20-a668-9d6bc1e6d433	Elimina

**Elimina:** Rimuovi il GroupID associato al piano Microsoft Planner. Eliminando il GroupID, il piano non verrà più associato a nessun progetto o applicazione.

**Ricarica della pagina:** Dopo aver apportato modifiche ai GroupID, l'applicazione impiegherà circa **10 secondi** per sincronizzarsi. Dopo questo periodo, ricarica la pagina per vedere i piani aggiornati nella selezione.

**Figura 5.24:** Tabella dei *GroupID* esistenti con opzione di eliminazione.

### 5.11.3 Pagina Gestisci Associazioni

La pagina *Gestisci Associazioni* (visibile in Figura 5.25) consente agli utenti di visualizzare, configurare ed eliminare le associazioni tra i piani di *Microsoft Planner* e i progetti di *Taiga*.

**Gestisci Associazioni**

[Torna alla Selezione](#)

Piano	Progetto	Azioni
Test my app	Test my app	<a href="#">Elimina</a> <a href="#">Configura UserStory</a> <a href="#">Configura Issue</a>
prova	Testing	<a href="#">Elimina</a> <a href="#">Configura UserStory</a> <a href="#">Configura Issue</a>

**Elimina:** Rimuove definitivamente l'associazione tra il piano di Microsoft Planner e il progetto Taiga. Dopo l'eliminazione, la sincronizzazione automatica tra task e user story non sarà più attiva.

**Configura UserStory:** Permette di associare gli stati delle "user story" di Taiga ai bucket di Planner. Quando si modifica manualmente lo stato di una user story in Taiga, la task associata in Planner verrà automaticamente spostata nel bucket corrispondente, in base alla configurazione effettuata. Ad esempio, se associ lo stato "Analizzata e Stimata" al bucket "Analizzata e Stimata", quando la user story raggiunge questo stato in Taiga, la task sarà spostata automaticamente in quel bucket su Planner.

**Configura Issue:** Funziona in modo simile alla configurazione delle user story, ma per le issue. Puoi associare gli stati delle issue di Taiga ai bucket in Planner. Anche qui, quando cambi manualmente lo stato di un'issue in Taiga, la task collegata in Planner verrà automaticamente spostata nel bucket corrispondente. Ad esempio, se imposti lo stato "In progress" per una issue, la task sarà automaticamente spostata nel bucket "In Sviluppo" in Planner.

**Figura 5.25:** Pagina Gestisci Associazioni.

#### Funzionalità e Dettagli sui Comandi

Le principali funzionalità della pagina includono:

- **Eliminazione di un'associazione:** ogni riga della tabella riporta un'associazione tra un piano di *Microsoft Planner* e un progetto di *Taiga*. Tramite un pulsante dedicato, l'utente può rimuovere l'associazione, interrompendo così la sincronizzazione automatica tra le *task* di *Planner* e le *user story* o *issue* del progetto di *Taiga* corrispondente.
- **Configurazione dello stato User Story:** tramite un pulsante, è possibile accedere a una finestra di configurazione (<http://localhost:8080/config>) per associare gli stati delle *user story* di *Taiga* ai *bucket* di *Planner*. Questo permette di sincronizzare automaticamente i cambiamenti di stato delle *user story* con lo spostamento delle *task* nei *bucket* corrispondenti in *Planner*.
- **Configurazione dello stato Issue:** in modo analogo alla configurazione delle *user story*, è disponibile un pulsante per accedere alla pagina di configurazione (<http://localhost:8080/config-issue>). Qui è possibile associare gli stati delle *issue* di *Taiga* ai *bucket* di *Planner*, garantendo una sincronizzazione automatica dei cambiamenti.

**Personalizzazione del ciclo di vita** Le funzionalità di configurazione dello stato delle *user story* e delle *issue* consentono di adattare o sovrascrivere il ciclo di vita *standard* definito nei requisiti (si veda Sezione 4.1.1 e Sezione 4.1.1). Questo è particolarmente utile per le aziende che hanno iniziato da poco a utilizzare strumenti come *Planner* e *Taiga*, dove ogni piano o progetto può avere esigenze specifiche.

Ad esempio, è possibile aggiungere, eliminare o modificare i *bucket* di *Planner* o gli stati delle *user story* e delle *issue* di *Taiga*. Nel caso in cui non vengano apportate modifiche per uno specifico *bucket* o stato, il sistema segue automaticamente il ciclo di vita *standard* previsto.

Questa flessibilità permette di implementare le modifiche "a caldo", adattando dinamicamente il flusso di lavoro senza interruzioni operative.

### Esempio di sincronizzazione

Supponiamo che un progetto *Taiga* includa una *user story* con stato *In Progress*. Se questo stato è stato configurato per corrispondere al *bucket In Sviluppo* in *Planner*, quando lo stato della *user story* cambia in *In Progress*, la task associata verrà automaticamente spostata nel *bucket In Sviluppo* nel piano corrispondente di *Planner*.

Un processo analogo si applica anche per le *issue* di *Taiga*, in cui ogni cambiamento di stato viene riflesso automaticamente nel *bucket* associato in *Planner*. Questo approccio garantisce una sincronizzazione costante e un allineamento accurato tra i due strumenti.

#### 5.11.4 Pagina Configurazione Collegamenti (UserStory)

La pagina *Configura Collegamenti per Progetto e Piano (UserStory)*, illustrata in Figura 5.26, consente di definire associazioni personalizzate tra gli stati delle *user story* di *Taiga* e i *bucket* di *Microsoft Planner*. Questa funzionalità permette di adattare il comportamento del sistema alle specifiche esigenze di ogni progetto o piano.

## Configura Collegamenti per Progetto e Piano (UserStory)

[Torna alla Gestione](#)

Seleziona Bucket (Planner)

Seleziona un bucket

Seleziona Stato User Story (Taiga)

Seleziona uno stato

[Salva Collegamento](#)

### Collegamenti Esistenti

Qui puoi vedere i collegamenti esistenti tra i bucket di Planner e gli stati delle user story di Taiga. [Gestisci Collegamenti](#)

Bucket	Stato User Story
In attesa informazioni	New

**Salva Collegamento:** Salva un nuovo collegamento tra uno stato specifico di una User Story (Taiga) e un bucket di Planner. Quando una User Story cambia stato in Taiga, la task corrispondente verrà automaticamente spostata nel bucket configurato in Planner.

**Gestisci Collegamenti:** Permette di visualizzare, modificare o eliminare i collegamenti già esistenti tra stati di User Story e bucket di Planner. Questo ti aiuta a mantenere sincronizzate le tue attività nei due sistemi.

**Figura 5.26:** Pagina di configurazione per collegamenti personalizzati tra bucket e stati delle user story.

### Dettagli operativi

La pagina offre le seguenti funzionalità principali:

- **Selezione del Bucket:** un *menù* a tendina consente di scegliere uno dei *bucket* disponibili nel piano di *Planner*.
- **Selezione dello Stato User Story:** è possibile associare un *bucket* di *Planner* a uno stato specifico delle *user story* in *Taiga*. Questa associazione viene salvata tramite il pulsante *Salva Collegamento*.
- **Visualizzazione dei collegamenti esistenti:** una tabella mostra tutti i collegamenti già definiti. Questi collegamenti controllano il comportamento delle *task* in *Planner* quando lo stato di una *user story* cambia in *Taiga*.
- **Gestione dei collegamenti esistenti:** è possibile accedere a una pagina dedicata per gestire i collegamenti già definiti. In questa sezione, l'utente può visualizzare o eliminare i collegamenti esistenti.

### Personalizzazione e ciclo di vita

I collegamenti configurati nella tabella sopra riportata consentono di personalizzare il ciclo di vita *standard* definito dal requisito 4.1.1. Tale requisito stabilisce il comportamento predefinito del sistema, che segue lo schema riportato di seguito:

- *In Progress* → *In Sviluppo*;
- *Ready for test* → *In Test*;
- *Done* → *In Produzione*;
- *Need info* → *In attesa informazioni*;
- *Rejected* → *Annullate*.

Quando un'associazione personalizzata viene definita, il sistema utilizza tale configurazione per sovrascrivere il ciclo di vita *standard*, ma solo per lo stato o il *bucket* specificato.

Ad esempio, come mostrato in Figura 5.26, se un *bucket* denominato *Idee* viene associato allo stato *Need info*, una modifica dello stato di una *user story* in *Need info* comporterà lo spostamento della *task* corrispondente nel *bucket Idee*.

### Comportamento predefinito

Gli stati delle *user story* non configurati nella tabella dei collegamenti continueranno a seguire il ciclo di vita *standard* definito dal requisito 4.1.1. Ad esempio, una *user story* che passa allo stato *Ready for test* verrà automaticamente spostata nel *bucket In Test*, a meno che non sia stata configurata una diversa associazione.

Questa flessibilità permette di adattare dinamicamente i flussi di lavoro alle esigenze operative, mantenendo comunque una base *standard* di comportamento per gli stati e i *bucket* non personalizzati.

### 5.11.5 Pagina Configurazione Collegamenti (Issue)

La pagina *Configura Collegamenti per Progetto e Piano (Issue)*, è analoga a quella descritta per le *user story* nella Sezione [5.11.4](#), con la differenza che gestisce le associazioni tra gli stati delle *issue* di *Taiga* e i *bucket* di *Planner*. Il comportamento predefinito è regolato dal requisito [4.1.1](#).

## Capitolo 6

# Integrazione Aziendale

Dopo aver completato lo sviluppo, la validazione e la documentazione della *PoC*, durante l'ultima settimana di stage ho esplorato diverse strategie per integrarla nell'ambiente aziendale. Tuttavia, tutte le soluzioni proposte si sono rivelate impraticabili, principalmente a causa di limitazioni tecniche o operative.

In questa sezione, analizzerò i tentativi effettuati, descrivendo ogni soluzione in dettaglio e spiegando i motivi per cui è stata scartata o modificata.

### 6.1 Soluzione 1: Polling

#### 6.1.1 Descrizione della Soluzione

La prima soluzione analizzata, in accordo con il *tutor* aziendale Enrico Merigliano, è stata quella di implementare un sistema di *polling*. Questo metodo prevedeva un confronto continuo tra i dati attuali e quelli precedentemente memorizzati.

Ad esempio, ogni modifica effettuata su *Taiga* o *Planner* sarebbe stata rilevata mediante una comparazione automatica e continua dei rispettivi *dataset*, al fine di mantenere i due sistemi sincronizzati.

#### 6.1.2 Considerazioni Tecniche

L'implementazione del *polling* è semplice dal punto di vista tecnico, poiché non richiede la configurazione di *endpoint* pubblici o [webhook](#), opzioni rifiutate sin dall'inizio per motivi di sicurezza.

L'azienda non desiderava aprire *endpoint* sui propri *server* per ricevere notifiche da fonti esterne, considerando i costi di gestione e i rischi legati all'apertura di punti di accesso alla rete aziendale.

Tuttavia, il *polling* presentava diversi svantaggi operativi. Poiché il sistema memorizzava copie dei dati per confrontarle a intervalli regolari, si sarebbero dovuti mantenere in memoria sia i dati attuali sia quelli aggiornati ogni pochi secondi, creando una duplicazione onerosa.

La sincronizzazione tra i sistemi risultava particolarmente lenta e imprevedibile, specialmente in presenza di progetti con migliaia di *task*, dove i confronti ripetuti avrebbero impattato negativamente sulle prestazioni.

### 6.1.3 Analisi della Fattibilità

Questo metodo è stato scartato poiché la sua applicazione pratica si è rivelata impraticabile. Ad esempio, la gestione di un progetto contenente 5000 *task* avrebbe richiesto un carico computazionale elevato, causando ritardi significativi e un degrado delle prestazioni complessive del sistema.

## 6.2 Soluzione 2: Utilizzo dei Webhook

### 6.2.1 Descrizione della Soluzione

Una seconda opzione esaminata è stata l'utilizzo dei *webhook*. Questa tecnica prevedeva la registrazione di notifiche *push* su *Microsoft Graph* e *Taiga* per rilevare in tempo reale le modifiche effettuate.

A differenza del *polling*, i *webhook* inviano notifiche solo in caso di modifiche effettive, riducendo il carico di lavoro del sistema.

### 6.2.2 Problemi Operativi

Nonostante i vantaggi teorici dei *webhook*, la loro implementazione richiedeva l'apertura di un *endpoint* pubblico sul *server* aziendale, una misura che l'azienda aveva deciso di evitare.

Le preoccupazioni includevano sia i costi di gestione sia i rischi associati all'esposizione di punti di accesso alla rete interna.

Due varianti principali di questa soluzione sono state esplorate, ma entrambe hanno evidenziato limiti significativi.

#### Esposizione dell'App su Internet

In questa variante, l'applicazione sviluppata avrebbe dovuto essere esposta a un *endpoint* pubblico per ricevere notifiche dai *webhook*.

Sebbene questa soluzione garantisse piena gestione e flessibilità, i rischi di sicurezza e i tempi di sviluppo richiesti l'hanno resa impraticabile.

#### Implementazione su Azure con Power Automate

La seconda variante prevedeva l'utilizzo di *Power Automate* per gestire i *webhook* tramite un *trigger HTTP*.

Questo approccio si basava interamente su *cloud*, evitando così l'utilizzo del *server* aziendale e garantendo una gestione più sicura delle notifiche.

Tuttavia, presentava limiti significativi, tra cui un ritardo fisso di circa cinque minuti nella sincronizzazione e la necessità di implementare *script* specifici in *Power Automate* per rilevare le modifiche nei *task* di *Planner*.

Inoltre, la sottoscrizione a *Microsoft Graph* scadeva ogni 72 ore, richiedendo un rinnovo costante per garantire la funzionalità.



## 6.3 Possibile Alternativa: Migrazione a SharePoint

### 6.3.1 Descrizione della Soluzione

Un'altra soluzione considerata è stata la migrazione della gestione dei *task* da *Planner* a *SharePoint*. Utilizzando le liste di *SharePoint*, sarebbe stato possibile monitorare le modifiche tramite i *trigger* nativi di *Power Automate*.

Questa opzione permetteva di rilevare automaticamente le modifiche senza la necessità di costruire logiche aggiuntive.

Per rendere l'interfaccia di *SharePoint* simile a quella di *Planner*, si sarebbe potuto personalizzare il *layout* attraverso i suoi strumenti integrati, come illustrato in Figura 6.1

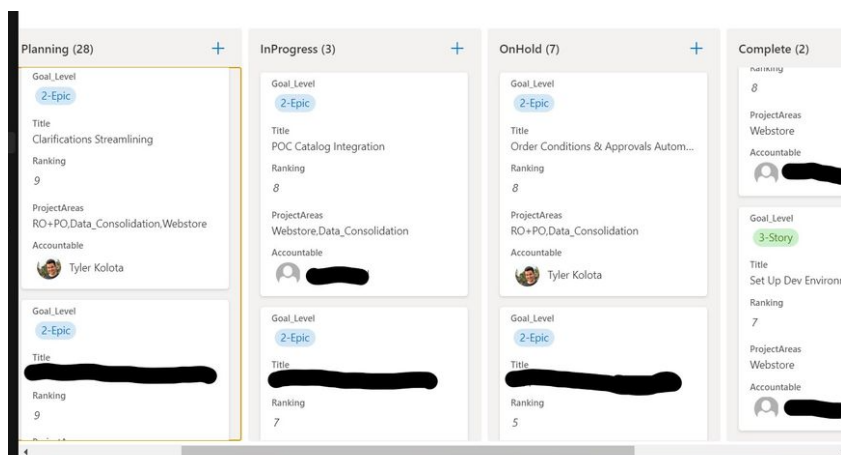


Figura 6.1: Pagina di configurazione per collegamenti personalizzati tra bucket e stati delle user story.

Una soluzione di questo tipo avrebbe migliorato il controllo sui dati, eliminato i limiti delle *API* e evitato l'esposizione di *endpoint* pubblici.

### 6.3.2 Conclusioni sulla Fattibilità

Nonostante i vantaggi, questa soluzione richiedeva una migrazione completa a una nuova piattaforma ed è stata scartata.

## 6.4 Soluzione Finale

### 6.4.1 Descrizione della Soluzione

La soluzione finale scelta combinava elementi dell'opzione basata su *webhook* con un approccio innovativo.

Per ricevere notifiche da *Taiga*, è stato utilizzato un *trigger HTTP* gestito da *Power Automate*, evitando così l'apertura di *endpoint* pubblici sul *server* aziendale.

Per le notifiche da *Planner*, dove non era disponibile un *trigger* nativo per rilevare modifiche ai *task*, è stato introdotto un utente fittizio. Questo utente veniva assegnato manualmente ai *task* modificati, attivando un flusso in *Power Automate* che

sincronizzava le modifiche con *Taiga*. Sebbene meno automatizzata, questa soluzione si è rivelata efficace e conforme ai requisiti aziendali.

## 6.4.2 Realizzazione della Soluzione

La soluzione è stata implementata attraverso due flussi principali:

- **Sincronizzazione da Planner a Taiga:** quando un *task* veniva assegnato all'utente fittizio, il flusso identificava i campi da aggiornare e li sincronizzava con la corrispondente *user story* su *Taiga*.
- **Sincronizzazione da Taiga a Planner:** le modifiche alle *user story* su *Taiga* venivano ricevute tramite *webhook* e propagate ai *task* corrispondenti su *Planner*.

## 6.4.3 Sincronizzazione da Planner a Taiga

### Creazione del Flusso in Power Automate

Un flusso, in *Power Automate*, è una sequenza di azioni automatizzate attivate da eventi specifici. Per implementare questa sincronizzazione, è stato creato un nuovo flusso *cloud* automatizzato.

### Analisi dei Trigger di Planner

Tra i *trigger* disponibili (Figura 6.2):

- Quando un'attività viene assegnata a un utente.
- Quando un'attività viene completata.
- Quando viene creata una nuova attività.

È stato scelto il primo, poiché rappresentava l'unico metodo per rilevare indirettamente una modifica. Assegnando il *task* all'utente fittizio, il flusso veniva avviato.

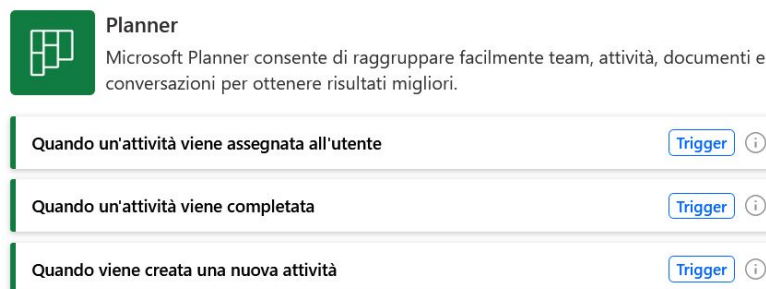


Figura 6.2: Trigger Planner disponibili in *Power Automate*.

## Spiegazione del Flusso

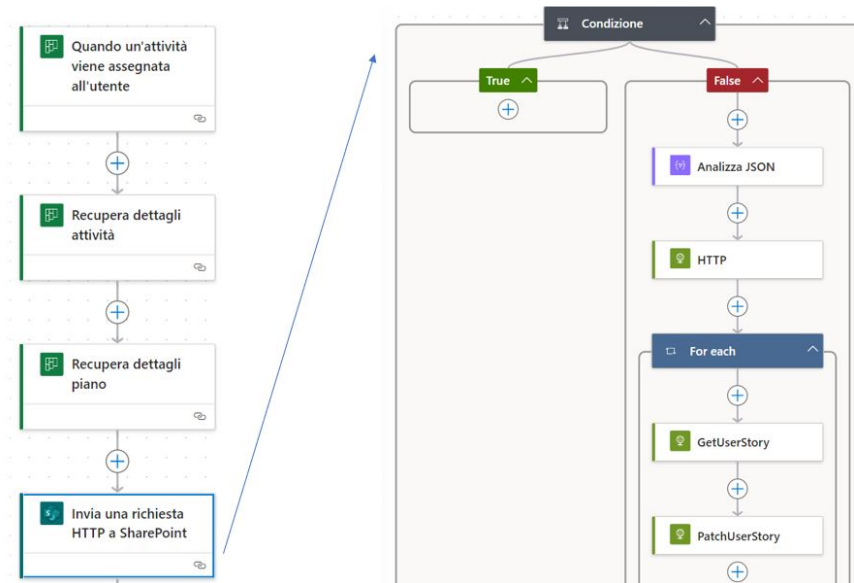


Figura 6.3: Schema del flusso Planner-Taiga in *Power Automate*.

Come illustrato nella Figura 6.3, il flusso è stato progettato per eseguire i seguenti passi:

1. Attivazione: il *trigger* “Quando un’attività viene assegnata a un utente” avvia il flusso. La frequenza minima di controllo è stata impostata a 1 minuto.
2. Recupero dei dettagli del *task* : il flusso acquisisce i dettagli dell’attività assegnata e del piano in cui è stata modificata.
3. Associazione con una *user story* in *Taiga*: attraverso una chiamata *HTTP* a una tabella in *SharePoint*, vengono mappati i *task* di *Planner* alle *user story* di *Taiga*.
  - La tabella su *SharePoint* contiene una riga per ogni associazione *task-user story*.
  - Il flusso utilizza la *query*:  

```
_api/web/lists/getbytitle('PlannerToTaigaID')/Items?$filter=LinkTitle eq '@{outputs('Recupera_dettagli_attività')}?['body/id']'
```
  - Se non viene trovata alcuna corrispondenza, il flusso si interrompe.
4. Aggiornamento della *user story*: se viene trovata una corrispondenza, il flusso autentica l’applicazione su *Taiga* tramite una chiamata *HTTP POST* per ottenere un *token*, e utilizza una chiamata *PATCH* per aggiornare la *user story*, sovrascrivendo tutti i campi.

#### 6.4.4 Sincronizzazione da Taiga a Planner

Per sincronizzare le modifiche da *Taiga* a *Planner*, è stato configurato un *webhook* direttamente dall'interfaccia di amministrazione di *Taiga*.

L'obiettivo era utilizzare un *trigger HTTP* in *Power Automate* per ricevere notifiche di modifica da *Taiga*.

##### Configurazione del Webhook

Il *trigger* scelto in *Power Automate* era "Quando ricevi una richiesta HTTP", che generava un *URL* univoco (si veda Figura 6.4) da utilizzare come *endpoint* per il *webhook*.

Nelle impostazioni di *Taiga*, questo *URL* è stato copiato e incollato nel campo dedicato al *webhook*. Tuttavia, è emersa una limitazione: *URL* del *trigger*, lungo 227 caratteri, superava il limite massimo consentito da *Taiga* (200 caratteri).

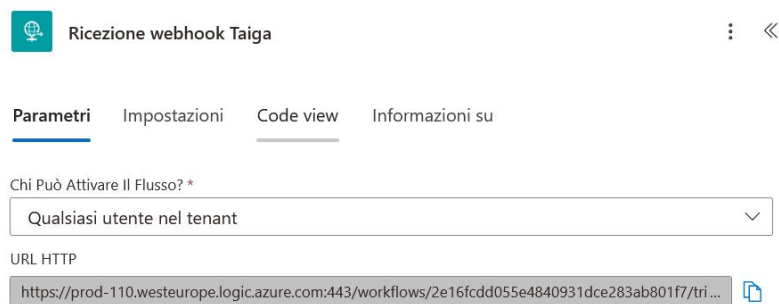


Figura 6.4: URL generato dal *trigger HTTP* in *Power Automate*.

##### Tentativo di Soluzione con URL Shortener

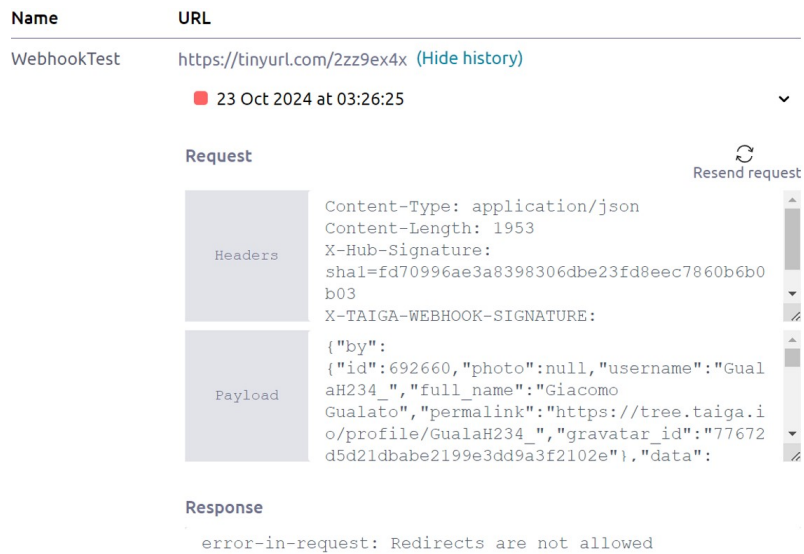
Per aggirare la limitazione della lunghezza *URL*, è stato utilizzato un servizio di accorciamento degli *URL*, come *TinyURL*.

L'idea era di creare un link più breve che reindirizzasse al *trigger HTTP* generato da *Power Automate*.

Nonostante il successo nell'accorciare *URL*, *Taiga* ha rilevato il reindirizzamento come una pratica non consentita e ha bloccato l'uso *URL* abbreviato, restituendo un errore specifico illustrato nella Figura 6.5.

##### Impossibilità della Soluzione

Le restrizioni imposte da *Taiga* hanno reso impossibile configurare correttamente il *webhook*. Senza un *URL* che soddisfacesse i requisiti di lunghezza e senza alternative accettabili, la sincronizzazione da *Taiga* a *Planner* non è risultata fattibile.



The screenshot shows a webhooks testing interface. At the top, there is a table with two columns: 'Name' and 'URL'. The first row contains 'WebhookTest' and 'https://tinyurl.com/2zz9ex4x (Hide history)'. Below the table, a red square icon indicates a failed request at '23 Oct 2024 at 03:26:25'. The 'Request' section is expanded, showing 'Headers' and 'Payload'. The headers include 'Content-Type: application/json', 'Content-Length: 1953', 'X-Hub-Signature: sha1=fd70996ae3a8398306dbe23fd8eec7860b6b0b03', and 'X-TAIGA-WEBHOOK-SIGNATURE:'. The payload is a JSON object: `{ "by": { "id": 692660, "photo": null, "username": "GualaH234_", "full_name": "Giacomo Gualato", "permalink": "https://tree.taiga.io/profile/GualaH234_", "gravatar_id": "77672d5d21dbabe2199e3dd9a3f2102e"}, "data":`. The 'Response' section shows the error message: `error-in-request: Redirects are not allowed`.

Figura 6.5: Errore di *Taiga* durante l'uso di un URL abbreviato.

### 6.4.5 Conclusioni sulla Sincronizzazione

La sincronizzazione da *Planner* a *Taiga* è stata implementata con successo grazie all'uso dell'utente fittizio e di *Power Automate*.

Al contrario, la sincronizzazione da *Taiga* a *Planner* è stata ostacolata da limiti tecnici legati alla lunghezza degli *URL* dei *webhook*, senza soluzioni alternative praticabili.

## Capitolo 7

# Conclusioni

Il progetto ha avuto come obiettivo principale la sincronizzazione tra le piattaforme *Planner* e *Taiga*, garantendo il rispetto dei requisiti aziendali.

Attraverso un'attenta analisi iniziale e una pianificazione efficace, ho sviluppato una soluzione *server-to-server* che automatizza i flussi di lavoro, migliorando l'efficienza dei processi aziendali.

Durante il percorso, ho affrontato sfide tecniche e operative, trovando soluzioni innovative e documentando eventuali limitazioni riscontrate.

La sincronizzazione implementata permette di replicare automaticamente nell'altro sistema le modifiche effettuate in uno dei due, garantendo coerenza e riducendo errori.

Per supportare questa funzionalità, ho utilizzato diversi strumenti, inclusi servizi di log che consentono di mantenere uno storico dettagliato delle operazioni effettuate.

Nell'integrazione aziendale, ho esplorato l'implementazione di una soluzione completamente basata su *cloud*, con l'obiettivo di evitare il carico computazionale interno all'azienda; tuttavia, la realizzazione è stata limitata da vincoli tecnici e operativi.

### 7.1 Raggiungimento degli Obiettivi

Il progetto ha raggiunto tutti gli obiettivi definiti nel capitolo 4, rispettando i requisiti obbligatori, facoltativi e desiderabili. La sincronizzazione tra *Planner* e *Taiga* è stata completata con successo in conformità alle esigenze aziendali.

L'unica eccezione riguarda il requisito 4.1.1, la cui non fattibilità è stata dimostrata e documentata. Questo risultato ha permesso di fornire un quadro chiaro delle limitazioni tecniche, evidenziando l'importanza di considerare approcci alternativi quando necessario.

Oltre agli obiettivi funzionali, ho prodotto l'intera documentazione richiesta, tra cui manuali tecnici e utente, istruzioni di installazione e report di analisi. Tutti i vincoli tecnici e aziendali sono stati rispettati, garantendo una soluzione in linea con le aspettative del progetto.

Infine, pur non essendo esplicitamente richiesto, ho dedicato del tempo a integrare la *PoC* in azienda. Tuttavia, a causa di limitazioni tecniche relative agli *URL* dei *webhook*, la sincronizzazione da *Taiga* a *Planner* non è stata completata. Anche in questo caso, la non fattibilità è stata documentata in dettaglio.

## 7.2 Conoscenze Acquisite

Questo progetto è stato un'importante occasione di crescita professionale. Tra le principali competenze sviluppate, si evidenziano:

- **Miglioramento della padronanza di Java:** Ho approfondito le operazioni sincrone e asincrone, migliorando la mia capacità di gestire la comunicazione tra sistemi e di ottimizzare le risorse disponibili.
- **Uso avanzato di strumenti di sviluppo:** L'impiego di *IntelliJ IDEA* e *Spring Boot* ha migliorato la qualità e l'efficienza del mio codice, aumentando la mia competenza nell'utilizzo di strumenti per lo sviluppo software.
- **Conoscenza della suite Microsoft 365:** Grazie a strumenti come *Teams*, *Outlook*, *Planner*, *SharePoint* e *Power Automate*, ho acquisito una solida esperienza nell'uso di piattaforme aziendali per la gestione collaborativa.
- **Gestione e pianificazione di progetti:** L'esperienza mi ha permesso di affinare le mie competenze nella pianificazione e nel monitoraggio delle attività, competenze già affrontate durante gli studi universitari ma ora messe in pratica in un contesto reale.

## 7.3 Esperienza Professionale e Formazione Universitaria

L'esperienza professionale ha completato e ampliato le competenze acquisite durante la formazione universitaria. Seppur i corsi di *Tecnologie Web* e *Ingegneria del Software* mi abbiano fornito una solida base teorica, questa esperienza mi ha permesso di affrontare per la prima volta una vera sfida aziendale.

Ho messo in pratica concetti teorici appresi durante il percorso accademico, adattandoli a situazioni complesse e dinamiche. La capacità di adattamento rapido e l'applicazione concreta delle soluzioni si sono rivelate cruciali, dimostrando quanto il contesto professionale possa valorizzare e rafforzare la preparazione universitaria.

Inoltre, il lavoro di squadra e l'interazione con colleghi esperti mi hanno insegnato l'importanza della comunicazione e della collaborazione, aspetti meno tangibili ma fondamentali per il successo in un ambiente aziendale.

## 7.4 Analisi Temporale dello Stage

La gestione del tempo è stata un punto di forza durante lo *stage*. Il piano iniziale è stato rispettato, concludendo le attività previste entro i tempi stabiliti. Inoltre, l'ultima settimana è stata dedicata all'ottimizzazione delle soluzioni e alla loro integrazione aziendale, consentendo di affinare ulteriormente il progetto.

## 7.5 Conclusioni Personali

Questa esperienza è stata altamente formativa, sia dal punto di vista tecnico che personale. Lavorare su un progetto complesso come la sincronizzazione tra *Planner* e *Taiga* mi ha permesso di affrontare problemi reali, trovando soluzioni innovative e consolidando le mie competenze.

Le difficoltà iniziali, come l'analisi delle tecnologie e la gestione delle problematiche tecniche, sono state superate grazie a un approccio metodico e al supporto dei colleghi. Passo dopo passo, ho costruito una soluzione che rispecchia le esigenze aziendali, rendendomi orgoglioso del risultato ottenuto.

Infine, questa esperienza mi ha permesso di apprezzare l'importanza della documentazione e della gestione del tempo, competenze che ritengo fondamentali per il mio futuro professionale. Se in futuro si presenteranno nuove opportunità di migliorare o estendere il progetto, sarò entusiasta di contribuire.

Sono grato per questa esperienza, che ha rafforzato la mia passione per lo sviluppo software e per le sfide che il mondo professionale offre quotidianamente.



# Acronimi e abbreviazioni

- API** Application Program Interface. ii, 97
- CEO** Chief Executive Officer. 2, 97
- EDR** Endpoint Detection and Response. 98
- ERP** Enterprise Resource Planning. 2, 98
- FAQ** Frequently Asked Questions. 33, 96
- HTML** HyperText Markup Language. 22, 96
- HTTP** HyperText Transfer Protocol. 21, 98
- ICT** Information and Communication Technology. 2, 99
- IDE** Integrated Development Environment. 22, 96
- JSON** JavaScript Object Notation. 21, 99
- PMI** Piccole e Medie Imprese. 2, 99
- PoC** Proof of Concept. 1, 99
- R&D** Ricerca e Sviluppo. 11, 100
- SSDLC** Secure Software Development Life Cycle. 3, 100

# Glossario

**agile** *Agile* è un insieme di metodologie di sviluppo software e gestione dei progetti che enfatizza la flessibilità, la collaborazione e l'adattamento ai cambiamenti. Basata su cicli iterativi e incrementali, la metodologia Agile mira a consegnare valore continuo agli utenti e a migliorare il prodotto attraverso feedback frequenti, rilasci rapidi e un alto coinvolgimento del cliente. [1](#), [97](#)

**API** in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. [96](#)

**Bearer** in informatica, il termine *Bearer* si riferisce a un tipo di token utilizzato per l'autenticazione nell'ambito di protocolli come OAuth 2.0. Un *Bearer token* è un token di accesso che consente a chiunque lo possieda di accedere a una risorsa protetta senza necessità di ulteriori verifiche. Per questo motivo, è fondamentale garantire la sicurezza del token, evitando che venga intercettato o utilizzato da terzi non autorizzati. [43](#)

**BugTask** nel contesto di *Microsoft Planner*, una BugTask è una *task* contrassegnata con il tag "Bug". Queste task sono utilizzate per gestire e risolvere problemi (*issue*) riscontrati nel progetto e corrispondono alle *issue* presenti nel progetto in *Taiga*. [27](#), [97](#)

**CEO** in una azienda con il termine *Chief Executive Officer* (ing. amministratore delegato) si indica l'amministratore delegato di un'azienda, responsabile delle decisioni strategiche e della gestione complessiva dell'organizzazione. Questa figura ha il compito di guidare l'azienda verso i suoi obiettivi, coordinando le varie funzioni operative e rappresentando il punto di riferimento principale per la direzione aziendale. [96](#)

**Client ID** Identificativo univoco di un'applicazione registrata su Azure Active Directory. Il *Client ID* viene utilizzato per identificare l'applicazione durante le richieste di autenticazione in contesti che utilizzano protocolli di autorizzazione come OAuth 2.0. [40](#), [97](#)

**Client Secret** In ambito informatico, un *Client Secret* è una stringa di caratteri generata per autenticare un'applicazione client in un sistema che utilizza protocolli

di autorizzazione come OAuth 2.0. Questo segreto, simile a una password, viene utilizzato insieme al ClientID per garantire la sicurezza delle comunicazioni tra il client e il server di autorizzazione. È fondamentale mantenerlo protetto e accessibile solo all'applicazione autorizzata per evitare accessi non autorizzati. [38](#), [97](#)

**cloud-based** Architettura o soluzione che sfrutta servizi e risorse ospitate nel cloud anziché su server locali o dispositivi fisici. Le soluzioni *cloud-based* offrono vantaggi come scalabilità, accessibilità da remoto e riduzione dei costi di manutenzione hardware, rendendole ideali per gestire dati e applicazioni in modo sicuro ed efficiente. [17](#), [98](#)

**drag-and-drop** Tecnica di interazione grafica che consente agli utenti di selezionare un oggetto, trascinarlo e rilasciarlo in una posizione desiderata all'interno di un'interfaccia. Questa modalità intuitiva è comunemente utilizzata in software e applicazioni per semplificare operazioni come organizzare elementi, costruire flussi di lavoro o configurare layout senza richiedere conoscenze tecniche avanzate. [18](#), [98](#)

**EDR** in informatica con il termine *Endpoint Detection and Response* (ing. Rilevamento e risposta degli endpoint) si indica una soluzione di sicurezza che monitora e analizza le attività sugli endpoint (dispositivi come computer e server) per identificare, rilevare e rispondere a minacce e attacchi informatici. [96](#)

**Endpoint** in informatica, un *endpoint* è un punto di accesso a un servizio o un sistema, che consente la comunicazione e lo scambio di dati tra dispositivi o applicazioni. Nel contesto della rete aziendale, un endpoint può rappresentare un'interfaccia di rete o un indirizzo specifico che permette a sistemi interni o esterni di interagire con un servizio, mantenendo la sicurezza e il controllo dell'accesso. [13](#), [98](#)

**ERP** in informatica con il termine *Enterprise Resource Planning* (ing. pianificazione delle risorse aziendali) si indica un sistema integrato di gestione aziendale progettato per raccogliere, archiviare, gestire e interpretare i dati delle attività di un'organizzazione. Gli ERP coordinano e ottimizzano i processi aziendali attraverso un'unica piattaforma software, facilitando la pianificazione delle risorse, il monitoraggio delle operazioni e il reporting. Le funzionalità tipiche di un sistema ERP includono la gestione della contabilità, delle vendite, della produzione, della logistica e delle risorse umane, contribuendo a migliorare l'efficienza operativa e la visibilità dei dati aziendali in tempo reale. [96](#)

**Group-ID** Identificativo univoco associato a un gruppo all'interno di *Microsoft Planner* o di altre piattaforme Microsoft. Il *group-id* viene utilizzato per identificare e sincronizzare i piani o le risorse associate a un determinato gruppo, facilitando l'integrazione e la gestione delle attività condivise. [26](#), [98](#)

**HTTP** in informatica, il termine *HyperText Transfer Protocol* (ing. protocollo di trasferimento di ipertesti) indica un protocollo di comunicazione utilizzato per il trasferimento di dati sul web. L'HTTP è alla base della comunicazione tra client e server e consente la trasmissione di richieste e risposte, utilizzate per la visualizzazione e gestione delle pagine web. [96](#)

- ICT** in informatica con il termine *Information and Communication Technology* (ing. tecnologia dell'informazione e della comunicazione) si indica l'insieme delle tecnologie che consentono la raccolta, l'archiviazione, la gestione e la trasmissione di informazioni attraverso sistemi informatici e reti di comunicazione. [96](#)
- JSON** in informatica, il termine *JavaScript Object Notation* (ing. Notazione a Oggetti JavaScript) indica un formato leggero di scambio dati, facilmente leggibile da esseri umani e macchine. Viene utilizzato per serializzare strutture dati complesse come oggetti e array, ed è largamente impiegato nelle API web per facilitare la comunicazione tra server e client. [96](#)
- Kanban** *Kanban* è una metodologia di gestione del flusso di lavoro e dei processi, originariamente sviluppata per la produzione industriale e successivamente adattata alla gestione dei progetti. La metodologia si basa sull'utilizzo di una bacheca visiva (spesso suddivisa in colonne come "Da fare," "In corso," e "Completato") per monitorare e ottimizzare il flusso delle attività, promuovendo la trasparenza, la riduzione degli sprechi e l'efficienza attraverso un sistema di miglioramento continuo. [1](#), [99](#)
- Lambda** in informatica, una *lambda expression* o *espressione lambda* è una funzione anonima, spesso utilizzata per definire comportamenti concisi e senza nome da passare come parametri a metodi o per implementare interfacce funzionali. Le espressioni lambda sono ampiamente utilizzate nei linguaggi. [22](#), [99](#)
- Object ID** Identificativo univoco di un'entità in Azure Active Directory. L'*Object ID* è assegnato automaticamente a oggetti come utenti, gruppi, applicazioni o dispositivi e viene utilizzato per identificarli in modo inequivocabile all'interno del tenant. [40](#), [99](#)
- open-source** in informatica, il termine *open-source* si riferisce a un modello di sviluppo del software in cui il codice sorgente è reso disponibile pubblicamente e può essere visualizzato, modificato e distribuito liberamente. I progetti open-source incoraggiano la collaborazione tra sviluppatori e aziende, consentendo una maggiore trasparenza e la possibilità di miglioramenti continui grazie ai contributi della comunità. [1](#), [99](#)
- Outsourcing** in informatica con il termine *Outsourcing* (ing. esternalizzazione) si indica la pratica aziendale di esternalizzare specifiche funzioni o processi a fornitori esterni specializzati, al fine di ottimizzare le risorse interne e migliorare l'efficienza operativa. Attraverso l'outsourcing, un'azienda delega attività come la gestione IT, la produzione, o i servizi di supporto, concentrandosi così sulle competenze principali. [2](#), [99](#)
- PMI** si indicano aziende caratterizzate da dimensioni ridotte e medie, che svolgono un ruolo importante nell'economia. [96](#)
- PoC** in informatica con il termine *Proof of Concept* (ing. prova di concetto) si indica una realizzazione preliminare volta a dimostrare la fattibilità di un'idea o di un metodo. Viene utilizzata per verificare la validità di un concetto o una soluzione tecnica prima di investire risorse nello sviluppo completo. [96](#)

**POST** Metodo del protocollo HTTP utilizzato per inviare dati al server. Generalmente utilizzato per creare o aggiornare risorse sul server, una richiesta *POST* include i dati nel corpo della richiesta, che possono essere in vari formati come JSON o XML. Nel contesto delle API, il metodo *POST* è spesso impiegato per inviare informazioni di autenticazione, come richieste per ottenere token di accesso tramite endpoint specifici. [42](#), [97](#), [100](#)

**Project Management** in ambito aziendale e tecnologico, il termine *Project Management* (ing. gestione dei progetti) si riferisce all'insieme di metodologie, tecniche e pratiche utilizzate per pianificare, eseguire e completare un progetto in modo efficiente e nei tempi previsti. Il project management include la gestione di risorse, budget, rischi e tempistiche per raggiungere obiettivi specifici, rispettando i vincoli di qualità e risorse. Il responsabile di progetto (Project Manager) coordina il lavoro dei vari team, monitora l'avanzamento e gestisce eventuali problemi per assicurare il successo del progetto. [1](#), [100](#)

**R&D** abbreviazione di *Ricerca e Sviluppo*, un'area aziendale che si occupa della progettazione e dell'innovazione di nuovi prodotti, tecnologie e processi per migliorare la competitività e rispondere a esigenze di mercato. L'R&D è fondamentale per l'innovazione aziendale e coinvolge attività di studio, sperimentazione e test. [96](#)

**RegularTask** nel contesto di *Microsoft Planner*, una RegularTask è una *task* priva del tag "Bug". Queste task rappresentano attività ordinarie del progetto e corrispondono alle *user story* definite nel progetto in *Taiga*. [27](#), [100](#)

**RESTful** acronimo di *Representational State Transfer*, un'architettura per la progettazione di servizi web basati su protocolli stateless. Le API RESTful usano richieste HTTP per eseguire operazioni su risorse server, rendendole scalabili e facilmente integrabili in applicazioni distribuite. [21](#), [100](#)

**Scrum** *Scrum* è una metodologia Agile specifica per la gestione dei progetti, in particolare per lo sviluppo software. Basato su cicli di lavoro brevi e strutturati chiamati *sprint*, Scrum si concentra sul lavoro di team interfunzionali che collaborano in modo iterativo per produrre versioni incrementali del prodotto. I ruoli principali in Scrum includono il Product Owner, il Scrum Master e il team di sviluppo, ognuno dei quali ha compiti definiti per garantire il successo del progetto. [1](#), [100](#)

**server-to-server** Modalità di comunicazione in cui due server interagiscono direttamente tra loro, scambiando dati e richieste senza necessità di intervento umano o di un client intermediario. Questa modalità è utilizzata per automatizzare processi, garantire sincronizzazione e coerenza dei dati tra sistemi diversi. [ii](#), [100](#)

**SSDLC** in informatica con il termine *Secure Software Development Life Cycle* (ing. Ciclo di vita sicuro dello sviluppo software) si indica un processo integrato per lo sviluppo di software che include pratiche di sicurezza in ogni fase del ciclo di vita del progetto. [96](#)

**System Integrator** in informatica con il termine *System Integrator* (ing. integratore di sistemi) si indica un'azienda o un professionista specializzato nell'integrazione di diversi sistemi e componenti tecnologici in un'unica soluzione coesa. Questa figura professionale si occupa di analizzare le esigenze specifiche di un'organizzazione, progettare soluzioni personalizzate e garantire l'efficienza del sistema. [2](#), [100](#)

**Tenant** In informatica, con il termine *tenant* si indica un'entità logica rappresentativa di un'organizzazione all'interno di un ambiente multi-tenant, come quello di *Microsoft Planner*. Un tenant isola e gestisce le risorse e i dati di un'organizzazione, garantendo la separazione dagli altri tenant presenti nello stesso sistema. In questo contesto, il tenant consente di accedere e gestire i piani associati all'organizzazione senza interferire con i dati di altre organizzazioni o altri tenant.. [32](#), [101](#)

**Tenant ID** Identificativo univoco del *tenant* di Azure Active Directory su cui è registrata un'applicazione. Il *Tenant ID* consente all'applicazione di operare esclusivamente nel contesto di quel tenant, garantendo un livello di isolamento e sicurezza. [40](#), [101](#)

**Token** Stringa crittografata utilizzata per autenticare ed autorizzare richieste tra sistemi o applicazioni. Esistono diverse tipologie di token, tra cui il token delegato e il token applicativo, che si differenziano per il loro utilizzo e le modalità di emissione. I token sono fondamentali per garantire la sicurezza nelle comunicazioni tra API e sistemi distribuiti, minimizzando il rischio di accessi non autorizzati. [26](#), [101](#)

**Token applicativo** Tipologia di token emesso per identificare un'applicazione piuttosto che un utente specifico. Il token applicativo consente all'applicazione di autenticarsi autonomamente per eseguire operazioni predeterminate senza richiedere l'interazione di un utente. È particolarmente utile per scenari di integrazione tra sistemi o automazione di processi. [101](#)

**Token delegato** Tipologia di token emesso a nome di un utente specifico per autorizzare operazioni in suo nome. Il token delegato richiede l'autenticazione dell'utente e include informazioni sulle autorizzazioni consentite all'applicazione di terze parti che lo utilizza. È comunemente usato in contesti dove l'utente fornisce esplicito consenso per operazioni effettuate da un'applicazione esterna. [26](#), [101](#)

**Utente tecnico** Account creato appositamente per effettuare chiamate API utilizzando credenziali note, simile a un bot ma specificamente configurato per l'integrazione tra Microsoft Planner e Taiga. Questo tipo di account è progettato per automatizzare la replicazione delle modifiche apportate in Planner all'interno di Taiga, riducendo la necessità di intervento manuale da parte del team di sviluppo. L'utente tecnico deve essere aggiunto a tutti i progetti Taiga che si desiderano automatizzare, garantendo così l'accesso e l'efficacia delle operazioni automatizzate.. [26](#), [101](#)

**webhook** Tecnologia di callback HTTP che consente la comunicazione in tempo reale tra applicazioni. Un *webhook* permette a un sistema di notificare un altro sistema ogni volta che si verifica un evento specifico, inviando una richiesta HTTP contenente i dati relativi all'evento. I *webhook* sono comunemente utilizzati per automatizzare processi e integrazioni tra servizi diversi, eliminando la necessità di interrogazioni periodiche. [86](#), [101](#)

# Bibliografia

## Riferimenti bibliografici

Sommerville, Ian. *Software Engineering, 10th ed.* Addison Wesley, 2014.

## Siti web consultati

*API Planner.* URL: <https://learn.microsoft.com/en-us/graph/api/resources/planner-overview?view=graph-rest-1.0>.

*API SharePoint.* URL: <https://learn.microsoft.com/en-us/graph/api/resources/sharepoint?view=graph-rest-1.0>.

*API Taiga.* URL: <https://docs.taiga.io/api.html>.

*Documentazione Creazione App Azure.* URL: <https://learn.microsoft.com/en-us/entra/identity-platform/quickstart-register-app?tabs=certificate>.

*Documentazione IntelliJ-IDEA.* URL: <https://www.jetbrains.com/help/idea/getting-started.html>.

*Documentazione Spring Boot.* URL: <https://spring.io/>.

*Graph Explorer.* URL: <https://developer.microsoft.com/en-us/graph/graph-explorer>.

*Manifesto Agile.* URL: <http://agilemanifesto.org/iso/it/>.