Immagine che contiene arte, emblema, cerchio, simbolo

Descrizione generata automaticamente**Corso di Ingegneria del Software**

**A. A. 2024 - 2025**

**Gruppo n. 10**: **IT Asset Management**

***POLITECH***

IT ASSET MANAGEMENT

Immagine che contiene grafica, schermata, Elementi grafici, cartone animato

Descrizione generata automaticamente

**Docente:**

Marina Mongiello

**Studenti:**

Falcone Claudio Vincenzo

Sambo Manuel

Sambo Mikaela

# **Sommario**

[1. Introduzione 3](#_Toc186995737)

[1.1 Ambito applicativo 3](#_Toc186995738)

[1.2 L’asset management 3](#_Toc186995739)

[1.3 L’asset Management nelle piccole e medie imprese 4](#_Toc186995740)

[1.4 Analisi delle soluzioni esistenti 4](#_Toc186995741)

[2. Progetto “Politech”: fase iniziale 6](#_Toc186995742)

[2.1 Obiettivi 6](#_Toc186995743)

[2.2 Diagramma di Gantt: Fasi del progetto 6](#_Toc186995744)

[2.4 Rischi e mitigazione 7](#_Toc186995745)

[2.5 Stima dei costi 7](#_Toc186995746)

[3. Metodo proposto e requisiti 8](#_Toc186995747)

[3.1 Modello di processo adottato 8](#_Toc186995748)

[3.2 Fasi principali del metodo proposto: 8](#_Toc186995749)

[3.3 Requisiti 9](#_Toc186995750)

[4. Architettura e *Tech Stack* 10](#_Toc186995751)

[4.1 Architettura 10](#_Toc186995752)

[4.2 Tech Stack 10](#_Toc186995753)

[5. Prototipo 11](#_Toc186995754)

[6. Validazione e verifica 11](#_Toc186995755)

[7. Discussione 11](#_Toc186995756)

[8. Conclusioni e sviluppi futuri 12](#_Toc186995757)

[Sitografia e Fonti 13](#_Toc186995758)

# **1. Introduz****ione**

1.1 Ambito applicativo

Un programma di *IT Asset Management* consente alle aziende di gestire in modo efficiente il ciclo di vita dei propri asset informatici e delle proprie risorse di aziendali al fine di ottimizzarne distribuzione, gestione e controllo.

1.2 L’asset management

In questo contesto storico ed economico, l’ IT *asset management* (ITAM)è ormai ritenuto fondamentale per l’organizzazione e lo sviluppo delle imprese. Non si tratta solo di una necessità operativa, ma di un vero e proprio strumento strategico per ottimizzare l’efficienza aziendale valorizzando al meglio le risorse e riducendo costi e rischi.

Il *National Institute of Standards and Technology* (NIST), un’agenzia statunitense di ricerca e sviluppo nonché ente governativo con funzioni di standardizzazione e supporto tecnologico per le industrie, già nel 2018 aveva evidenziato l’importanza di implementare un sistema di *IT asset management*. In quella stessa occasione, il NIST aveva proposto una soluzione esemplificativa che pur rivolta principalmente al settore finanziario lasciava emergere chiaramente l’importanza, la versatilità e le funzionalità di un tale strumento.

*“ (…) At this scale, the technology base required to ensure smooth business operations (including computers, mobile devices, operating systems, applications, data, and network resources) is massive. To effectively manage, use, and secure each of those assets, you need to know their locations and functions. While physical assets can be labeled with bar codes and tracked in a database, this approach does not answer questions such as “What operating systems are our laptops running?” and “Which devices are vulnerable to the latest threat?” Computer security professionals in the financial services sector told us they are challenged by the vast diversity of hardware and software they attempt to track, and by a lack of centralized control: A large financial services organization can include subsidiaries, branches, third-party partners, contractors, as well as temporary workers and guests. This complexity makes it difficult to assess vulnerabilities or to respond quickly to threats, and to accurately assess risk in the first place (...)” [NIST SP 1800-5A: IT Asset Management”, Settembre 2018]*

Dunque, le funzionalità di un programma di ITAM sono molteplici e fondamentali per l’efficienza aziendale: vi è la possibilità di **ottimizzare l’utilizzo delle risorse disponibili**, riducendo al minimo inutilizzazioni e sottoutilizzazioni; un altro aspetto essenziale riguarda il **monitoraggio della conformità legale**, come il controllo delle scadenze delle licenze o la verifica della conformità delle attrezzature; un programma di asset management è, inoltre, cruciale per garantire la **sicurezza delle risorse aziendali**: monitorando costantemente le performance degli asset, è possibile pianificare interventi di manutenzione, anche preventiva, intervenendo prontamente in caso di problemi e prevenendo interruzioni che potrebbero compromettere l’efficienza operativa dell’azienda

Non va sottovalutata l’importanza che l’utilizzo di un programma di asset management può avere in termini di **sostenibilità e responsabilità ambientale**: attraverso il monitoraggio del ciclo di vita degli *asset*, è possibile una gestione più consapevole delle risorse aziendali, inclusa la loro dismissione, contribuendo così alla riduzione dell’*e-waste.*

1.3 L’asset Management nelle piccole e medie imprese

Le grandi aziende trovano indispensabili le soluzioni di *IT Asset Management* (ITAM), basti pensare a colossi come *Deloitte* e *PwC*, che da anni sottolineano l’importanza di queste pratiche non solo attraverso report e studi dedicati, ma anche tramite investimenti mirati e il finanziamento di progetti di sviluppo.

Si tratta di un’osservazione forse ovvia: gestire migliaia di dispositivi e dipendenti non è un’operazione fattibile manualmente. Tuttavia, ciò non significa che le soluzioni ITAM non siano altrettanto importanti per le Piccole e Medie Imprese (PMI).

Le PMI rappresentano la stragrande maggioranza delle imprese italiane ed europee e svolgono un ruolo cruciale nell’innovazione e nello sviluppo economico locale, non è , però, frequente che queste realtà si affidino al SaaS (*Software as a Service*) in generale o all’ITAM in particolare.

Le ragioni di tale reticenza sono molteplici; dal punto di vista economico, le PMI vedono l’ITAM come un **investimento costoso** che difficilmente riesce a generare un ritorno economico immediato, inoltre, i **programmi ITAM spesso si presentano come complessi** e difficili da integrare nelle infrastrutture IT esistenti, richiedendo personale qualificato di cui molte PMI non dispongono.

Dal punto di vista strategico è significativo che le PMI non siano consapevoli dei rischi informatici legati a software obsoleti o dispositivi non monitorati, inoltre la gestione IT è spesso vista come un onere, come un costo operativo, anziché come un investimento strategico e ciò limita la comprensione dei benefici a lungo termine che un buon ITAM può portare.

1.4 Analisi delle soluzioni esistenti

Tra i principali applicativi competitor sul mercato, due dei più utilizzati sono *Service Now* e *OCS Inventory*, a seguito di un’analisi è stato possibile individuarne limiti e criticità.

* ***Service Now***

*ServiceNow* è una delle soluzioni leader nel campo dell’ITAM, è sicuramente un programma che offre numerose funzionalità, tuttavia, non è una soluzione che si adatta ad ogni tipo di impresa.

Elementi principali di criticità sono:

1. **Costo Elevato:** *ServiceNow* è noto per essere una soluzione costosa, spesso più adatta a grandi aziende con budget significativi. Le PMI potrebbero trovare difficile giustificare l'investimento richiesto per l'implementazione e la manutenzione della piattaforma.
2. **Complessità:** La vasta gamma di funzionalità offerte da *ServiceNow* può risultare complessa da configurare e gestire. L'implementazione richiede spesso personale specializzato, il che aumenterebbe ulteriormente costi e tempi di adozione.

* ***OCS Inventory***

*OCS Inventory* è un’alternativa *open-source* per la gestione degli *asset IT*.

Si tratta di uno strumento molto popolare proprio tra le PMI proprio perché gratuito e *open-source*, tuttavia presenta alcuni limiti.

1. **Interfaccia Utente Limitata:** Dalle segnalazioni degli stessi utenti è emerso che l’interfaccia di *OCS Inventory* può risultare meno intuitiva e *user-friendly* rispetto ad altre soluzioni, rendendo l'esperienza d'uso meno agevole, specialmente per chi non ha competenze tecniche avanzate.
2. **Problemi di Aggiornamento:** Sono stati riportati casi in cui gli agenti di OCS Inventory non aggiornano correttamente le informazioni sul server, causando discrepanze nei dati degli asset. Questo può derivare da configurazioni errate o bug nel software.
3. **Dipendenza da Componenti Specifici:** *OCS Inventory* richiede l'installazione e la configurazione di specifici componenti server, come moduli *Perl* per *Apache*. Una configurazione non corretta di questi componenti può portare a malfunzionamenti o errori nell'applicazione
4. **Scalabilità:** Sebbene OCS Inventory sia adatto per piccole e medie imprese, può presentare difficoltà in ambienti IT di grandi dimensioni, la gestione di grandi volumi di dispositivi e informazioni può, infatti, rallentare la piattaforma, il che rende questa soluzione non adatta alle imprese, anche piccole e medie, che procedano ad uno sviluppo o espansione.

2. Progetto “Politech”: fase iniziale

2.1 Obiettivi

Alla luce delle ricerche preliminari i due obiettivi fondamentali individuati sono stati:

* **Semplicità d’uso e completezza**

Dal punto di vista **tecnico** l’obiettivo è quello di fornire una piattaforma per la gestione centralizzata degli *asset IT* che sia esaustiva nei contenuti ma allo stesso tempo intuitiva e *user-friendly,* che permetta di monitorare lo stato di dispositivi, accessori e licenze fornendo anche informazioni relative a sedi e dipendenti assegnatari.

* È stata quindi sviluppata un'applicazione modulare che integri database relazionali, interfacce minimaliste e strumenti di visualizzazione dei dati.
* **Estensione del target di riferimento**

Dal punto di vista della **fruibilità del servizio,** l’obiettivo è estendere il target delle piccole e medie imprese (PMI) attraverso una strategia di *business freemium*. Questo implica offrire un servizio di base gratuito, adeguato alle esigenze delle PMI, con la possibilità di un *upgrade* a pagamento per accedere a funzionalità avanzate e scalabili.

In questo modo, si supporta l’evoluzione dell’impresa, fidelizzandola e motivandola ad adottare un piano più ricco, in grado di soddisfare le sue necessità di espansione.

* È stata, quindi, prevista la possibilità di un abbonamento “*premium*” che consenta l’ampliamento delle funzioni disponibili.

2.2 Diagramma di Gantt: Fasi del progetto

Immagine che contiene testo, schermata, Carattere, cerchio

Descrizione generata automaticamente

2.4 Rischi e mitigazione

La gestione dei rischi è una componente cruciale per il successo di un progetto software. Riconoscere i potenziali problemi, valutarne la probabilità e adottare soluzioni preventive consente di minimizzare gli impatti negativi sullo sviluppo del prodotto.

È stato possibile individuare **cinque rischi probabili**, raggruppati per categoria, e sono state formulate le relative soluzioni.

1. **Cancellazione accidentale dei file**: (probabilità bassa): può essere risolto effettuando *backup* periodici.
2. **Problemi tecnici non previsti** (probabilità media): la soluzione migliore è pianificare cicli di test approfonditi nelle diverse fasi del progetto, includendo debugging sistematico e sessioni di revisione del codice.
3. **Mancanza di comunicazione tra i membri del team** (probabilità bassa): è possibile evitare questa problematica stabilendo processi chiari di comunicazione e collaborazione, anche attraverso l’uso di strumenti condivisi.
4. **Scelta inadeguata dei linguaggi di programmazione** (probabilità media): è sufficiente valutare e studiare approfonditamente dei linguaggi di programmazione selezionati ed effettuare un'analisi comparativa.
5. **Modifiche ai requisiti** (probabilità alta): i requisiti di un progetto software sono spesso soggetti a cambiamenti durante lo sviluppo, per poter evitare questo rischio è possibile adottare un approccio *Agile*, con iterazioni e revisioni frequenti.

2.5 Stima dei costi

Per stimare i costi e gli sforzi richiesti nello sviluppo di software è stato sfruttato il modello COCOMO (*COnstructive COst MOdel*).

Si tratta di uno dei modelli più utilizzati per la pianificazione dei progetti software e si basa su formule empiriche che considerano dimensioni del progetto, complessità e una serie di parametri che descrivono il contesto del progetto stesso.

La formula fondamentale del modello è:

**Effort=A⋅(size)^B⋅M**

* **A** è una costante che dipende dal tipo di progetto (valore tipico: 2.94 per progetti organici);
* **size** è la dimensione del progetto espressa in migliaia di linee di codice (KLOC);
* **B** è un esponente che tiene conto della complessità crescente dei progetti più grandi (con valori che variano tra 1.1 e 1.24);
* **M** è un moltiplicatore che dipende da diversi fattori relativi al prodotto, al processo e al personale:

PERS: capacità del personale → 1

RCPX: complessità del prodotto → 1.3

RUSE: livello di riuso → 1

PDIF: difficoltà legate alla piattaforma utilizzata → 1

PREX: esperienza del personale → 0.8

FCIL: funzionalità di supporto → 0.8

SCED: tempistica → 1

Per il progetto “*Politech*” i valori assegnati ai parametri sono:

* **size** = 7 KLOC (migliaia di linee di codice);
* **B** = 1.1 (adatto a progetti di media complessità);
* Moltiplicando il valore di ogni fattore ricaviamo **M**=0.83

Il risultato finale arrotondato è Effort≈21 Person-Months: risultato ideale in quando il tempo disponibile è nettamente inferiore.

3. Metodo proposto e requisiti

3.1 Modello di processo adottato

Il modello di processo adottato è ***Agile***, approccio particolarmente adatto per iterare rapidamente tra design, sviluppo e test e che consente lo sviluppo di funzionalità in modo incrementale, con continui miglioramenti in base al feedback ottenuto durante ogni fase.

3.2 Fasi principali del metodo proposto:

1. **Pianificazione Iniziale**:
   1. **Definizione dell'ambito**: Chiarire gli obiettivi del progetto e le funzionalità chiave dell'IT Asset Manager.
2. **Raccolta e Analisi dei Requisiti**:
   1. **Documentazione dei requisiti**: Redigere un documento dettagliato che elenchi i requisiti funzionali e non funzionali
3. **Progettazione dell'Architettura**:
   1. **Scelta del tech stack**: Selezionare le tecnologie più adatte, come MySQL per il database e Node.js per il backend.
   2. **Definizione dei moduli**: Stabilire i componenti principali del sistema e le loro interazioni.
4. **Sviluppo Incrementale**:
   1. **Iterazioni brevi**: Sviluppare funzionalità in cicli brevi, con rilascio di versioni incrementali.
5. **Test e Validazione**:
   1. **Test unitari e integrati**: Assicurare la qualità del codice e l'integrazione corretta dei componenti.

3.3 Requisiti

##### ***Requisiti Funzionali:***

* **Gestione degli Asset**:
  + Registrazione, aggiornamento e cancellazione di asset IT.
  + Tracciamento del ciclo di vita degli asset, inclusi acquisto, manutenzione e dismissione.
* **Inventario Centralizzato**:
  + Visualizzazione di un inventario dettagliato di tutte le risorse IT.
  + Ricerca e filtraggio avanzati per facilitare l'accesso alle informazioni.
* **Gestione delle Licenze Software**:
  + Monitoraggio delle licenze software per garantire la conformità.
  + Notifiche per le scadenze delle licenze e rinnovi.
* **Reportistica e Analisi**:
  + Generazione di report personalizzati sulle risorse IT.
  + Analisi dei dati per supportare decisioni strategiche.

##### ***Requisiti Non Funzionali:***

* **Scalabilità**: Il sistema deve poter gestire un numero crescente di asset senza degradare le prestazioni.
* **Usabilità**: Interfaccia utente intuitiva e accessibile, che richieda una formazione minima per gli utenti finali.
* **Sicurezza**:
  + Autenticazione e autorizzazione per garantire l'accesso controllato alle informazioni.
  + Protezione dei dati sensibili attraverso crittografia e altre misure di sicurezza.
* **Interoperabilità**: Capacità di integrarsi con altri sistemi aziendali esistenti, come piattaforme di gestione dei servizi IT.
* **Manutenibilità**: Architettura modulare che faciliti aggiornamenti e modifiche future.

4. Architettura e *Tech Stack*

4.1 Architettura

L’architettura è di tipo *Client-Server* con ***server*** gestito da Node.js e il ***database*** *MySQL* per memorizzare i dati.

Immagine che contiene testo, schermata, Blu elettrico, Elementi grafici

Descrizione generata automaticamente

4.2 Tech Stack

#### **1. Frontend**

Il *frontend* è la parte visibile e interattiva per l'utente.

* **Linguaggi**:
  + **HTML5**: Per strutturare i contenuti delle pagine.
  + **CSS3**: Per progettare layout e stili visivamente accattivanti.
  + **JavaScript**: Per funzionalità interattive e dinamiche.

#### **2. Backend**

Il *backend* è responsabile della logica di business e della gestione dei dati.

* **Linguaggio**:
  + ***Node.js***: Scelto per la sua velocità e per la capacità di gestire richieste asincrone, ideale per applicazioni moderne.
* ***Framework****:*
  + ***Express.js****:* *Framework* leggero *per Node.js*, utile per costruire API REST in modo rapido ed efficiente.
* **Autenticazione e Sicurezza**:
  + ***JSON Web Token (JWT)****:* Per gestire autenticazione e autorizzazione sicure.

#### **3. Database**

Il database memorizza tutti i dati relativi agli asset, alle licenze, ai dipendenti e alle sedi.

* **Tipo di database**:
  + ***MySQL****:* Database relazionale, scelto per la sua robustezza e flessibilità.
* ***Tool* di gestione**:
  + ***DBeaver****:* Per amministrare e progettare il database.

#### **4. Altri Strumenti**

* **Controllo Versione**:
  + ***Git***: Per gestire le versioni del codice.
  + ***GitHub*** *o* ***GitLab***: Per il repository del progetto.
* ***Workflow* e Collaborazione**:
  + ***Notion***: Per organizzare le attività.

5. Prototipo

Di seguito l’analisi del codice con le relative funzionalità principali

<https://www.notion.so/Documentazione-16bc6bbff61f80c185f6dad05b422cc9?pvs=4>

6. Validazione e verifica

Durante lo sviluppo dell’applicativo sono stati utilizzati vari strumenti per test di integrazione come *Postman* per testare le API RESTful o lo strumento *console* dei browser per verificare i *log* generati e conseguentemente correggere errori.

7. Discussione

La progettazione del sistema ha seguito un approccio incrementale, concentrandosi inizialmente sulle funzionalità chiave quali la gestione degli asset e delle licenze, per poi espandersi verso funzionalità secondarie, come la gestione delle sedi e l’integrazione di dashboard analitici.

**Architettura scalabile**:

La separazione tra *frontend* e *backend* ha dimostrato di essere una decisione efficace, permettendo una facile estendibilità del sistema. La struttura modulare del codice consente l’aggiunta di nuove funzionalità senza modificare significativamente le componenti esistenti.

**Punti di forza**:

* Navigazione intuitiva grazie al menu laterale.
* Design responsivo che si adatta bene a diversi dispositivi.

**Punti di debolezza**:

* + Mancanza di tooltip o messaggi di aiuto per i nuovi utenti.
  + La funzionalità di filtro degli asset potrebbe essere più avanzata (ad esempio, supportando più filtri simultanei).

8. Conclusioni e sviluppi futuri

1. **Ottimizzazioni di performance**:

* + Implementare un sistema di cache per velocizzare le query SQL.
  + Migliorare il tempo di risposta delle API tramite compressione e ottimizzazione dei payload.

2. **Estensione delle funzionalità**:

* + Integrare una dashboard analitica avanzata per visualizzare metriche chiave sugli asset aziendali.
  + Aggiungere un modulo per la gestione automatizzata delle licenze scadute.

3. **Preparazione alla produzione**:

* + Migrare il database su un ambiente cloud per garantire una maggiore affidabilità e scalabilità.

Sitografia e Fonti

[NIST Publications](https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.1800-5.pdf?utm_source=chatgpt.com)

[www.Deloitte.com](http://www.Deloitte.com)

[*www.Servicenow.com*](http://www.Servicenow.com)

[www.ocsinventory-ng.org](http://www.ocsinventory-ng.org)