

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**Developer Portal Hub: una soluzione
centralizzata per la gestione sicura ed
intuitiva della documentazione API**

Tesi di laurea

Relatore

Prof.ssa Ombretta Gaggi

Laureando

Andrea Meneghello 2009084

ANNO ACCADEMICO 2022-2023

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il seguente documento ha lo scopo di descrivere in modo dettagliato il lavoro svolto durante il periodo di stage, dal laureando Andrea Meneghello, della durata di trecentododici ore, presso l'azienda THRON S.p.A. L'obiettivo principale del progetto di stage è stato realizzare un portale per favorire la consultazione delle API pubbliche e private di THRON, attraverso una soluzione centralizzata.

Il portale è stato sviluppato utilizzando il framework Vue.js accompagnato da vari strumenti del suo ecosistema, ed è stato protetto con autenticazione seguendo lo standard OAuth2, integrandosi con il provider aziendale Azure. Oltre alla consultazione della documentazione, il portale deve permettere all'utente di provare le API direttamente dall'interfaccia in modo intuitivo, permettendo inoltre il download delle stesse in formato yaml.

Infine tutte le componenti implementate sono state opportunamente documentate e il loro corretto funzionamento è stato verificato tramite test di unità e di accettazione.

“Life is really simple, but we insist on making it complicated”

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Ombretta Gaggi, relatore della mia tesi, per l’aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto la mia famiglia per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Settembre 2023

Andrea Meneghello

Indice

1	Introduzione	1
1.1	L'azienda	1
1.2	Metodologie di sviluppo	2
1.3	Strumenti di sviluppo	2
1.4	Organizzazione del testo	2
2	Descrizione del progetto di stage	4
2.1	Introduzione al progetto	4
2.2	Obiettivi dello stage	4
2.3	Analisi preventiva dei rischi	4
3	Analisi dei requisiti	5
3.1	Descrizione dell'applicazione	5
3.2	Casi d'uso	5
3.3	Tracciamento dei requisiti	6
4	Progettazione e codifica	7
4.1	Tecnologie utilizzate	7
4.1.1	Frontend	7
4.1.1.1	Vue.js	7
4.1.1.2	Typescript	8
4.1.1.3	Vite.js	8
4.1.1.4	Sass	8
4.1.2	Backend	8
4.1.2.1	Nest.js	8
4.1.2.2	Express.js	9
4.1.3	Altre tecnologie di supporto	9
4.1.3.1	Node.js	9
4.1.3.2	Pinia	9
4.1.3.3	Vue router	9
4.1.4	Versionamento	9

4.1.4.1	Git	9
4.1.4.2	CodeCommit	9
4.1.5	Verifica	9
4.1.5.1	ESLint	9
4.1.5.2	Vitest	9
4.1.5.3	Jest	9
4.1.6	Librerie esterne utilizzate	9
4.1.6.1	THRON Components	9
4.1.6.2	THRON Helpers	9
4.1.6.3	Azure msal	9
4.1.6.4	Swagger ui	9
4.2	Struttura principale del sistema	9
4.2.1	Configurazione ambiente del progetto	10
4.2.2	Infrastruttura	10
4.3	Progettazione	10
4.3.1	Architettura front-end	10
4.3.1.1	Architettura Vue.js	10
4.3.2	Architettura back-end	11
4.3.2.1	Architettura Nest.js	11
4.4	Codifica	11
4.4.1	Codifica front-end	11
4.4.1.1	Utils	11
4.4.1.2	Components	11
4.4.1.3	Views	11
4.4.2	Codifica back-end	11
5	Attività di verifica e validazione	12
5.1	Test di unità	12
5.2	Collaudo	12
5.3	Documentazione	12
5.4	Migliorie future	12
5.5	Presentazione finale	12
6	Conclusioni	13
6.1	Obiettivi raggiunti e consuntivo finale	13
6.2	Conoscenze acquisite	13
6.3	Scenari di applicabilità e sviluppi futuri	13
6.4	Valutazione personale	13
A	Appendice A	14

<i>INDICE</i>	vii
Acronimi e abbreviazioni	15
Glossario	16
Bibliografia	17

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Il seguente capitolo vuole introdurre brevemente l'azienda e il relativo contesto aziendale.

1.1 L'azienda

THRON S.p.A. è un'azienda italiana con sede a Piazzola sul Brenta specializzata nello sviluppo di SaaS (Software as a Service) e opera nel settore dei DAM (Digital Asset Management), offrendo servizi di marketing e business intelligence.

Il suo prodotto principale è la "Thron DAM Platform", una piattaforma per la gestione dei contenuti digitali, nata con l'obiettivo di valorizzare e gestire le informazioni sui prodotti in modo separato dalla piattaforma di distribuzione finale.

La Thron DAM Platform è una soluzione completa per la gestione dei contenuti aziendali, inclusi documenti, video, immagini, audio e molto altro. L'obiettivo è garantire una gestione centralizzata dei contenuti e semplificarne l'adattamento e la distribuzione su diversi canali in modo efficiente. La piattaforma è supportata da un motore semantico che consente l'arricchimento dei contenuti con tag e metadati, facilitando l'organizzazione e la categorizzazione dei materiali.

THRON S.p.A. ha strutturato l'area RD in due principali team: il team Contenuti, focalizzato sulle tematiche legate al DAM e alle sue funzionalità, e il team Prodotto, responsabile della gestione del PIM (Product Information Management) e delle funzionalità legate alle tag sui prodotti. Il metodo di lavoro adottato è il framework agile SCRUM, che coinvolge attivamente gli stakeholder nel processo decisionale, incoraggiando suggerimenti e miglioramenti.

L'azienda mira a una vasta ed eterogenea clientela, per cui offre un prodotto flessibile e adattabile alle specifiche esigenze dei clienti. Inizialmente, la piattaforma viene venduta con un modulo base che offre le funzionalità standard, ma attraverso un marketplace, è possibile acquistare o aggiungere moduli gratuiti che ampliano e migliorano la piatta-

forma.

La cultura di prossimità con gli stakeholder e il contatto costante con diverse esigenze dei clienti portano THRON a un continuo processo di innovazione per soddisfare le richieste e rimanere all'avanguardia nel settore.

Attualmente, THRON conta circa cinquanta dipendenti, suddivisi in team in base alle rispettive competenze. La collaborazione tra i dipartimenti e la capacità di adattarsi alle esigenze del mercato e dei clienti sono elementi fondamentali nel successo dell'azienda. Un'ulteriore peculiarità dell'azienda è la possibilità di offrire soluzioni personalizzate per specifici casi d'uso dei clienti, garantendo un servizio su misura per le loro esigenze.

La continua evoluzione e innovazione della Thron DAM Platform, insieme all'impegno costante nel fornire soluzioni di alto livello, consolidano la posizione di THRON S.p.A. nel mercato del Digital Asset Management, portando il "made in Italy" in un ambito altamente competitivo e in continua crescita.

1.2 Metodologie di sviluppo

Introduzione all'idea dello stage.

1.3 Strumenti di sviluppo

1.4 Organizzazione del testo

Il **secondo capitolo** describe

Il **terzo capitolo** approfondisce ...

Il **quarto capitolo** approfondisce ...

Il **quinto capitolo** approfondisce ...

Il **sesto capitolo** approfondisce ...

Nel **settimo capitolo** describe ...

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*^[§];

- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Descrizione del progetto di stage

Il seguente capitolo vuole introdurre brevemente il progetto affrontato durante lo stage

2.1 Introduzione al progetto

2.2 Obiettivi dello stage

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

1. Performance del simulatore hardware

Descrizione: le performance del simulatore hardware e la comunicazione con questo potrebbero risultare lenti o non abbastanza buoni da causare il fallimento dei test.

Soluzione: coinvolgimento del responsabile a capo del progetto relativo il simulatore hardware.

2.3 Analisi preventiva dei rischi

Capitolo 3

Analisi dei requisiti

In questo capitolo viene esposta l'analisi dei requisiti effettuata durante lo stage, dove si vanno ad illustrare le funzionalità tramite casi d'uso e requisiti identificati, con l'obiettivo di creare un'immagine più chiara e definita del sistema.

3.1 Descrizione dell'applicazione

3.2 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo [Unified Modeling Language \(UML\)](#) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

UC1: Login

Attori Principali: Utente non autenticato.

Precondizioni: .

- L'utente possiede un account valido per autenticazione tramite microsoft 365;
- L'utente è all'interno del gruppo autorizzato per il login al portale.

Descrizione: L'utente vuole accedere al portale e deve inserire le proprie credenziali

per accedervi.

Postcondizioni: L'utente è autenticato correttamente e può procedere con l'utilizzo di tutte le funzionalità disponibili all'interno del portale.

- L'utente inserisce le proprie credenziali di accesso.

3.3 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.

Capitolo 4

Progettazione e codifica

In questo capitolo saranno descritte le attività di progettazione e codifica dell'applicativo. Inoltre verranno descritte le tecnologie utilizzate durante lo sviluppo del progetto, le scelte architetturali e i componenti sviluppati.

4.1 Tecnologie utilizzate

Di seguito viene data una panoramica delle tecnologie utilizzate durante lo sviluppo del progetto di stage.

4.1.1 Frontend

4.1.1.1 Vue.js

Vue.js è un framework JavaScript progressivo e reattivo, utilizzato per lo sviluppo di interfacce utente dinamiche e moderne. Creato da Evan You, Vue.js è apprezzato per la sua semplicità d'uso e flessibilità. Con un sistema di reattività basato su un modello di oggetti e dipendenze, Vue.js rende facile il monitoraggio e l'aggiornamento automatico dell'interfaccia utente in base ai cambiamenti di stato dei dati. La sua architettura basata su componenti consente di organizzare il codice in moduli riutilizzabili e autonomi, semplificando la creazione di applicazioni complesse. Grazie alle direttive, è possibile arricchire il DOM con funzionalità reattive, mentre il sistema di routing agevola la creazione di single page application. Con una crescita costante della comunità di sviluppatori, Vue.js è diventato un'opzione popolare nel mondo dello sviluppo frontend. Per il mio progetto sono andato ad utilizzare la versione 3 di Vue.js, insieme allo script setup, che è una nuova sintassi per definire componenti progettata per semplificare la struttura del codice e migliorare la leggibilità.

4.1.1.2 Typescript

TypeScript è un linguaggio di programmazione open-source sviluppato da Microsoft. Si basa su JavaScript e offre tipizzazione statica opzionale, consentendo agli sviluppatori di specificare tipi per variabili, parametri di funzioni e oggetti. Questa caratteristica aiuta a individuare errori e a migliorare la manutenibilità del codice. TypeScript viene poi compilato in JavaScript, rendendolo compatibile con tutti i browser e le piattaforme. Con una sintassi familiare per i programmatori JavaScript, TypeScript è adottato sempre di più nella community sia per lo sviluppo frontend che backend. All'interno del mio progetto sono andato a creare per la parte frontend una cartella chiamata `types` che contiene un file `typescript` contenente tutti i tipi utilizzati all'interno del progetto.

4.1.1.3 Vite.js

Vite.js è un build tool utilizzato per lo sviluppo di applicazioni web. È stato creato da Evan You, lo stesso creatore di Vue.js, e si basa su rollup.js. Vite.js è stato progettato per essere veloce, semplice da utilizzare e facile da configurare. La sua velocità è dovuta al fatto che utilizza la tecnica dell'ESM (ECMAScript Modules) che permette di caricare i moduli in modo asincrono, riducendo i tempi di compilazione e di hot-reload.

4.1.1.4 Sass

Sass è un'estensione di CSS che offre funzionalità aggiuntive e avanzate per semplificare e organizzare il modo in cui viene scritto e gestito il codice CSS. Può essere considerato un preprocessore CSS, in quanto viene compilato in CSS prima di essere interpretato dal browser. Sass inoltre permette di utilizzare funzionalità non disponibili in CSS nativo, offrendo una serie di funzioni, variabili, mixin e altro. Il codice in Sass utilizza una sintassi più leggibile e mantenibile rispetto a quella di CSS, e successivamente il preprocessore si occupa di tradurlo in CSS valido. Insieme a sass, ho utilizzato bem, ovvero una metodologia di naming convention utilizzata nel mondo dello sviluppo web.

4.1.2 Backend

4.1.2.1 Nest.js

Nest.js è un framework per applicazioni server-side basato su Node.js. Si basa su Express.js e TypeScript ed è progettato per creare applicazioni scalabili e performanti. Il framework NestJS combina concetti e caratteristiche provenienti da diversi paradigmi di sviluppo, tra cui la programmazione orientata agli oggetti (OOP), la programmazione funzionale e la programmazione reattiva.

4.1.2.2 Express.js

4.1.3 Altre tecnologie di supporto

4.1.3.1 Node.js

4.1.3.2 Pinia

4.1.3.3 Vue router

4.1.4 Versionamento

4.1.4.1 Git

4.1.4.2 CodeCommit

4.1.5 Verifica

4.1.5.1 ESLint

4.1.5.2 Vitest

4.1.5.3 Jest

4.1.6 Librerie esterne utilizzate

4.1.6.1 THRON Components

4.1.6.2 THRON Helpers

4.1.6.3 Azure msal

Azure msal è una libreria che permette di integrare il login con Azure Active Directory all'interno di un'applicazione web.

4.1.6.4 Swagger ui

Swagger ui

4.2 Struttura principale del sistema

Il sistema è composto da due principali sezioni:

- Front-end, ovvero l'interfaccia utente dell'applicazione web che permette all'utente di interagire con il sistema. È responsabile di presentare i contenuti in modo visivamente attraente e interattivo, consentendo agli utenti di navigare, inserire dati e svolgere azioni specifiche. Per lo sviluppo di questa parte del sistema è stato utilizzato il framework Vue.js;

- Back-end, ovvero la parte del sistema che elabora le richieste provenienti dal front-end e restituisce i risultati. È responsabile della gestione dei dati e della logica di business. Lo sviluppo di questa parte del sistema è stato realizzato utilizzando il framework Nest.js.

4.2.1 Configurazione ambiente del progetto

Microservice

4.2.2 Infrastruttura

Da capire

4.3 Progettazione

4.3.1 Architettura front-end

4.3.1.1 Architettura Vue.js

Vue.js è un framework utilizzato nelle single page application, che permette di definire le pagine web in modo modulare, utilizzando componenti riutilizzabili. I componenti costituiscono la base dell'architettura di Vue. Essi rappresentano una parte isolata dell'interfaccia, che può contenere il proprio modello, i propri stili e la propria logica, infatti ogni componente ha il proprio template scritto in HTML, il proprio script scritto nel mio caso in TypeScript e i propri stili scritti nel mio caso in scss. Come già accennato in precedenza, i componenti sono riutilizzabili all'interno di un'applicazione e possono essere composti tra loro per creare gerarchie di interfacce ancora più complesse.

L'architettura di Vue.js è basata sul pattern architetturale MVVM (Model-View-ViewModel), che è una variante del pattern MVC (Model-View-Controller), dove:

- Model: rappresenta lo stato, i dati e le regole di business dell'applicazione, che gestiscono l'accesso e la modifica di tali dati. Lo stato viene definito tramite l'uso di particolari variabili di tipo reattivo, che permettono di aggiornare automaticamente la view associata in caso di modifiche;
- View: è l'interfaccia utente, che visualizza i dati contenuti nel model e si occupa di reagire agli input dell'utente. La view è definita utilizzando i template Vue.js e viene reattivamente aggiornata in base ai cambiamenti del modello. La vista viene definita utilizzando un template, ovvero una dichiarativa dell'HTML, arricchita con alcune direttive Vue.js. Queste particolari direttive permettono di collegare elementi del DOM a proprietà o metodi del modello, in modo che la

view possa reagire agli input dell'utente e aggiornare automaticamente lo stato dell'applicazione.

- **ViewModel**: è l'intermediario tra la view e il model. Il ViewModel gestisce la logica dell'interfaccia utente e mantiene lo stato dell'applicazione sincronizzato con la view. Il ViewModel è rappresentato da un componente Vue.js, infatti esso è un'istanza che collega il modello e la vista. All'interno di un componente è possibile definire metodi, proprietà computate, metodi del ciclo di vita, gestione di eventi e molte altre funzionalità. Questo consente di definire la logica di presentazione e di manipolare i dati all'interno di un contesto definito.

In breve, l'architettura è incentrata sulla creazione e utilizzo di componenti riutilizzabili che al loro interno incorporano sia il modello che la vista. Un'aspetto che rende Vue.js diverso da altri framework è proprio il concetto di reattività, infatti Vue.js è in grado di rilevare automaticamente le dipendenze tra i componenti, in modo da poter aggiornare automaticamente.

4.3.2 Architettura back-end

4.3.2.1 Architettura Nest.js

Nest.js

4.4 Codifica

4.4.1 Codifica front-end

da scrivere

4.4.1.1 Utils

4.4.1.2 Store

4.4.1.3 Router

4.4.1.4 Components

4.4.1.5 Views

4.4.2 Codifica back-end

Capitolo 5

Attività di verifica e validazione

5.1 Test di unità

5.2 Collaudo

5.3 Documentazione

5.4 Migliorie future

5.5 Presentazione finale

Capitolo 6

Conclusioni

6.1 Obiettivi raggiunti e consuntivo finale

6.2 Conoscenze acquisite

6.3 Scenari di applicabilità e sviluppi futuri

6.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Acronimi e abbreviazioni

UML [Unified Modeling Language](#). [4](#), [12](#)

Glossario

UML in ingegneria del software *UML*, *Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di “lingua franca” nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. [11](#)

Bibliografia

Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Siti web consultati

Manifesto Agile. URL: <http://agilemanifesto.org/iso/it/>.