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, consectetuer 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 opportunatamente 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

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

code build e code commit

1.4 Organizzazione del testo

Il secondo capitolo descrive

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

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*[g];
- 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

La seguente sezione illustra i casi d'uso individuati durante l'analisi dei requisiti.

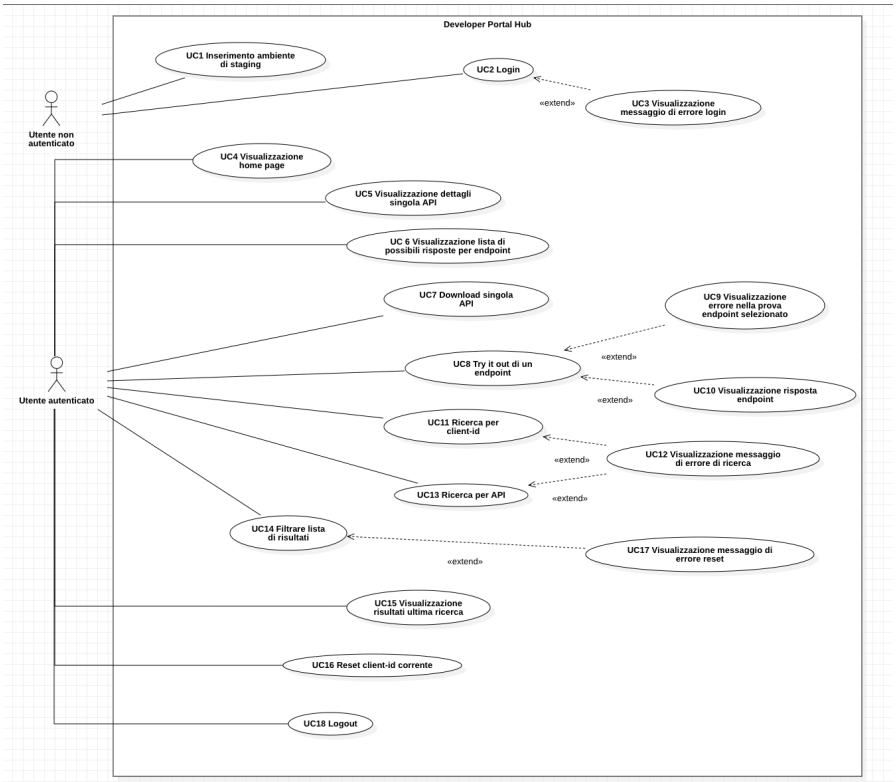

Figura 3.1: Scenario principale

3.2.1 Notazione utilizzata

3.2.2 Attori

UC

3.2.3 Descrizione del sistema

UC1: Inserimento ambiente di staging

Attori Principali: Utente non autenticato.

Precondizioni: L'utente non è autenticato e non ha ancora avviato l'applicazione web.

Descrizione: L'utente vuole avviare l'applicazione web e deve scegliere in che ambiente di staging farlo.

Postcondizioni: L'utente avvia l'applicazione nell'ambiente di staging scelto.

Scenario Principale:

1. L'utente seleziona il link del portale a seconda dell'ambiente di staging che vuole avviare (development, quality e production)

UC2: Login

Attori Principali: Utente non autenticato.

Precondizioni: L'utente possiede un account valido per autenticazione tramite microsoft 365 che appartiene al gruppo autorizzato per il login al portale. Inoltre l'utente non è autenticato e si trova nella pagina di login.

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.

Scenario Principale:

1. L'utente inserisce la propria e-mail;
2. L'utente inserisce la propria password.

Estensioni:

1. Visualizzazione messaggio di errore login UC3.

Generalizzazioni:

1. Inserimento e-mail UC2.1;
2. Inserimento password UC2.2.

UC2.1: Inserimento e-mail

Attori Principali: Utente non autenticato.

Precondizioni: L'utente possiede un account valido per autenticazione tramite microsoft 365 che appartiene al gruppo autorizzato per il login al portale. Inoltre l'utente non è autenticato e si trova nella pagina di login.

Descrizione: L'utente deve inserire la propria e-mail per autenticarsi al portale.

Postcondizioni: L'utente ha inserito la propria e-mail, può quindi procedere a completare il processo di autenticazione.

Scenario Principale:

1. L'utente inserisce nell'apposito campo la propria e-mail.

UC2.2: Inserimento password

Attori Principali: Utente non autenticato.

Precondizioni: L'utente possiede un account valido per autenticazione tramite microsoft 365 che appartiene al gruppo autorizzato per il login al portale. Inoltre l'utente non è autenticato e si trova nella pagina di login.

Descrizione: Descrizione → L'utente deve inserire la propria password per autenticarsi al portale.

Postcondizioni: L'utente ha inserito la propria password e può concludere il processo di autenticazione.

Scenario Principale:

1. L'utente inserisce nell'apposito campo la propria password.

UC3: Visualizzazione messggio di errore login

Attori Principali: Utente non autenticato.

Precondizioni: L'utente ha inserito una tra le due credenziali email o password in modo errato.

Descrizione: L'utente deve inserire delle credenziali corrette per poter effettuare il login correttamente.

Postcondizioni: L'utente ha inserito una tra le due credenziali errate.

Scenario Principale:

1. L'utente visualizza un messaggio di errore che lo informa che una delle credenziali che ha inserito per autenticarsi al portale è sbagliata.

UC4: Visualizzazione home page

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato ed è stato reindirizzato alla pagina principale del portale.

Descrizione: L'utente visualizza la pagina principale del portale.

Postcondizioni: L'utente ha visualizzato la pagina principale del portale.

Scenario Principale:

1. L'utente visualizza la pagina principale del portale.

Generalizzazioni:

1. Visualizzazione lista APIs disponibili UC4.1;
2. Visualizzazione client-id di default UC4.2;
3. Visualizzazione dettagli utente autenticato UC4.3.

UC4.1: Visualizzazione lista APIs disponibili

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato ed è stato reindirizzato alla pagina principale del portale.

Descrizione: L'utente visualizza la lista di API disponibili per la consultazione all'interno del portale.

Postcondizioni: L'utente ha visualizzato la lista di API disponibili nel portale.

Scenario Principale:

1. L'utente visualizza la lista di API disponibili nel portale.

UC4.2: Visualizzazione client-id di default

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato ed è stato reindirizzato alla pagina principale del portale.

Descrizione: L'utente visualizza il client id di default impostato nell'ambiente corrente.

Postcondizioni: L'utente ha visualizzato il client id di default impostato nell'ambiente corrente.

Scenario Principale:

1. L'utente visualizza il client id di default impostato nell'ambiente corrente.

UC4.3: Visualizzazione dettagli utente autenticato

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC5: Visualizzazione dettaglio singola API

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e si trova nella pagina principale del portale.

Descrizione: L'utente vuole visualizzare la pagina di dettaglio di una singola API.

Postcondizioni: L'utente ha visualizzato la pagina di dettaglio di una singola API tra quelle presenti nel portale.

Scenario Principale:

1. L'utente clicca su una delle API presenti nella lista di API disponibili all'interno del portale.

2. L'utente visualizza la pagina di dettaglio della singola API selezionata.

Generalizzazioni:

1. Visualizzaazione lista endpoint disponibili UC5.1.

UC5.1: Visualizzazione lista endpoint disponibili

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta visualizzando la pagina di dettaglio di una singola API.

Descrizione: L'utente vuole visualizzare la lista completa di endpoint disponibili per l'API.

Postcondizioni: L'utente visualizza la lista completa di endpoint disponibili per l'API.

Scenario Principale:

1. L'utente visualizza la lista completa di endpoint disponibili per l'API che ha selezionato.

Generalizzazioni:

1. Visualizzazione dettaglio singolo endpoint UC5.1.1.

UC5.1.1: Visualizzazione dettaglio singolo endpoint

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, sta visualizzando la pagina di dettaglio di una singola API contenente la lista di endpoint disponibili.

Descrizione: L'utente vuole visualizzare la pagina di dettaglio di un singolo endpoint.

Postcondizioni: L'utente visualizza la pagina di dettaglio di un singolo endpoint.

Scenario Principale:

1. L'utente clicca sullo specifico endpoint che vuole visualizzare;
2. L'utente visualizza i dettagli disponibili per l'endpoint selezionato.

Estensioni:

1. Visualizzazione lista di possibili risposte per endpoint UC6.

UC6: Visualizzazione lista di possibili risposte per endpoint

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, sta visualizzando la sezione di dettaglio di un singolo endpoint.

Descrizione: L'utente vuole visualizzare la lista dei possibili risultati possibili per l'endpoint selezionato.

Postcondizioni: L'utente visualizza la lista delle possibili risposte disponibili per l'endpoint che ha selezionato.

Scenario Principale:

1. L'utente visualizza i dettagli riguardanti le possibili risposte che l'endpoint selezionato può ritornare.

UC7: Download singola API

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato ed ha selezionato una API dalla lista di API disponibili.

Descrizione: L'utente vuole poter scaricare in formato yaml un API dal portale.

Postcondizioni: L'utente ha scaricato in formato yaml un API dal portale.

Scenario Principale:

1. L'utente scarica il formato yaml di un API dal portale, tra quelle disponibili;
2. Una nuova pagina si apre e il download dell'API viene eseguito;
3. Il nome del file è già impostato con il nome dell'API scaricata.

UC8: Try it out endpoint

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, sta visualizzando i dettagli di un singolo endpoint di un API disponibile nel portale.

Descrizione: L'utente vuole poter provare l'endpoint selezionato.

Postcondizioni: L'utente ha provato l'endpoint selezionato.

Scenario Principale:

1. L'utente ha selezionato una determinata API
2. L'utente ha selezionato un determinato endpoint di quell'API
3. L'utente ha cliccato sul pulsante per provare l'endpoint selezionato

Estensioni:

1. Visualizzazione errore nella prova dell'endpoint selezionato UC9;
2. Visualizzazione risposta endpoint UC10;

Generalizzazioni:

1. Inserimento parametri per try it out endpoint UC8.1;
2. Definire campi aggiuntivi per try it out endpoint UC8.2.

UC8.1: Inserimento parametri per try it out endpoint

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, sta visualizzando la schermata di try it out nella sezione di inserimento dei parametri.

Descrizione: L'utente vuole poter inserire i parametri necessari per la prova dell'endpoint.

Postcondizioni: L'utente ha inserito i parametri necessari alla chiamata verso l'endpoint.

Scenario Principale:

1. L'utente è nella sezione di prova dell'endpoint che ha selezionato;
2. L'utente inserisce i parametri richiesti per la chiamata verso l'endpoint selezionato.

UC8.2: Definire campi aggiuntivi per try it out endpoint

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, sta visualizzando la schermata di try it out nella sezione dei parametri aggiuntivi.

Descrizione: L'utente vuole poter definire dei campi aggiuntivi ai parametri già esistenti, per poi andare a provare la chiamata verso l'endpoint.

Postcondizioni: L'utente ha creato dei parametri aggiuntivi per la chiamata verso l'endpoint.

Scenario Principale:

1. L'utente è nella sezione di aggiunta dei parametri aggiuntivi, nel try it out dell'endpoint selezionato;
2. L'utente definisce dei parametri da aggiungere a quelli già esistenti.

UC9: Visualizzazione errore nella prova dell'endpoint selezionato

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, è nella sezione di prova di un endpoint ed ha inserito dei parametri errati.

Descrizione: L'utente deve inserire dei parametri corretti per poter provare l'endpoint senza riscontrare errori.

Postcondizioni: L'utente ha inserito dei parametri parzialmente o in modo scorretto e non può procedere con l'esecuzione della chiamata.

Scenario Principale:

1. L'utente visualizza un messaggio di errore che lo avvisa che i parametri inseriti non sono corretti, o risultano incompleti.

UC10: Visualizzazione risposta endpoint

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, è nella sezione di prova di un endpoint ed ha inserito dei parametri corretti.

Descrizione: L'utente vule visualizzare la risposta dell'endpoint .

Postcondizioni: L'utente visualizza la risposta adeguata alla chiamata verso l'endpoint, in base ai parametri inseriti.

Scenario Principale:

1. L'utente visualizza la risposta dell'endpoint, uno tra quelle possibili contenute nella descrizione dell'endpoint. La risposta varia in base ai parametri inseriti.

UC11: Ricerca per client-id

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

Estensioni:

1. Visualizzazione messaggio di errore di ricerca UC 12.

Generalizzazioni:

1. Inserimento client-id UC11.1.

UC11.1: Inserimento client-id

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC12: Visualizzazione messaggio di errore di ricerca

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC13: Ricerca per API

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

Estensioni:

1. Visualizzazione messaggio di errore di ricerca UC12.

Generalizzazioni:

1. Inserimento nome api UC13.1.

UC13.1: Inserimento nome api

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC14: Filtrare lista di risultati

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC15: Visualizzazione risultati ultima ricerca

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC16: Reset client-id corrente

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

Estensioni:

1. Visualizzazione messaggio di errore reset UC 17.

UC17: Visualizzazione messaggio di errore reset

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

UC18: Logout

Attori Principali: Utente autenticato.

Precondizioni: .

Descrizione: .

Postcondizioni: .

Scenario Principale:

1.

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 architettuturali 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

buildspec -> cartella con configuraazione dei build, per codebuild primo step, builda sia portal che il middleware quando finsice la build ti attacca allo step di deploy, definite in infra. PEr il portale costrutto cloudfrontdistribution -> genera il template di aws clodfront e esegue il deploy degli asset statici su s3. S3 container Secondo costrutto -> throndockerlambda e questo crea la lambda con quando viene invocata avvia un docker con all'interno il mio middleware.

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 interfaccie ancora più complesse.

L'architettura di Vue.js è basata sul pattern architettuturale 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

Auth

Debounce

EndpointApiCall

GetClients

GetResults

MsGraphApiCall

SwaggerUtils

4.4.1.2 Config

ConfigAuth

4.4.1.3 Store

Auth

Store

4.4.1.4 Router

CustomNavigation

4.4.1.5 Views

LoginView LoginView testo

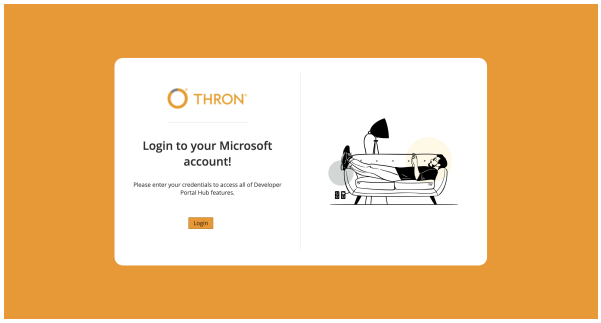

Figura 4.1: LoginView

HomeView HomeView testo

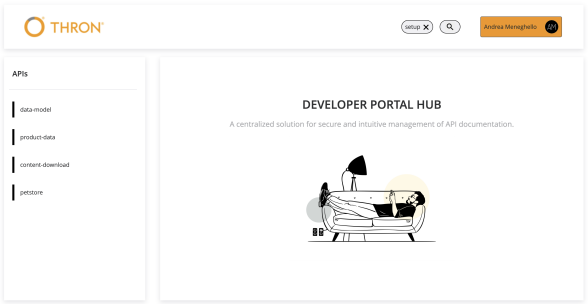

Figura 4.2: LoginView

NotFoundView NotFoundView testo

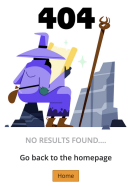

Figura 4.3: NotFoundView

4.4.1.6 Components

HeaderNav HeaderNav testo

MainContent Swagger testo

SideBar Sidebar testo

StartPage Start page testo

SwaggerLoader Swagger loader testo

Figura 4.4: SwaggerLoader

DownloadButton download button testo

LogoutButtom logout button testo

LoginButton login button testo

SearchButton SearchButton testo

SearchBar Search bar testo

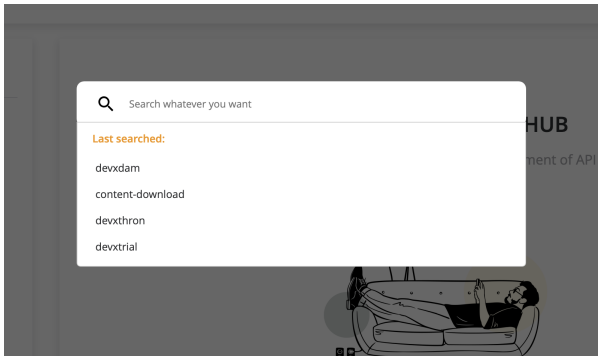

Figura 4.5: SearchBar

Autocomplete Autocomplete testo

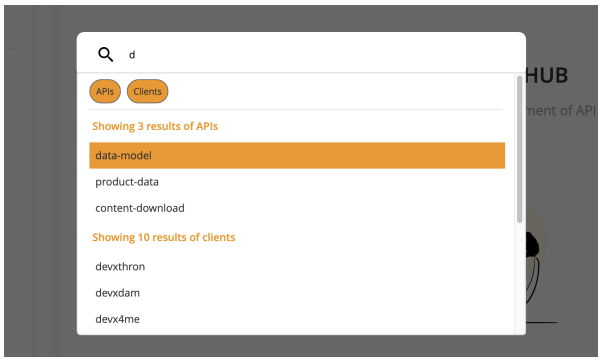

Figura 4.6: Autocomplete

Chip chip testo da scrivere

Figura 4.7: Chip

Filter filter testo

OptionList option list testo

OptionListItem OptionListItem testo

SnackBar snack bar testo da scrivere

Figura 4.8: SnackBar

Loader Loader testo da scrivere

Figura 4.9: Loader

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

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/.