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

Andrea Meneghello: *Developer Portal Hub: una soluzione centralizzata per la gestione sicura ed intuitiva della documentazione API,* Tesi di laurea, © Settembre 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 R&D 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 pi-

attaforma.
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

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

In questa sezione vengono elencati gli obiettivi prefissati da raggiungere durante lo stage, suddivisi in obbligatori, desiderabili e opzionali.

Si farà riferimento ai requisiti secondo le seguenti notazioni:

- **OB**: per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto;
- **DE**: per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- **OP**: per i requisiti opzionali, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, che identificano univocamente ogni requisito.

Obbligatori

- **OB1**: Realizzazione di un portale che consenta la consultazione degli OpenAPIs schemas dei servizi pubblici e privati offerti da THRON;
- **OB2**: Rendere possibile l'utilizzo delle API direttamente dal portale (con inserimento manuale del token di autenticazione);
- **OB3**: Documentazione delle funzionalità implementate;

- **OB4**: Realizzazione di test di unità delle funzionalità implementate.

Desiderabili

- **DE1**: Implementare la funzionalità di recupero automatico degli OpenAPI schemas;
- **DE2**: Implementare la funzionalità di autenticazione al portale.

Opzionali

- **OP1**: Implementare la funzionalità di download dello schema di uno specifico servizio (formato YAML);
- **OP2**: Implementare la funzionalità di recupero automatico del token di autenticazione per l'utilizzo della API direttamente dal portale.

2.3 Analisi preventiva dei rischi

Durante la fase iniziale sono stati individuati dei possibili rischi a cui si poteva andare incontro. Per contrastare ciò, si è cercato di porre rimedio con delle contromisure appropriate.

1. **Stack tecnologico**: per ovviare a questo rischio, è stato previsto un periodo di formazione iniziale, durante il quale è stato possibile studiare le tecnologie da utilizzare e sperimentarle in piccoli progetti di prova;

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

Il progetto consiste nel creare un portale che permetta la consultazione di tutte le API Thron con la possibilità di provarle in modo semplice e veloce, direttamente dal portale. Il prodotto verrà utilizzato internamente all'azienda, più precisamente all'interno della Product Area, con lo scopo di facilitare attività di sviluppo, di testing e di supporto durante le attività giornaliere aziendali.
Il portale è disponibile in tre ambienti di staging: development, quality e production, ognuno dei quali è identificato da un link diverso, che servirà per accedere al portale in quel determinato ambiente. La differenza principale tra ogni suddivisione è che i client sono diversi per ogni ambiente, ciò evita che un utente possa utilizzare un client di produzione involontariamente, che può causare problemi di sicurezza. Inoltre è utile per evitare prove indesiderate, dato che per esempio una chiamata delete su un client di produzione cancellerebbe effettivamente un dato.
Ogni API è formata da più endpoint al suo interno, che possono essere provati singolarmente. Ogni endpoint ha un elenco di possibili risposte che può ritornare, che dipende dai parametri inseriti nella chiamata. Alcune chiamate come le POST o DELETE avranno dei parametri obbligatori da inserire, mentre altre chiamate come le GET, avranno dei parametri opzionali.
In caso l'utente voglia provare le diverse chiamate al di fuori del portale, è possibile scaricare le API in formato YAML, così da poter usare strumenti di terze parti.

3.2 Casi d'uso

Questa sezione illustra i casi d'uso individuati nel corso dell'analisi dei requisiti del progetto che sono stati definiti utilizzando il linguaggio Unified Modeling Language (UML). Ogni caso d'uso offre una panoramica chiara dei diversi attori coinvolti e delle intereazioni che essi intraprendono nel contesto del sistema.
Ogni caso d'uso è indentificato da un codice univoco, che segue la seguente notazione:

UC[Codice-padre].[Codice-figlio]

3.2.1 Attori

Gli attori individuati nel sistema sono i seguenti:

- **Utente non autenticato**: è un utente che non è autenticato nel sistema e non può accedere alle funzionalità del portale;
- **Utente autenticato**: è un utente che è autenticato nel sistema e che può accedere alle funzionalità del portale;
- **Microsoft 365**: è un servizio di autenticazione di Microsoft che permette di autenticarsi tramite account aziendale.

3.2.2 Descrizione del sistema

Di seguito viene illustrato un diagramma riassuntivo che mostra i casi d'uso individuati nel sistema e le relazioni tra di essi.

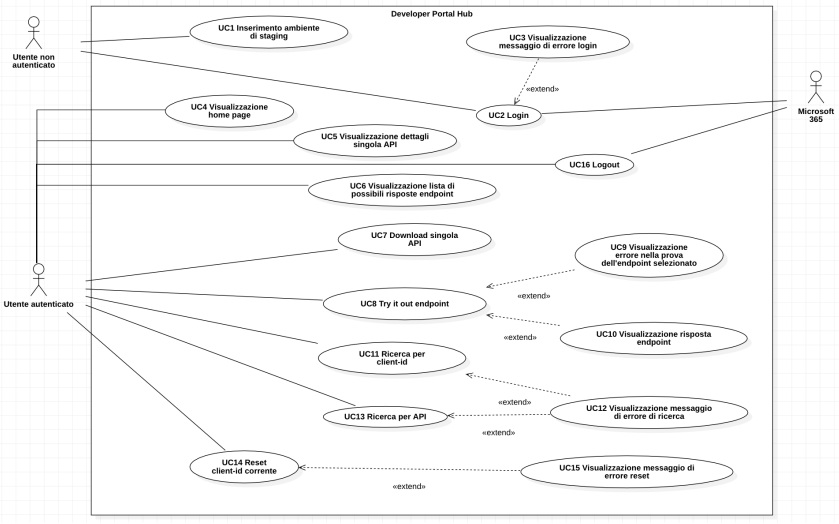

Figura 3.1: Scenario principale

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, Microsoft 365.

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

Descrizione: L'utente vuole accedere al sistema 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 sistema.

Scenario Principale:

1. L'utente inserisce la propria e-mail;
2. L'utente inserisce la propria password;
3. Microsoft 365 verifica le credenziali inserite, in base ai permessi configurati.

Estensioni:

1. Visualizzazione messaggio di errore login UC3.

Generalizzazioni:

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

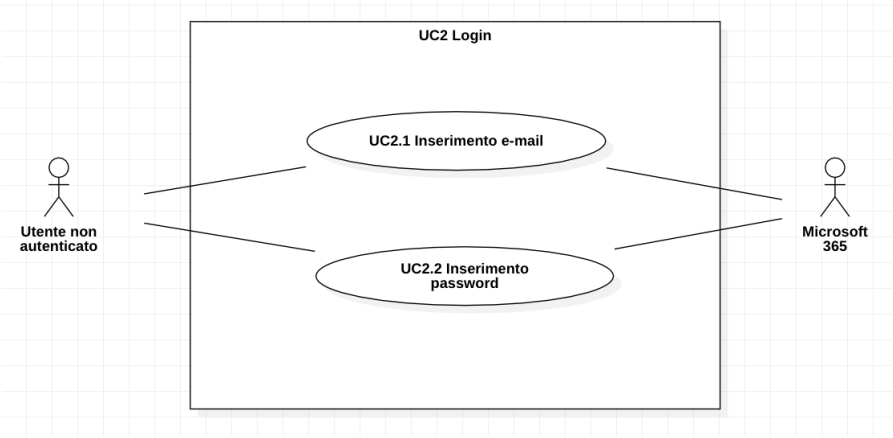

Figura 3.2: UC2 Login

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 sistema. 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 sistema.

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 sistema. Inoltre l'utente non è autenticato e si trova nella pagina di login.

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

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 sistema è sbagliata.

UC4: Visualizzazione home page

Attori Principali: Utente autenticato.

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

Descrizione: L'utente vuole visualizzare la pagina principale.

Postcondizioni: L'utente ha visualizzato la pagina principale.

Scenario Principale:

1. L'utente visualizza la pagina principale.

Generalizzazioni:

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

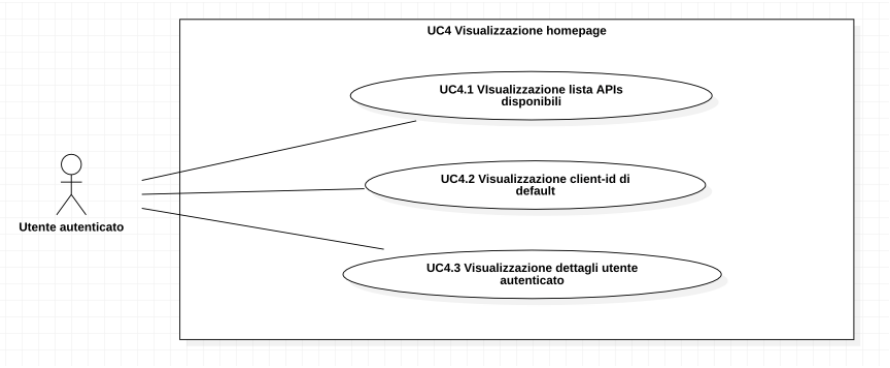

Figura 3.3: UC4 Visualizzazione home page

UC4.1: Visualizzazione lista APIs disponibili

Attori Principali: Utente autenticato.

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

Descrizione: L'utente vuole visualizzare la lista di API disponibili per la consultazione all'interno del sistema.

Postcondizioni: L'utente ha visualizzato la lista di API disponibili all'interno del sistema.

Scenario Principale:

1. L'utente visualizza la lista di API disponibili all'interno del sistema.

UC4.2: Visualizzazione client-id di default

Attori Principali: Utente autenticato.

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

Descrizione: L'utente vuole visualizzare il client-id di default impostato nell'ambiente corrente.

Postcondizioni: L'utente ha visualizzato il client-id di default impostato nell'ambiente corrente in cui si trova.

Scenario Principale:

1. L'utente visualizza il client-id di default impostato nell'ambiente corrente in cui si trova.

UC4.3: Visualizzazione dettagli utente autenticato

Attori Principali: Utente autenticato.

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

Descrizione: L'utente vuole visualizzare i propri dati personali, ovvero del proprio utente autenticato.

Postcondizioni: L'utente visualizza i dati personali del proprio account autenticato nel sistema.

Scenario Principale:

1. L'utente visualizza le informazioni personali del proprio account autenticato nel sistema.

UC5: Visualizzazione dettagli singola API

Attori Principali: Utente autenticato.

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

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 sistema.

Scenario Principale:

1. L'utente clicca su una delle API presenti nella lista di API disponibili all'interno del sistema.
2. L'utente visualizza la pagina di dettaglio della singola API selezionata.

Generalizzazioni:

1. Visualizzazione lista endpoint disponibili UC5.1.

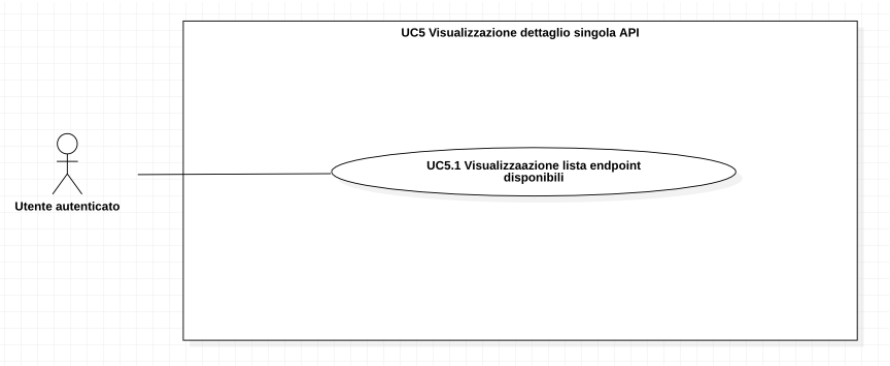

Figura 3.4: UC5 Visualizzazione dettaglio singola API

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.

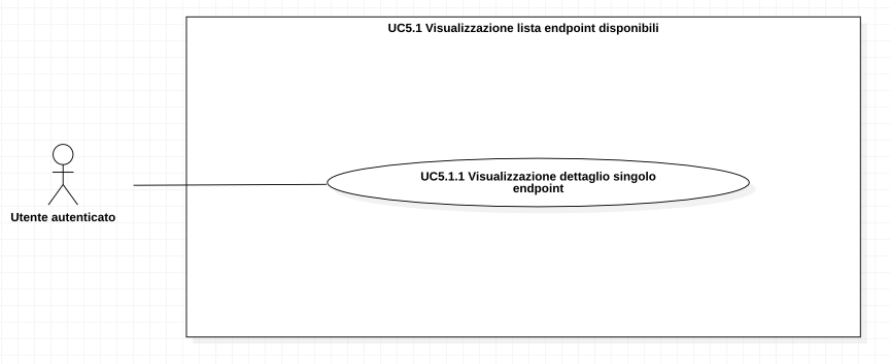

Figura 3.5: UC5.1 Visualizzazione lista endpoint disponibili

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 endpoint UC6.

UC6: Visualizzazione lista di possibili risposte 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 nel sistema.

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 sistema.

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.

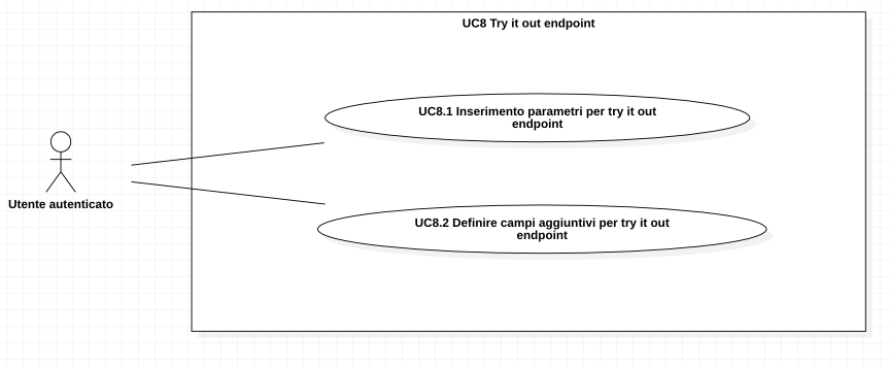

Figura 3.6: UC8 Try it out endpoint

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 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 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 vuole 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: L'utente è autenticato e sta navigando all'interno del sistema.

Descrizione: L'utente vuole poter ricercare un client-id all'interno del sistema.

Postcondizioni: L'utente effettua una ricerca per client-id e il sistema ha restituito i risultati della ricerca.

Scenario Principale:

1. L'utente clicca il bottone per effettuare la ricerca per client-id;
2. L'utente inserisce nell'apposito campo il client-id che vuole cercare all'interno del sistema;
3. Il sistema ricerca il client-id inserito e restituisce i risultati della ricerca;
4. Il sistema aggiunge l'ultima ricerca effettuata alla cronologia delle ultime ricerche.

Estensioni:

1. Visualizzazione messaggio di errore di ricerca UC 12.

UC12: Visualizzazione messaggio di errore di ricerca

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e ha cercato un client-id o un API inesistente nel sistema.

Descrizione: L'utente deve inserire un client-id o un API esistente nel sistema per poter effettuare la ricerca. Inoltre il client-id deve esistere per l'ambiente selezionato.

Postcondizioni: L'utente ha inserito un client-id o un API inesistente nel sistema e visualizza un messaggio di errore.

Scenario Principale:

1. L'utente visualizza un messaggio di errore che lo informa che la ricerca effettuata non ha prodotto nessun risultato, indicando che ciò è dovuto al fatto che il client-id o l'API cercata non è presente nel sistema, oppure non esiste quel client-id per l'ambiente selezionato.

UC13: Ricerca per API

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e sta navigando all'interno del sistema.

Descrizione: L'utente vuole poter ricercare un API all'interno del sistema.

Postcondizioni: L'utente effettua una ricerca per API e il sistema ha restituito i risultati della ricerca.

Scenario Principale:

1. L'utente clicca il bottone per effettuare la ricerca per API;
2. L'utente inserisce nell'apposito campo l'API che vuole cercare all'interno del sistema;
3. Il sistema ricerca l'API inserita e restituisce i risultati della ricerca;
4. Il sistema aggiunge l'ultima ricerca effettuata alla cronologia delle ultime ricerche.

Estensioni:

1. Visualizzazione messaggio di errore di ricerca UC12.

UC14: Reset client-id corrente

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato, sta navigando all'interno del sistema e ha già selezionato un client-id che non è quello di default.

Descrizione: L'utente vuole poter resettare il client-id corrente e tornare al client-id di default per l'ambiente selezionato.

Postcondizioni: L'utente ha resettato il client-id corrente e ora visualizza il client-id di default per l'ambiente selezionato.

Scenario Principale:

1. L'utente clicca il bottone per resettare il client-id corrente;
2. Il sistema resetta il client-id corrente ed imposta il client-id al valore di default a seconda dell'ambiente selezionato.

Estensioni:

1. Visualizzazione messaggio di errore reset UC15.

UC15: Visualizzazione messaggio di errore reset

Attori Principali: Utente autenticato.

Precondizioni: L'utente è autenticato e ha provato a resettare il client-id corrente che è quello di default per l'ambiente selezionato.

Descrizione: L'utente deve aver selezionato un client-id diverso da quello di default, per poterlo resettare.

Postcondizioni: L'utente ha provato a resettare il client-id corrente, ma risulta essere quello di default.

Scenario Principale:

1. L'utente visualizza un messaggio di errore che lo informa che il client-id corrente è quello di default per l'ambiente selezionato e non può essere resettato.

UC16: Logout

Attori Principali: Utente autenticato, Microsoft 365.

Precondizioni: L'utente è autenticato e vuole uscire dalla sessione corrente.

Descrizione: L'utente vuole effettuare il logout dal sistema.

Postcondizioni: L'utente effettua il logout dal sistema terminando la sessione corrente, non è più autenticato e viene reindirizzato alla pagina di login.

Scenario Principale:

1. L'utente clicca il bottone per effettuare il logout;
2. Microsoft 365 effettua il logout dell'utente terminando la sessione.

3.3 Tracciamento dei requisiti

In questa sezione vengono riportati i requisiti indiduati durante il progetto di stage. Questo capitolo si sofferma in particolare sulla classificazione dei requisiti in tre categorie principali:

- **Requisiti funzionali**: delineano le funzionalità che il sistema deve offrire. Essi delineano le azioni specifiche che il sistema deve eseguire, le risposte attese a determinati input e le dinamiche generali delle operazioni;
- **Requisiti qualitativi**: definiscono gli aspetti legati alla qualità, all'usabilità e alle prestazioni del sistema;
- **Requisiti di vincolo**: delineano le restrizioni e i parametri che il sistema deve rispettare durante lo sviluppo e l'implementazione.

Inoltre viene fatta una classificazione dei requisiti in base alla priorità assegnata a ciascun requisito.

3.3.0.1 Notazione

Ciascun requisito è identificato da un codice univoco, che segue la seguente notazione:

R[Priorità][Tipo]-[Codice]

dove:

- **Priorità** indica il livello di priorità assegnato: obbligatorio (**O**), desiderabile (**D**) e opzionale (**Z**);
- **Tipo** indica il tipo di requisito: funzionale (**F**) , qualititativo (**Q**) e di vincolo (**V**);
- **Codice** indica il codice identificativo del requisito.

Nelle tabelle 3.1, 3.3 e 3.5 sono riassunti i requisiti tramite una breve descrizione accompagnata dalle fonti da cui è stato individuato il requisito per facilitarne la tracciabilità.

3.3.0.2 Requisiti funzionali

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Fonte
RFO-1	L'utente deve scegliere l'ambiente di staging	UC1

RFO-2	Il sistema deve permettere l'autenticazione ad un utente con un account valido	UC2
RFO-2.1	L'utente deve inserire la propria mail	UC2.1
RFO-2.2	L'utente deve inserire la propria password	UC2.2
RFO-3	Il sistema deve avvisare l'utente tramite un messaggio di errore che le credenziali inserite nel login sono errate	UC3
RFO-4	Il sistema deve reindirizzare l'utente alla pagina principale, dopo che il login è andato a buon fine	UC4
RFO-4.1	L'utente deve poter visualizzare la lista di API disponibili nel sistema	UC4.1
RFO-4.2	L'utente deve poter visualizzare la lista di client-id di default impostata nell'ambiente di staging in cui si trova	UC4.2
RFO-4.3	L'utente deve poter visualizzare i dettagli relativi al suo account	UC4.3
RFO-5	L'utente deve poter visualizzare i dettagli relativi ad una singola API	UC5
RFO-5.1	L'utente deve poter visualizzare la lista di endpoint disponibili all'interno del sistema	UC5.1
RFO-5.1.1	L'utente deve poter visualizzare i dettagli relativi ad un singolo endpoint di una determinata API	UC5.1.1
RFO-6	L'utente deve poter visualizzare l'elenco delle possibili risposte per il determinato endpoint selezionato	UC6
RFO-7	Il sistema deve permettere il download all'utente di una singola API	UC7
RFO-8	Il sistema deve permettere il try it out di un singolo endpoint all'utente	UC8
RFO-8.1	Il sistema deve permettere l'inserimento dei parametri necessari per il try it out di un endpoint	UC8.1
RFD-8.2	Il sistema deve permettere la possibilità di definire dei campi aggiuntivi per il try it out di un endpoint	UC8.2
RFO-9	Il sistema deve avvisare l'utente tramite un messaggio di errore che i parametri inseriti non sono corretti o incompleti	UC9
RFO-10	L'utente deve poter visualizzare la risposta dell'endpoint che ha provato	UC10

RFO-11	Il sistema deve permettere all'utente di poter effettuare una ricerca per client-id	UC11
RFO-12	Il sistema deve avvisare l'utente tramite un messaggio di errore che la ricerca effettuata non ha portato a risultati presenti nel sistema	UC12
RFO-13	Il sistema deve permettere all'utente di poter effettuare una ricerca per API	UC13
RFD-14	Il sistema deve permettere il reset del client-id corrente	UC14
RFO-15	Il sistema deve avvisare l'utente tramite un messaggio di errore che il client-id è già di default	UC15
RFO-16	Il sistema deve permettere il logout all'utente, uscendo dalla sessione	UC16

3.3.0.3 Requisiti qualitativi

Tabella 3.3: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Fonte
RQO-1	Il progetto deve essere accompagnato da documentazione tecnica e funzionale	Interno
RQO-2	La parte frontend del progetto deve essere coperta da test di unità	Interno
RQZ-3	L'applicazione web deve avere un'interfaccia responsive	Interno
RQD-5	L'applicativo deve essere accessibile utilizzando i principali browser	Interno

3.3.0.4 Requisiti di vincolo

Tabella 3.5: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Fonte
RVO-1	L'applicazione deve essere sviluppata utilizzando il framework Vue.js 3, usando Typescript come linguaggio di programmazione	UC1

RVO-2	I componenti devono essere scritti utilizzando le Composition API di Vue.js 3	UC1
RVD-3	I componenti di base devono essere implementati utilizzando la libreria THRON Components	UC2
RVZ-4	L'interfaccia dell'applicazione deve seguire il design system THRON	UC2

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.
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.
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.

4.1.3 Altre tecnologie di supporto

4.1.3.1 Node.js

Node.jsè un ambiente di runtime JavaScript open-source progettato per eseguire codice lato server. Per gestire le dipendenze del mio progetto, ho deciso di utilizzare pnpm

come gestore di pacchetti. Questa selezione ha portato a un miglior utilizzo delle risorse di sistema e ha notevolmente accelerato il processo di installazione delle dipendenze.

4.1.3.2 Pinia

Pinia è una libreria per la gestione dello stato per applicazioni Vue.js. Promuove l'uso di store modulari, ognuno dei quali gestisce uno stato specifico dell'applicazione.

4.1.3.3 Vue router

Vue-router è una libreria per la gestione delle route per le applicazione Vue.js. Permette di definire le route dell'applicazione e di navigare tra le pagine.

4.1.4 Versionamento

4.1.4.1 Git

Git è un sistema di controllo versione distribuito e altamente flessibile, utile per tenere traccia delle modifiche apportate al codice sorgente durante lo sviluppo di un progetto software.

4.1.4.2 CodeCommit

AWS CodeCommit rappresenta un servizio di hosting di repository altamente scalabile, che è gestito all'interno dell'ecosistema di Amazon Web Services (AWS). Con CodeCommit, è possibile ospitare repository Git privati in un ambiente sicuro e flessibile.

4.1.5 Verifica

4.1.5.1 ESLint

lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

4.1.5.2 Vitest

lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

4.1.6 Librerie esterne utilizzate

4.1.6.1 THRON Components

4.1.6.2 Azure msal

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

4.1.6.3 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

Il progetto di stage, necessita di cartelle per la configurazione dell'ambiente di sviluppo, per la configurazione del progetto e per la configurazione del deploy. Più precisamente, il progetto segue la seguente struttura:

Buildspec

La cartella buildspec contiene i file di configurazione per la build dell'applicazione su AWS CodeBuild. Questo è il primo step del processo di deploy, in quanto viene eseguito la build sia del middleware che del portale. La cartella è formata a sua volta da tre file: buildspec_development, buildspec_quality e buildspec_production. Ognuno di questi file configura la build dell'applicazione in base all'ambiente di deploy.
Successivamente dopo aver effettuato tutti i comandi specificati nei file buildspec, inizia lo step di deploy, specificato nella cartella Infra. Ognuno dei file buildspec è un file YAML, che è formato dalle seguenti sezioni:

- La sezione env dove vengono specificate le variabili d'ambiente utilizzate nel progetto;
- La sezione di pre-build dove vengono specificati i comandi da eseguire prima della build;
- La sezione di build dove vengono specificati i comandi per la build del portale e del middleware.

Infra

La cartella infra contiene i file di configurazione per il deploy dell'applicazione su AWS. È scritta utilizzando il linguaggio di programmazione Python.
La cartella è formata a sua volta da due file: app.py e stack.py. Il primo file contiene la configurazione per il deploy dell'applicazione, mentre il secondo file contiene la configurazione per il deploy dell'infrastruttura.
Per quanto riguarda l'infrastruttura del progetto, sono andato ad utilizzare ed a configurare due costrutti, anche chiamati CDK Constructs o semplicemente Constructs, che sono delle classi che rappresentano un componente dell'infrastruttura. Il primo costrutto si chiama ThronCloudFrontDistribution e consente di accelerare la distribuzione dei contenuti web statici e dinamici, come file .html, .css, .js e immagini, agli utenti. In breve questo costrutto genera il template di AWS CloudFront ed esegui il deploy degli asset statici su un bucket S3.
Il secondo costrutto che sono andato ad utilizzare si chiama ThronDockerLambda e consente di creare una funzione lambda in cui il gestore è un'immagine docker. Nel caso del mio progetto, mi è servito per creare un lambda che quando viene invocata avvia un docker con all'interno il mio middleware.

Middleware

La cartella middleware contiene il progetto backend in nest-js. Nello specifico la cartella è formata dalle seguenti sezioni:

- La cartella src che contiene il codice sorgente dell'applicazione, dove vengono specificati i vari endpoint che utilizzo sulla parte frontend, lo script per il setting della lambda, una cartella con gli helpers e la cartella del middleware.
- La cartella test che contiene i test di unità dell'applicazione;
- File .env che contiene le variabili d'ambiente utilizzate nel progetto;
- Una cartella node_modules che contiene le dipendenze del progetto.

Portal

La cartella portal contiene il progetto frontend in vue-js. Nello specifico la cartella è formata dalle seguenti sezioni:

- La cartella public che contiene il file index.html, che è il file principale dell'applicazione;
- Una cartella node_modules che contiene le dipendenze del progetto;
- La cartella src che contiene il codice sorgente dell'applicazione, dove vengono specificati i vari componenti che utilizzo, i vari store, i vari router e i vari utils.
- File ėnv che contiene le variabili d'ambiente utilizzate nel progetto.

Dockerfile

Il file Dockerfile è un file docker che contiene le istruzioni per creare un'immagine che verrà utilizzata per eseguire il middleware all'interno di una lambda.
Utilizzando la struttura Dockerfile, sono andato a creare un ambiente isolato in cui il middleware può essere configurato e utilizzato come parte della mia funzione lambda. Quando la funzione viene invocata, avvia il Docker container ed esegue il middleware all'interno dell'ambiente containerizzato, garantendo che esso sia parte integrante dell'applicazione serverless.

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

In questa sezione vengono descritti i componenti e le funzionalità sviluppate durante il progetto di stage. Per questioni di chiarezza e di ordine, questa sezione è suddivisa in due parti: la prima rappresenta la parte frontend, mentre la seconda rappresenta la parte backend.

4.4.1 Codifica front-end

4.4.1.1 Utils

La cartella utils contiene una serie di file typescript che contengono funzioni, utilities o moduli che forniscono funzionalità di supporto a varie parti dell'applicazione. Questi file sono progettati per semplificare compiti ripetitivi, astrazioni complesse o per fornire funzionalità condivise in più componenti. Ogni utils è accompagnato da un file di test che ne verifica il corretto funzionamento.
Di seguito sono elencati gli utils che ho sviluppato durante il progetto di stage.

Auth Il file Auth è un utils per la gestione dell'autenticazione utilizzando come supporto la libreria Azure msal. Essa è utile per interfacciarsi all'autenticazione verso il servizio Azure Active Directory.
Auth è composto da più metodi, a partire dalla funzione di login fino alla funzione di acquisizione del token di accesso. (Ha senso spiegare funzionamento silent token?)

Debounce Il file Debounce è un utils che ho creato per ritardare l'esecuzione di una funzione fino a quando non si verifica un certo intervallo di tempo in cui non si verificano chiamate.
Questo utils è stato utilizzato per ritardare l'esecuzione della chiamata POST per il filtraggio dei risultati della ricerca. In questo modo, quando l'utente digita nella barra di ricerca, la chiamata POST viene eseguita ogni 300 millisecondi, ignorando le chiamate precedenti.

EndpointApiCall L'utils EndpointApiCall è una funzione che ho creato per gestire le chiamate ai vari endpoint del middleware, per ottenere i dati relativi alle API disponibili nel sistema. Queste chiamate necessitano di token di autenticazione valido per la sessione corrente, necessario per evitare chiamate non autorizzate.

GetClients L'utils GetClients è una funzione progettata per effettuare una richiesta Http GET al progetto del middleware, al fine di ottenere la lista dei client disponibili nel sistema. Questa funzione è stata utilizzata per popolare la lista dei client disponibili per la ricerca, in modo da poter selezionare uno dei client disponibile, in base all'ambiente di staging. La chiamata necessita di un token di autenticazione valido per la sessione corrente, necessario per evitare chiamate non autorizzate.

GetResults L'utils getResults è una funzione progettata per effettuare una richiesta Http POST al middleware, che sono andato ad utilizzare per ritornare i risultati della ricerca. Infatti, la ricerca nel mio progetto è spostata lato beckend, che è una best practise usata in azienda. La chiamata è autenticata, e viene utilizzata insieme alla

funzione di debounce, per ritardare l'esecuzione della chiamata POST. Come body della chiamata, viene passato il testo che l'utente sta scrivendo nella barra di ricerca.

MsGraphApiCall L'utils MsGraphApiCall contine al suo interno due funzione che ho creato per effettuare chiamate all'API di Microsoft Graph. La prima funzione si tratta di una chiamata GET, che viene utilizzata per ottenere i dati relativi all'utente che ha effettuato l'accesso al portale. La seconda funzione è sempre una chiamata GET ad un altro endpoint di Microsoft, per ottenere informazioni secondarie sull'utente, come ad esempio l'immagine del profilo. Entrambe le chiamate necessitano di un token, diverso dal token utilizzato per le chiamate al middleware. Infatti, questo token è specifico per le chiamate verso Microsoft Graph.

SwaggerUtils

4.4.1.2 Config

ConfigAuth Il seguente file rappresenta un modulo di configurazione per l'autenticazione. Esso contiene le configurazioni per l'autenticazione verso Azure Active Directory, come il clientId, il redirectUri o l'autority.

4.4.1.3 Stores

La cartella stores contiene i vari store utilizzati all'interno del progetto in vue.js. Gli store sono utili per gestire lo stato dell'applicazione, in modo da poterlo condividere tra più componenti.
Di seguito sono descritti gli store che ho sviluppato durante il progetto di stage.

Auth Il seguente store è stato creato per gestire lo stato dell'autenticazione utente e l'interazione con un servizio di autenticazione esterno. Esso utilizza al suo interno l'utils Auth che ho descritto in precedenza, e contiene delle variabili reattive per la gestione dello stato dell'autenticazione e per la gestione dei token.
Ho scritto lo store Auth utilizzando uno stile di programmazione basato su funzione closure di JavaScript. Questo approccio consente di incapsulare lo stato dell'autenticazione consentendo un maggiore controllo sull'accesso alle variabili e alle funzioni all'interno del modulo.

Store Il seguente store, a differenza del precedente, utilizza Pinia, una libreria apposita per la gestione dello stato. Esso è utilizzato per la gestione delle API e client-id disponibili nell'applicazione. Infatti, lo store inizializza le variabili reattive per la gestione delle API e dei client-id, effettuando una chiamata al middleware, utilizzando gli utils appositi definiti precedentemente.

4.4.1.4 Router

La cartella router contiene le rotte definite nel progetto, ovvero le route che l'utente può visitare all'interno dell'applicazione e un helper per la gestione del router.

CustomNavigation Il seguente file rappresenta una classe con due metodi: il primo utilizzato nel flusso di autenticazione, più precisamente nelle richieste con pop-up, mentre il secondo è utilizazto per convertire qualsiasi URI di reindirizzamento completoq in un URI di reindirizzamento relativo, in modo che il Vue Router possa gestire correttamente il reindirizzamento in modo sicuro.

4.4.1.5 Views

La seguente sezione contiene le views, ovvero le pagine dell'applicazione. Di seguito sono descritte le views che ho sviluppato durante il progetto di stage.

LoginView La LoginView è la prima pagina con cui l'utente interagisce. Essa contiene un bottone che permette di effettuare il login tramite pop-up, tramite un account Microsoft 365. La pagina consente di accedere al portale, ed è l'unica pagina di tutto il progetto che non richiede l'autenticazione.
Quando un'utente non autenticato tenterà di accedere a qualsiasi altra pagina del portale, verrà sempre reindirizzamento automaticamente a questa view. La pagina al suo interno contiene un unico componente, ovvero LoginButton, che verrà descritto in seguito. Inoltre, per permettere un uso comodo anche in schermi di piccole dimensioni,

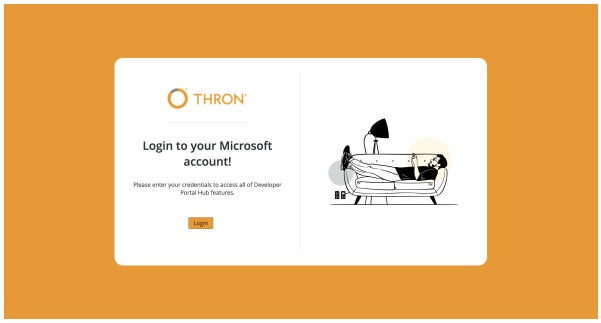

Figura 4.1: LoginView

o comunnque utilizzando una scheda del browser ristretta, ho sviluppato un design responsive, che permette di visualizzare il contenuto in maniera ottimale anche su spazi ridotti. Ciò secondo me è necessario perchè trattandosi di un portale per la consultazione di documentazione, è possibile che l'utente lo utilizzi in combinazione con altre schede del browser.

da aggiugere foto design responsive

HomeView La seguente view rappresenta la prima pagina visibile dopo aver effettuato il login, infatti dopo che l'autenticazione si è conclusa con successo, l'utente viene reindirizzato a questa pagina.

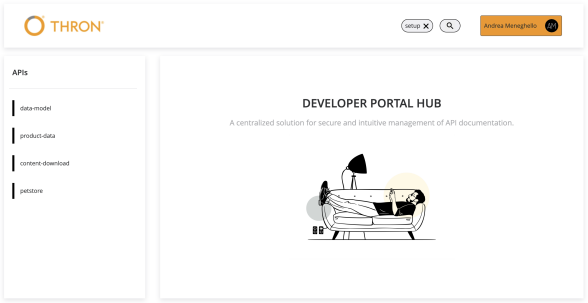

Figura 4.2: HomeView

NotFoundView NotFoundView testo

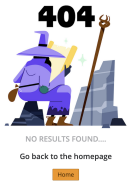

Figura 4.3: NotFoundView

4.4.1.6 Components

HeaderNav HeaderNav testo

MainContent Swagger testo

SideBar Sidebar testo

StartPage Start page testo

SwaggerLoader Swagger loader testo

DownloadButton download button testo

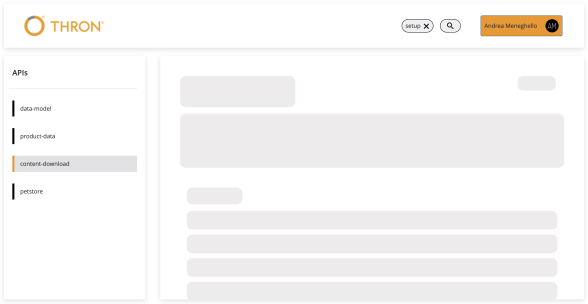

Figura 4.4: SwaggerLoader

LogoutButton logout button testo

LoginButton login button testo

SearchButton SearchButton testo

SearchBar Search bar testo

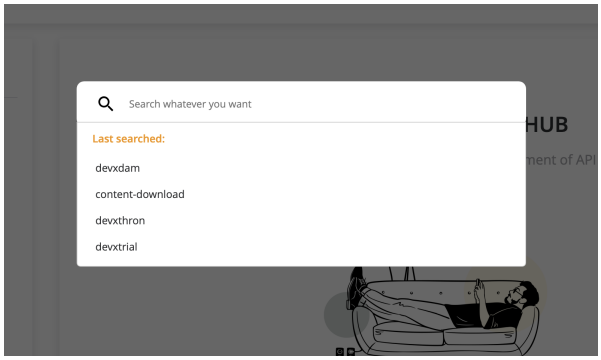

Figura 4.5: SearchBar

Autocomplete Autocomplete testo

Chip chip testo da scrivere

Filter filter testo

OptionList option list testo

OptionListItem OptionListItem testo

SnackBar snack bar testo da scrivere

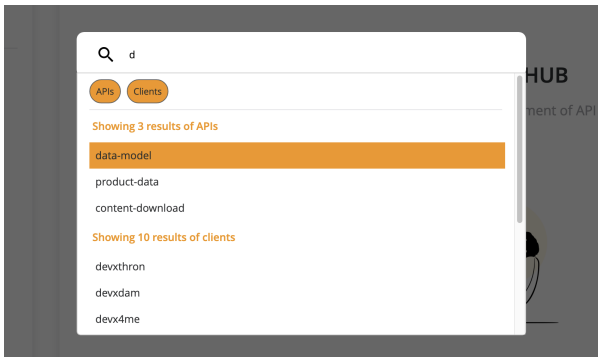

Figura 4.6: Autocomplete

Figura 4.7: Chip

Loader Loader testo da scrivere

4.4.2 Codifica back-end

Figura 4.8: SnackBar

Figura 4.9: Loader

Capitolo 5

Attività di verifica e validazione

In questo capitolo saranno descritti i processi di verifica e validazione del prodotto. In particolare verranno illustrati i test implementati, i risultati ottenuti e le possibili migliorie future del prodotto.

5.1 Test di unità

intro

Views

Tabella 5.1: Tabella del tracciamento dei test di unità views

Codice	Oggetto	Descrizione	Stato
TU-1	LoginView	Il progetto deve essere accompagnato da documentazione tecnica e funzionale	Superato
TU-2	HomeView	text	Superato
TU-3	NotFoundView	text	Superato

Components

Tabella 5.3: Tabella del tracciamento dei test di unità componenti

Codice	Oggetto	Descrizione	Stato
TU-4	HeaderNav	text	Superato
TU-5	MainContent	text	Superato
TU-6	SideBar	text	Superato

TU-7	StartPage	text	Superato
TU-8	DownloadButton	text	Superato
TU-9	LogoutButton	text	Superato
TU-10	LoginButton	text	Superato
TU-11	SearchButton	text	Superato
TU-12	SearchBar	text	Superato
TU-13	Autocomplete	text	Superato
TU-14	Chip	text	Superato
TU-15	Filter	text	Superato
TU-16	OptionList	text	Superato
TU-17	OptionListItem	text	Superato
TU-18	SnackBar	text	Superato

Utils

Tabella 5.5: Tabella del tracciamento dei test di unità utils

Codice	Oggetto	Descrizione	Stato
TU-19	Auth	Il progetto deve essere accompagnato da documentazione tecnica e funzionale	Superato
TU-20	Debounce	text	Superato
TU-21	endpointsApiCall	text	Superato
TU-22	getClients	text	Superato
TU-23	getResults	text	Superato
TU-24	MsGraphApiCall	text	Superato

logout button

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/.