

SWEG

Software Engineering Group

swegsoftware@gmail.com

Manuale Sviluppatore

Informazioni sul documento

Redattori:	Davide M.	Gabriel R.	Marco B.
Verificatori:	Davide S.	Andrea M.	Gabriel R.
Approvazione:	Davide M.		
Destinatari:	T. Vardanega	R. Cardin	Zero12
Versione:	1.0.0		
Uso:	Esterno		

Registro dei Cambiamenti - Changelog

<i>Versione</i>	<i>Data</i>	<i>Autore</i>	<i>Verificatore</i>	<i>Dettaglio</i>
1.0.0	2023-05-17	Davide Milan	Gabriel Rovesti	Approvazione
0.9.9	2023-05-16	Davide Milan	Gabriel Rovesti	Scrittura sezione 7 e relative sottosezioni. Vecchia sezione 7 diventata sezione 8. Aggiornato Sommario
0.9.7	2023-05-11	Davide Milan	Andrea Meneghello	Completamento chiamate API e relative sottosezioni
0.9.5	2023-05-11	Gabriel Rovesti	Davide Milan	Aggiunta indici e bozza chiamate API
0.9.0	2023-04-28	Marco Bernardi	Davide Milan	Scrittura sezione 6 e relative sottosezioni
0.8.0	2023-04-26	Davide Milan	Andrea Meneghello	Scrittura sezione 5 e relative sottosezioni
0.7.0	2023-04-24	Davide Milan	Andrea Meneghello	Ampliamento sezioni 4,5, ristrutturazione struttura e indicizzazione contenuti; aggiunta sezione 7
0.6.0	2023-04-20	Gabriel Rovesti	Andrea Meneghello	Correzione sezione 2 e bozza impostazione sezione 5, impostazione base sezione 6
0.4.0	2023-04-10	Gabriel Rovesti	Andrea Meneghello	Scrittura sezioni 3 e 4 con sottosezioni; inizio sezione 5
0.3.0	2023-04-03	Gabriel Rovesti	Andrea Meneghello	Scrittura sezione 2 e sezione 3, con corrispondenti sottosezioni
0.2.0	2023-03-27	Davide Milan	Davide Sgrazzutti	Scrittura sezione 1 e sottosezioni dello stesso capitolo
0.1.0	2023-03-20	Davide Milan	Davide Sgrazzutti	Inizializzazione documento e creazione indici e struttura

Sommario

Sommario	2
Elenco delle immagini	7
Elenco delle tabelle.....	9
1 Introduzione.....	10
1.1 Scopo del documento	10
1.2 Scopo del prodotto	10
1.3 Glossario	11
1.4 Maturità e miglioramenti.....	11
1.5 Riferimenti	11
1.5.1 Riferimenti normativi	11
1.5.2 Riferimenti informativi	11
1.5.3 Riferimenti tecnici	11
2 Tecnologie.....	12
3 Configurazione	13
3.1 Requisiti hardware	13
3.2 Requisiti software	14
4 Installazione	14
4.1 Clonazione del repository	15
4.2 Avvio completo dell'applicazione	15
5 Preparazione ambiente di lavoro	16
5.1 AWS SDK	16
5.2 AWS Amplify	16
6 Documentazione API	17
6.1 Lista di chiamate GET	17
6.1.1 getAllTenants.....	17
6.1.1.1 Endpoint.....	17
6.1.1.2 Esempio di richiesta.....	17
6.1.1.3 Esempio di risposta.....	17
6.1.2 getTenant	18
6.1.2.1 Endpoint.....	18
6.1.2.2 Parametri	18
6.1.2.3 Esempio di richiesta.....	18
6.1.2.4 Esempio di risposta.....	18
6.1.3 getAdmins.....	19
6.1.3.1 Endpoint.....	19
6.1.3.2 Parametri	19
6.1.3.3 Esempio di richiesta.....	19
6.1.3.4 Esempio di risposta.....	19
6.1.4 tenantGetUsers	20
6.1.4.1 Endpoint.....	20

6.1.4.2 Parametri	20
6.1.4.3 Esempio di richiesta	20
6.1.4.4 Esempio di risposta	20
6.1.5 getTenantLanguages	21
6.1.5.1 Endpoint.....	21
6.1.5.2 Parametri	21
6.1.5.3 Esempio di richiesta	21
6.1.5.4 Esempio di risposta	21
6.1.6 getAllCategories.....	21
6.1.6.1 Endpoint.....	21
6.1.6.2 Parametri	21
6.1.6.3 Esempio di richiesta	21
6.1.6.4 Esempio di risposta	22
6.1.7 getResetCode	22
6.1.7.1 Endpoint.....	22
6.1.7.2 Parametri	22
6.1.7.3 Esempio di richiesta	22
6.1.7.4 Esempio di risposta	22
6.1.8 getAllUsers	23
6.1.8.1 Endpoint.....	23
6.1.8.2 Esempio di richiesta	23
6.1.8.3 Esempio di risposta	23
6.1.9 adminGetUser	24
6.1.9.1 Endpoint.....	24
6.1.9.2 Parametri	24
6.1.9.3 Esempio di richiesta	24
6.1.9.4 Esempio di risposta	24
6.1.10 getUserTenant.....	25
6.1.10.1 Endpoint.....	25
6.1.10.2 Parametri	25
6.1.10.3 Esempio di richiesta	25
6.1.10.4 Esempio di risposta	25
6.1.11 getAllTexts	26
6.1.11.1 Endpoint.....	26
6.1.11.2 Parametri	26
6.1.11.3 Esempio di richiesta	26
6.1.11.4 Esempio di risposta	26
6.1.12 getText.....	27
6.1.12.1 Endpoint.....	27
6.1.12.2 Parametri	27
6.1.12.3 Esempio di richiesta	27
6.1.12.4 Esempio di risposta	27

6.1.13	getTranslationLanguages.....	28
6.1.13.1	Endpoint.....	28
6.1.13.2	Parametri	28
6.1.13.3	Esempio di richiesta	28
6.1.13.4	Esempio di risposta	28
6.2	Lista di chiamate PUT	29
6.2.1	putOriginalText.....	29
6.2.1.1	Endpoint.....	29
6.2.1.2	Parametri	29
6.2.1.3	Esempio di richiesta	29
6.2.1.4	Esempio di risposta	30
6.2.2	putTranslation	30
6.2.2.1	Endpoint.....	30
6.2.2.2	Parametri	30
6.2.2.3	Esempio di richiesta	30
6.2.2.4	Esempio di risposta	31
6.2.3	putAcceptText	31
6.2.3.1	Endpoint.....	31
6.2.3.2	Parametri	31
6.2.3.3	Esempio di richiesta	32
6.2.3.4	Esempio di risposta	32
6.2.4	putRejectText	32
6.2.4.1	Endpoint.....	32
6.2.4.2	Parametri	32
6.2.4.3	Esempio di richiesta	32
6.2.4.4	Esempio di risposta	33
6.3	Lista di chiamate POST	33
6.3.1	addTenant	33
6.3.1.1	Endpoint.....	33
6.3.1.2	Esempio di richiesta	33
6.3.1.3	Esempio di risposta	34
6.3.2	addLanguage	34
6.3.2.1	Endpoint.....	34
6.3.2.2	Parametri	34
6.3.2.3	Esempio di richiesta	34
6.3.2.4	Esempio di risposta	35
6.3.3	signUpUser	35
6.3.3.1	Endpoint.....	35
6.3.3.2	Parametri	35
6.3.3.3	Esempio di richiesta	35
6.3.3.4	Esempio di risposta	36
6.3.4	postOriginalText	36

6.3.4.1 Endpoint.....	36
6.3.4.2 Esempio di richiesta.....	36
6.3.4.3 Parametri	37
6.3.4.4 Esempio di risposta.....	37
6.3.5 resetPassword	37
6.3.5.1 Endpoint.....	37
6.3.5.2 Esempio di richiesta.....	37
6.3.5.3 Parametri	38
6.3.5.4 Esempio di risposta.....	38
6.4 Lista di chiamate DELETE	38
6.4.1 deleteTenant	38
6.4.1.1 Endpoint.....	38
6.4.1.2 Esempio di richiesta.....	38
6.4.1.3 Parametri	38
6.4.1.4 Esempio di risposta.....	39
6.4.2 deleteLanguage	39
6.4.2.1 Endpoint.....	39
6.4.2.2 Esempio di richiesta.....	39
6.4.2.3 Parametri	39
6.4.2.4 Esempio di risposta.....	39
6.4.3 removeCategory.....	40
6.4.3.1 Endpoint.....	40
6.4.3.2 Esempio di richiesta.....	40
6.4.3.3 Parametri	40
6.4.3.4 Esempio di risposta.....	40
6.4.4 deleteUser	40
6.4.4.1 Endpoint.....	40
6.4.4.2 Esempio di richiesta.....	40
6.4.4.3 Parametri	40
6.4.4.4 Esempio di risposta.....	41
6.4.5 deleteOriginalText.....	41
6.4.5.1 Endpoint.....	41
6.4.5.2 Esempio di richiesta.....	41
6.4.5.3 Parametri	41
7 Documentazione libreria client	42
7.1 Struttura libreria	42
7.1.1 Funzioni non esportate	42
7.1.1.1 findElementsMatchingIdPattern()	42
7.1.1.2 addTextToElement(element, text).....	43
7.1.2 Oggetto esportato : Translatify	43
7.1.3 Metodi Oggetto	44
7.1.3.1 constructor(tenantId, apiKey).....	44

7.1.3.2	async retrieveLanguages()	45
7.1.3.3	getText(categoryName, title)	45
7.1.3.4	handleLanguageChange(newLanguage)	46
7.2	Chiamate API utilizzate	47
7.2.1	getLanguages	47
7.2.1.1	Endpoint	47
7.2.1.2	Parametri	47
7.2.1.3	Esempio di richiesta	47
7.2.1.4	Esempio di risposta	47
7.2.2	getText	47
7.2.2.1	Endpoint	47
7.2.2.2	Parametri	48
7.2.2.3	Esempio di richiesta	48
7.2.2.4	Esempio di risposta	48
7.3	Esempio di utilizzo	48
7.3.1	Codice HTML	48
7.3.2	Script di esempio per una pagina interattiva	49
7.3.3	Risultato finale	51
8.1	Aggiunta di nuovi ruoli per utente	54
8.2	Aggiunta di funzionalità di ricerca e di filtro avanzate	55

Elenco delle immagini

[Immagine 1: Esempio di risposta della chiamata getAllTenants](#)

[Immagine 2: Esempio di risposta della chiamata getTenant](#)

[Immagine 3: Esempio di risposta della chiamata getAdmins](#)

[Immagine 4: Esempio di risposta della chiamata tenantGetUsers](#)

[Immagine 5: Esempio di risposta della chiamata getTenantLanguages](#)

[Immagine 6: Esempio di risposta della chiamata getAllCategories](#)

[Immagine 7: Esempio di risposta della chiamata getResetCode](#)

[Immagine 8: Esempio di risposta della chiamata getAllUsers](#)

[Immagine 9: Esempio di risposta della chiamata adminGetUser](#)

[Immagine 10: Esempio di risposta della chiamata getUserTenant](#)

[Immagine 11: Esempio di risposta della chiamata getUserTenant](#)

[Immagine 12: Esempio di risposta della chiamata getText](#)

[Immagine 13: Esempio di risposta della chiamata getTranslationLanguages](#)

[Immagine 14: Corpo della chiamata putOriginalText](#)

[Immagine 15: Esempio di risposta della chiamata putOriginalText](#)

[Immagine 16: Corpo della chiamata putTranslation](#)

[Immagine 17: Esempio di risposta della chiamata putTranslation](#)

[Immagine 18: Esempio di risposta della chiamata putAcceptText](#)

[Immagine 19: Corpo della chiamata putRejectText](#)

[Immagine 20: Esempio di risposta della chiamata putRejectText](#)

[Immagine 21: Corpo della chiamata addTenant](#)

[Immagine 22: Esempio di risposta della chiamata addTenant](#)

[Immagine 23: Corpo della chiamata addLanguage](#)

[Immagine 24: Esempio di risposta della chiamata addLanguage](#)

[Immagine 25: Corpo della chiamata signUpUser](#)

[Immagine 26: Esempio di risposta della chiamata signUpUser](#)

[Immagine 27: Corpo della chiamata signUpUser](#)

[Immagine 28: Esempio di risposta della chiamata signUpUser](#)

[Immagine 29: Corpo della chiamata signUpUser](#)

[Immagine 30: Esempio di risposta della chiamata deleteTenant](#)

[Immagine 31: Esempio di risposta della chiamata delete language](#)

[Immagine 32: Esempio di risposta della chiamata remove category](#)

[Immagine 33: Esempio di risposta della chiamata deleteUser](#)

[Immagine 34: Esempio di risposta della chiamata deleteOriginalText](#)

[Immagine 35: Codice funzione findElementsMatchingIdPattern\(\)](#)

[Immagine 36: Codice funzione addTextToElement\(\)](#)

[Immagine 37: Struttura oggetto Translatify](#)

[Immagine 38: Codice costruttore oggetto Translatify](#)

[Immagine 39: Codice metodo retrieveLanguages\(\)](#)

[Immagine 40: Codice metodo getText\(categoryName, title\)](#)

[Immagine 41: Codice metodo handleLanguageChange\(newLanguage\)](#)

[Immagine 42: Esempio di risposta della chiamata getLanguages](#)

[Immagine 43: Esempio di risposta della chiamata getText](#)

[Immagine 44: Esempio di ID per un elemento HTML](#)

[Immagine 45: Importare libreria in un file JS/TS](#)

[Immagine 46: Esempio di come impostare un picker ed una funzione da utilizzare per gestire il cambio lingua da utilizzare con la libreria](#)

[Immagine 47: Esempio di come impostare Translatify e gestirlo con il localStorage](#)

[Immagine 48: Esempio di come impostare la lingua da utilizzare sull'oggetto Translatify al caricamento e salvarla in localStorage](#)

[Immagine 49: Esempio di come impostare la lingua da utilizzare sul picker al caricamento e le options del picker](#)

[Immagine 50: Demo con lingua impostata in Italiano](#)

[Immagine 51: Demo con lingua impostata in Inglese](#)

[Immagine 52: Demo con lingua impostata in Cinese](#)

[Immagine 53: Link di navigazione forniti per utente](#)

[Immagine 54: Utenti a cui è permessa la navigazione](#)

[Immagine 55: Funzionalità di ricerca offerte](#)

[Immagine 56: Codice del componente Picker](#)

[Immagine 57: Esempio di utilizzo del componente Picker](#)

Elenco delle tabelle

[Tabella 1 - Tabella di descrizione delle tecnologie](#)

[Tabella 2 - Tabella di descrizione degli strumenti di test](#)

[Tabella 3 - Tabella di descrizione dei requisiti hardware](#)

[Tabella 4 - Tabella di descrizione dei requisiti software](#)

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è di fornire agli sviluppatori le informazioni necessarie per comprendere l'architettura del software e le sue funzionalità, nonché le procedure per configurare, installare e utilizzare il software. Il Manuale Sviluppatore fornisce inoltre indicazioni sulla struttura del codice sorgente, sulle librerie e sulle API^G utilizzate, sui requisiti di sistema, sulla gestione degli errori e sui test da effettuare per garantire la qualità del software.

Il documento è rivolto ai programmatori e agli sviluppatori che intendono estendere o personalizzare il software, in modo da poter comprendere a fondo la struttura e il funzionamento del prodotto. In questo modo, il documento fornisce le informazioni necessarie per garantire una corretta manutenzione e tutti i dettagli per un successivo utilizzo ed implementazione.

1.2 Scopo del prodotto

L'obiettivo del progetto è realizzare una piattaforma per la gestione delle traduzioni tramite un'infrastruttura multi-tenant^G, permettendo di gestire testi in lingue diverse e a diversi utenti. Esso si rivolge a siti che offrono la possibilità di essere visualizzati in lingue differenti e gestire bilateralmente le traduzioni dei testi che li compongono, secondo un meccanismo di accesso, collaborazione, approvazione e rifiuto.

La piattaforma consente agli utenti di accedere al proprio Tenant e interagire a diverso livello con le traduzioni presenti. L'utilizzo delle tecnologie AWS^G permette di riconoscere il tipo di utente tramite uno specifico token^G a lui assegnato, permettendogli una specifica gestione della piattaforma. In particolare, possiamo distinguere:

- gli utenti finali (definiti come User^G) potranno visualizzare un insieme di testi che dovranno tradurre;
- gli utenti amministratori (definiti come Admin^G), che gestiscono un singolo Tenant, potranno impostare una serie di lingue secondarie, visualizzare e modificare testi originali, approvare e rifiutare le traduzioni presenti. Essi possono inoltre creare, modificare e cancellare categorie di traduzioni presenti e inviare testi tradotti assegnati;
- gli utenti definiti come SuperAdmin^G, con permessi di gestione di tutti i Tenant, possono gestire ciascuno di questi e gli utenti ad essi associati.

Ogni traduzione è raggruppata in una categoria, e le traduzioni sono suddivise tra lingua di default impostata e una serie di lingue secondarie. Inoltre, l'applicativo consente di visualizzare le traduzioni eseguite o non eseguite e di visualizzare i testi tradotti in una lingua tramite API.

Per fornire la massima compatibilità, essa sarà fruibile tramite browser^G, in grado di supportare correttamente le tecnologie di base HTML^G, CSS^G, JavaScript^G ed altre successivamente definite.

Sarà inoltre resa disponibile una libreria client sviluppata in Javascript per permettere la fruizione di tutti questi testi da qualsiasi webapp.

1.3 Glossario

Al fine di evitare incomprensioni relative alla terminologia usata all'interno del documento, viene fornito un Glossario nel file apposito (Glossario v.2.0.0), tale da non avere terminologie ambigue nell'attività progettuale individuata e dandone una definizione precisa. Ogni termine avrà nel documento una lettera G come apice, per meglio evidenziare la loro appartenenza al documento indicato.

1.4 Maturità e miglioramenti

Il presente documento è redatto con un approccio incrementale, al fine di poter implementare facilmente cambiamenti nel corso del tempo a seconda di esigenze concordate bilateralmente tra membri del gruppo e proponente. Pertanto, non può essere considerato definitivo e completo in questa versione.

1.5 Riferimenti

1.5.1 Riferimenti normativi

- Capitolato^G C4-Piattaforma di localizzazione testi:
 - <https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C4.pdf>
- Regolamento progetto didattico:
 - <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/PD02.pdf>
- Norme di Progetto v.2.0.0

1.5.2 Riferimenti informativi

- Slide T07 del corso di Ingegneria del Software - Analisi dei requisiti:
 - <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T06.pdf>
- Slide del corso di Ingegneria del Software - Progettazione e programmazione: Diagrammi delle classi:
 - <https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20Use%20Case.pdf>
- Slide del corso di Ingegneria del Software - Solid Programming:
 - https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf
- Repository interna delle API:
 - <https://github.com/SWEG-Software-Engineering-Group/API>

1.5.3 Riferimenti tecnici

- Documentazione di React:
 - <https://react.dev/>
- NPM:
 - <https://docs.npmjs.com/>
- Documentazione di TypeScript:
 - <https://www.typescriptlang.org/docs/>

- Documentazione di AWS:
 - <https://docs.aws.amazon.com/>

2 Tecnologie

In questa sezione viene fornita una panoramica generale delle tecnologie utilizzate per la realizzazione del prodotto in questione. Vengono infatti descritte le tecnologie, gli strumenti e le librerie necessari per lo sviluppo, il test e la distribuzione del prodotto. In particolare, verranno trattate le tecnologie impiegate per la realizzazione del front-end e del back-end, nonché per la gestione del database e l'integrazione con altri servizi.

<i>Tecnologia</i>	<i>Descrizione</i>	<i>Versione</i>
Linguaggi		
JavaScript	Linguaggio utilizzato per la gestione di eventi invocati dall'utente,	ECMAScript 2021
TypeScript ^G	Utilizzo di tecnologie scalabili	5.0.x
Strumenti e servizi		
Node.js ^G	Ambiente di runtime open-source per l'esecuzione di codice JavaScript lato server tramite appositi script	19.0.x
NPM ^G	Gestore di pacchetti per il linguaggio JavaScript e l'ambiente di esecuzione Node.js	9.6.x
Axios	Libreria JavaScript per il client HTTP basata su Promise, che consente di eseguire richieste HTTP da browser e da Node.js	1.6.x
AWS Cognito ^G	Servizio di gestione delle identità ed autenticazione basato sui ruoli	2023-16-02
AWS DynamoDB ^G	Servizio di database non relazionale gestito in modo scalabile	2019-11-21
AWS Lambda ^G	Servizio di elaborazione serverless ^G di codice senza dover configurare/scalare l'infrastruttura server, eseguendo il codice in modo scalabile	2023-03-16
AWS API Gateway ^G	Servizio di gestione delle API ^G che permette di creare, pubblicare e proteggere e le stesse in modo sicuro, concentrandosi sulla logica di business	2023-04-06
Git ^G	Sistema di controllo versione distribuito utilizzato per la gestione del codice sorgente dal parte del gruppo di progetto	2.4.x

Tabella 1 - Tabella di descrizione delle tecnologie

Inoltre, vengono definite le seguenti tecnologie per la fase di test^G:

Tecnologia	Descrizione	Versione
Analisi statica		
ESLint	Strumento JavaScript che aiuta a individuare gli errori di codice e le pratiche non ottimali	8.38.x
Prettier	Strumento di formattazione del codice che aiuta a mantenere uno stile di codifica coerente e leggibile	3.0.x
Jest	Framework di test basato su JavaScript con funzionalità di creazione di mock e il testing del codice in modo asincrono.	29.0.x
Analisi dinamica		
React Testing Library	Libreria di test integrata nativamente che consente di testare il comportamento dei componenti ^G React da una prospettiva degli utenti finali.	14.0.x
GitHub Actions	Servizio di CI/CD ^G per automatizzare il processo di build ^G , test e deploy ^G del progetto software	/

Tabella 2 - Tabella di descrizione degli strumenti di test

3 Configurazione

In questa sezione sono elencati i requisiti minimi necessari all'esecuzione dell'applicazione, elencando le caratteristiche necessarie atte alla configurazione dell'ambiente di sviluppo del progetto.

3.1 Requisiti hardware

L'applicazione esegue su browser, come tale non si individuano dei requisiti specifici, fissati da parte del proponente, del capitolato o dal progetto stesso. I seguenti, pertanto, sono individuati come riferimento di massima per l'esecuzione del prodotto creato.

Componente	Requisito
Processore	Quad-Core 3,2 GHz
Memoria	8GB DDR4
Scheda grafica	Supporto a scheda grafica integrata con supporto OpenGL
Connessione Internet	Connessione Internet stabile e veloce, in grado di supportare le esigenze di traffico dell'applicazione

Tabella 3 - Tabella di descrizione dei requisiti hardware

3.2 Requisiti software

L'applicazione è stata testata sui browser principali, di cui si riportano le versioni iniziali, dalle quali si è cominciato a sviluppare il progetto, considerando incrementalmente le versioni più recenti dei singoli browser.

<i>Browser</i>	<i>Versione</i>
Google Chrome	110.0
Microsoft Edge	110.0
Mozilla Firefox	109.0
Opera	95.0
Safari	16.0

Tabella 4 - Tabella di descrizione dei requisiti software

Inoltre, come descritto sopra, per poter accedere a tutte le funzionalità di sistema, occorrerà disporre della versione 19.0 del software Node.js, al fine di poter integrare facilmente le ultime funzionalità. Si consiglia una versione superiore alla 17.0 per poter pienamente usufruire del supporto della libreria grafica Material UI.

4 Installazione

Per sviluppare ulteriormente l'applicazione considera i seguenti passi, definiti entrambi come obbligatori e meglio dettagliati nelle successive sottosezioni. Tali passi avvengono in sequenza, come poi spiegato:

- Clonazione del repository
- Avvio completo dell'applicazione

4.1 Clonazione del repository

1. Scaricare il codice direttamente in formato `.zip` dal seguente link:
 - a. <https://github.com/SWEG-Software-Engineering-Group/Translatify/archive/refs/heads/main.zip>

Alternativamente, è possibile:

1. Avviare un terminale
2. Spostarsi nella cartella dove si vuole clonare la repository con il comando:
 - a. `cd path`
3. Utilizzando Git che deve essere installato in locale, la repo viene clonata utilizzando il comando:
 - a. `git clone https://github.com/SWEG-Software-Engineering-Group/Translatify`
4. Utilizzando Git che deve essere installato in locale, la repo viene clonata utilizzando il comando:
 - a. `cd path/Translatify`

4.2 Avvio completo dell'applicazione

Una volta clonato il repository di riferimento, per l'avvio del server si usino i seguenti comandi:

1. Aprire un terminale all'interno della cartella contenente i file denominata *"Translatify"*
2. Usare il comando (necessario per installare le dipendenze al primo avvio):
`npm install`
3. Successivamente fare la build dell'app con il comando:
`npm run build`
4. L'applicazione eseguirà automaticamente sul browser predefinito all'indirizzo:
`http://localhost:3000`

5 Preparazione ambiente di lavoro

Questa sezione spiega come configurare l'ambiente di lavoro di modo che possa essere replicato da chiunque per operare nelle stesse condizioni del team *SWEG*. Tutti gli strumenti di seguito elencati sono consigliati in fase di preparazione, al fine di interagire correttamente con il prodotto e installare le relative dipendenze appena possibile.

5.1 AWS SDK

Per permettere l'interazione di base con il prodotto è necessario, come stabilito sopra, installare Node.js. Successivamente, occorrerà aprire un terminale oppure la console di Node.js ed eseguire il comando:

```
npm install -g aws-sdk
```

Si consiglia l'aggiornamento frequente della libreria *aws-sdk* che viene settimanalmente aggiornata, ai fini di garantire la massima compatibilità.

5.2 AWS Amplify

Il framework AWS Amplify facilita l'integrazione dei servizi di AWS per i servizi di autenticazione e comunicazione con il servizio Cognito, utile per l'opportuna generazione dei ruoli e differenziazione delle aree utente previste.

Successivamente, occorrerà aprire un terminale oppure la console di Node.js ed eseguire il comando:

```
npm install -g @aws-amplify/cli
```

La prima configurazione dell'ambiente di lavoro, stabilendo le dipendenze necessarie e la scelta del server e regione di riferimento sarà prevista con il comando:

```
amplify configure
```

L'impostazione opportuna di permessi e ruoli per collegamento del progetto al backend su cloud AWS sarà invece possibile, spostandosi nella cartella principale (*root*) del progetto con il comando:

```
amplify init
```

6 Documentazione API

La seguente sezione descrive dettagliatamente la libreria creata dal gruppo *SWEG* per comunicare con l'applicazione, fornendo una panoramica completa delle API^G disponibili, la sintassi di ciascuna richiesta, compresi parametri ed esempi di codice per il loro utilizzo. In questo modo, gli sviluppatori interessati ad estendere le funzionalità avranno una piena comprensione del software, implementando nuove funzionalità in modo programmatico ed automatizzato senza dover accedere manualmente all'interfaccia utente.

6.1 Lista di chiamate GET

6.1.1 getAllTenants

- Ritorna la lista di tutti i Tenant presenti nel sistema

6.1.1.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/allTenants
```

6.1.1.2 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/allTenants
```

6.1.1.3 Esempio di risposta

```

1  [
2    {
3      "defaultLanguage": "string",
4      "tenantName": "string",
5      "languages": [
6        "string"
7      ],
8      "categories": [
9        "string"
10     ],
11     "creationDate": 0,
12     "admins": [
13       "string"
14     ],
15     "users": [
16       "string"
17     ]
18   }
19 ]
20
```

Immagine 1: Esempio di risposta della chiamata getAllTenants

6.1.2 getTenant

- Ritorna le informazioni relative ad un Tenant specifico

6.1.2.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/info`

6.1.2.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.1.2.3 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/info
```

6.1.2.4 Esempio di risposta

```

1  {
2      "tenant": {
3          "tenantName": "DocAPI",
4          "languages": [
5              "English",
6              "Japanese"
7          ],
8          "defaultLanguage": "Italian",
9          "admins": [
10             "59e5680a-c781-4c92-b77e-47a00c62482e",
11             "453131f0-c061-449b-9388-bae39cfbba18"
12         ],
13         "categories": [
14             {
15                 "name": "Header",
16                 "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
17             }
18         ],
19         "creationDate": 1683821162.969,
20         "id": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
21         "users": [
22             "a216c5f4-c2cc-4db1-ad50-b70eb629f7f4"
23         ]
24     }
25 }
```

Immagine 2: Esempio di risposta della chiamata getTenant

6.1.3 getAdmins

- Ritorna la lista degli admin associati al Tenant

6.1.3.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/admins`

6.1.3.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.1.3.3 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/admins`

6.1.3.4 Esempio di risposta

```

1  {
2      "Admins": [
3          {
4              "Username": "59e5680a-c781-4c92-b77e-47a00c62482e",
5              "UserAttributes": [
6                  {
7                      "Name": "custom:surname",
8                      "Value": "api"
9                  },
10                 {
11                     "Name": "sub",
12                     "Value": "59e5680a-c781-4c92-b77e-47a00c62482e"
13                 },
14                 {
15                     "Name": "email_verified",
16                     "Value": "true"
17                 },
18                 {
19                     "Name": "name",
20                     "Value": "admin"
21                 },
22                 {
23                     "Name": "email",
24                     "Value": "adminDocAPI@sweg.it"
25                 }
26             ],
27             "UserCreateDate": "2023-05-11T16:10:15.560Z",
28             "UserLastModifiedDate": "2023-05-11T16:45:59.892Z",
29             "Enabled": true,
30             "UserStatus": "CONFIRMED"
31         },

```

Immagine 3: Esempio di risposta della chiamata getAdmins

6.1.4 tenantGetUsers

- Ritorna la lista degli utenti associati al Tenant

6.1.4.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/users`

6.1.4.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.1.4.3 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/users`

6.1.4.4 Esempio di risposta

```

1  {
2      "Users": [
3          {
4              "Username": "a216c5f4-c2cc-4db1-ad50-b70eb629f7f4",
5              "UserAttributes": [
6                  {
7                      "Name": "custom:surname",
8                      "Value": "api"
9                  },
10                 {
11                     "Name": "sub",
12                     "Value": "a216c5f4-c2cc-4db1-ad50-b70eb629f7f4"
13                 },
14                 {
15                     "Name": "email_verified",
16                     "Value": "true"
17                 },
18                 {
19                     "Name": "name",
20                     "Value": "user"
21                 },
22                 {
23                     "Name": "email",
24                     "Value": "userDocAPI@sweg.it"
25                 }
26             ],
27             "UserCreateDate": "2023-05-11T16:10:52.166Z",
28             "UserLastModifiedDate": "2023-05-11T16:46:14.213Z",
29             "Enabled": true,
30             "UserStatus": "CONFIRMED"
31         }
32     ]
33 }
```

Immagine 4: Esempio di risposta della chiamata tenantGetUsers

6.1.5 getTenantLanguages

- Ritorna la lista delle lingue associate al Tenant

6.1.5.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/secondaryLanguages
```

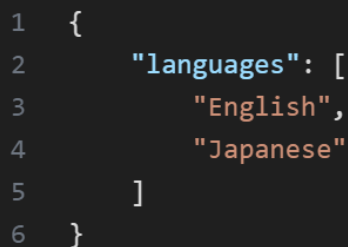
6.1.5.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.1.5.3 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/secondaryLanguages
```

6.1.5.4 Esempio di risposta



```
1  {
2      "languages": [
3          "English",
4          "Japanese"
5      ]
6  }
```

Immagine 5: Esempio di risposta della chiamata getTenantLanguages

6.1.6 getAllCategories

- Ritorna la lista delle categorie presenti nel Tenant

6.1.6.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/allCategories
```

6.1.6.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.1.6.3 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/allCategories
```

6.1.6.4 Esempio di risposta

```

1  {
2      "Categories": [
3          {
4              "name": "Header",
5              "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
6          }
7      ]
8  }
```

Immagine 6: Esempio di risposta della chiamata getAllCategories

6.1.7 getResetCode

- Invoca la procedura di invio token alla email dell'utente per iniziare il reset della password

6.1.7.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/{username-email}/getResetCode`

6.1.7.2 Parametri

- username
 - Identificativo dell'Utente associato al Tenant di riferimento: il suo valore può essere sia l'id dell'utente che la sua email.

6.1.7.3 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/59e5680a-c781-4c92-b77e-47a00c62482e/getResetCode`

oppure (ugualmente)

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/adminDocAPI@sweg.it/getResetCode`

6.1.7.4 Esempio di risposta

```

1  {
2      "user": "Reset code sent"
3  }
```

Immagine 7: Esempio di risposta della chiamata getResetCode

6.1.8 getAllUsers

- Ritorna la lista degli Utenti registrati nel sistema

6.1.8.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/users`

6.1.8.2 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/users`

6.1.8.3 Esempio di risposta

```

1  {
2    "users": [
3      {
4        "Username": "453131f0-c061-449b-9388-bae39cfbba18",
5        "Attributes": [
6          {
7            "Name": "custom:surname",
8            "Value": "MILAN"
9          },
10         {
11           "Name": "sub",
12           "Value": "453131f0-c061-449b-9388-bae39cfbba18"
13         },
14         {
15           "Name": "email_verified",
16           "Value": "true"
17         },
18         {
19           "Name": "name",
20           "Value": "DAVIDE"
21         },
22         {
23           "Name": "email",
24           "Value": "davide.milan.2@studenti.unipd.it"
25         }
26       ],
27       "UserCreateDate": "2023-05-11T16:48:04.380Z",
28       "UserLastModifiedDate": "2023-05-11T16:53:58.665Z",
29       "Enabled": true,
30       "UserStatus": "CONFIRMED"
31     },
32     {
33       "Username": "59e5680a-c781-4c92-b77e-47a00c62482e",
34       "Attributes": [
35         {
36           "Name": "custom:surname",
37           "Value": "api"
38         },

```

Immagine 8: Esempio di risposta della chiamata getAllUsers

6.1.9 adminGetUser

- Ritorna le informazioni relative ad un utente specifico

6.1.9.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/{username}/adminGetUser`

6.1.9.2 Parametri

- Username
 - Identificativo dell'Utente associato al Tenant di riferimento

6.1.9.3 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/59e5680a-c781-4c92-b77e-47a00c62482e/adminGetUser`

6.1.9.4 Esempio di risposta

```

1  {
2      "users": {
3          "Username": "59e5680a-c781-4c92-b77e-47a00c62482e",
4          "UserAttributes": [
5              {
6                  "Name": "custom:surname",
7                  "Value": "api"
8              },
9              {
10                 "Name": "sub",
11                 "Value": "59e5680a-c781-4c92-b77e-47a00c62482e"
12             },
13             {
14                 "Name": "email_verified",
15                 "Value": "true"
16             },
17             {
18                 "Name": "name",
19                 "Value": "admin"
20             },
21             {
22                 "Name": "email",
23                 "Value": "adminDocAPI@sweg.it"
24             }
25         ],
26         "UserCreateDate": "2023-05-11T16:10:15.560Z",
27         "UserLastModifiedDate": "2023-05-11T16:45:59.892Z",
28         "Enabled": true,
29         "UserStatus": "CONFIRMED"
30     }
31 }
```

Immagine 9: Esempio di risposta della chiamata adminGetUser

6.1.10 getUserTenant

- Ritorna il Tenant associato all'utente

6.1.10.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/{username}/tenant`

6.1.10.2 Parametri

- Username
 - Identificativo dell'Utente associato al Tenant di riferimento

6.1.10.3 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/59e5680a-c781-4c92-b77e-47a00c62482e/tenant`

6.1.10.4 Esempio di risposta

```

1  {
2    "tenants": [
3      {
4        "tenantName": "DocAPI",
5        "languages": [
6          "English",
7          "Japanese"
8        ],
9        "defaultLanguage": "Italian",
10       "admins": [
11         "59e5680a-c781-4c92-b77e-47a00c62482e",
12         "453131f0-c061-449b-9388-bae39cfbba18"
13       ],
14       "categories": [
15         {
16           "name": "Header",
17           "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
18         }
19       ],
20       "creationDate": 1683821162.969,
21       "id": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
22       "users": [
23         "a216c5f4-c2cc-4db1-ad50-b70eb629f7f4"
24       ]
25     }
26   ]
27 }
```

Immagine 10: Esempio di risposta della chiamata getUserTenant

6.1.11 getAllTexts

- Ritorna la lista di tutti i testi associati al Tenant

6.1.11.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/allTexts`

6.1.11.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.1.11.3 Esempio di richiesta

`curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/allTexts`

6.1.11.4 Esempio di risposta

```

1  {
2      "response": [
3          {
4              "idTenant": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
5              "language": "English",
6              "category": {
7                  "name": "Header",
8                  "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
9              },
10             "title": "TestRejectedText",
11             "text": null,
12             "state": 4,
13             "comment": "TestRejectedText Comment",
14             "link": "TestRejectedText Link",
15             "feedback": null
16         },
17         {
18             "idTenant": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
19             "language": "English",
20             "category": {
21                 "name": "Header",
22                 "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
23             },
24             "title": "TestText",
25             "text": null,
26             "state": 1,
27             "comment": "TestText comment",
28             "link": "TestText link",
29             "feedback": null
30         },

```

Immagine 11: Esempio di risposta della chiamata getUserTenant

6.1.12 getText

- Ritorna le informazioni relative ad un testo specifico

6.1.12.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/{Language}/{Category}/{Title}/text
```

6.1.12.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione del testo di interesse
- Category
 - Categoria di traduzione
- Title
 - Titolo del testo

6.1.12.3 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/English/Header/TestText/text
```

6.1.12.4 Esempio di risposta

```

1  {
2      "Text": {
3          "idTenant": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
4          "language": "English",
5          "category": {
6              "name": "Header",
7              "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
8          },
9          "title": "TestText",
10         "text": null,
11         "state": 1,
12         "comment": "TestText comment",
13         "link": "TestText link",
14         "feedback": null
15     }
16 }
```

Immagine 12: Esempio di risposta della chiamata getText

6.1.13 getTranslationLanguages

- Ritorna la lista delle lingue in tradurre/in cui è stato tradotto un testo specifico

6.1.13.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/category/{Category}/TestText/translationLanguages
```

6.1.13.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione del testo di interesse
- Category
 - Categoria di traduzione
- Title
 - Titolo del testo

6.1.13.3 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/category/244de0d7-d3fb-42c6-9c04-f7e64f83f319/{Title}/translationLanguages
```

6.1.13.4 Esempio di risposta

```
1 {
2   "response": [
3     "English",
4     "Japanese"
5   ]
6 }
```

Immagine 13: Esempio di risposta della chiamata getTranslationLanguages

6.2 Lista di chiamate PUT

6.2.1 putOriginalText

- Aggiorna le informazioni relative ad un testo

6.2.1.1 Endpoint

```
https://iibtel0n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/category/{Category}/{Title}/originalText
```

6.2.1.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione del testo di interesse
- Category
 - Categoria di traduzione
- Title
 - Titolo del testo

6.2.1.3 Esempio di richiesta

```
curl -X GET https://iibtel0n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/category/244de0d7-d3fb-42c6-9c04-f7e64f83f319/{Title}/originalText
```

Body

```
1 {
2   "Category":"Header",
3   "Text":"TestText edited text",
4   "Comment":"TestText edited comment",
5   "Link":"TestText edited link",
6   "Languages":["English", "Japanese"]
7 }
```

Immagine 14: Corpo della chiamata putOriginalText

6.2.1.4 Esempio di risposta

```

1  {
2    "result": {
3      "idTenant": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
4      "language": "Italian",
5      "category": {
6        "name": "Header",
7        "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
8      },
9      "title": "TestText",
10     "text": "TestText edited text",
11     "state": 0,
12     "comment": "TestText edited comment",
13     "link": "TestText edited link",
14     "feedback": null
15   }
16 }

```

Immagine 15: Esempio di risposta della chiamata putOriginalText

6.2.2 putTranslation

- Ritorna la lista delle lingue tradotte/da tradurre di un testo specifico

6.2.2.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/category/{Category}/{Title}/translation`

6.2.2.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione del testo di interesse
- Category
 - Categoria di traduzione
- Title
 - Titolo del testo

6.2.2.3 Esempio di richiesta

`curl -X PUT https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/category/244de0d7-d3fb-42c6-9c04-f7e64f83f319/TestText/translation`

Body

```

1  {
2      "Text": "Translation text for TestText in English"
3  }

```

Immagine 16: Corpo della chiamata putTranslation

6.2.2.4 Esempio di risposta

```

1  {
2      "result": {
3          "idTenant": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
4          "language": "English",
5          "category": {
6              "name": "Header",
7              "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
8          },
9          "title": "TestText",
10         "text": "Translation text for TestText in English",
11         "state": 2,
12         "comment": "TestText edited comment",
13         "link": "TestText edited link",
14         "feedback": null
15     }
16 }

```

Immagine 17: Esempio di risposta della chiamata putTranslation

6.2.3 putAcceptText

- Approva la traduzione di un testo

6.2.3.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/category/{Category}/{Title}/approveTranslation`

6.2.3.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione del testo di interesse
- Category
 - Categoria di traduzione

- Title
 - Titolo del testo

6.2.3.3 Esempio di richiesta

```
curl -X PUT https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/category/244de0d7-d3fb-42c6-9c04-f7e64f83f319/TestToBeVerifiedText/approveTranslation
```

6.2.3.4 Esempio di risposta



```
1  {
2    "message": "success"
3  }
```

Immagine 18: Esempio di risposta della chiamata putAcceptText

6.2.4 putRejectText

- Rifiuta la traduzione di un testo

6.2.4.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/category/{Category}/{Title}/rejectTranslation
```

6.2.4.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione del testo di interesse
- Category
 - Categoria di traduzione
- Title
 - Titolo del testo

6.2.4.3 Esempio di richiesta

```
curl -X PUT https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/category/244de0d7-d3fb-42c6-9c04-f7e64f83f319/TestToBeVerifiedText/rejectTranslation
```

Body

```

1  {
2      "Feedback": "Feedback for a rejection"
3  }

```

Immagine 19: Corpo

della chiamata

putRejectText

6.2.4.4 Esempio di risposta

```

1  {
2      "message": "success"
3  }

```

Immagine 20: Esempio di risposta della chiamata putRejectText

6.3 Lista di chiamate POST**6.3.1 addTenant**

- Crea un Tenant

6.3.1.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/create
```

6.3.1.2 Esempio di richiesta

```
curl -X POST https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/create
```

Body

```

1  {
2      "tenantName": "addTenantTest",
3      "defaultLanguage": "Italian",
4      "creationDate": 100000,
5      "languages": ["English", "Korean"],
6      "admins": ["a2893ce3-1db6-4677-9245-d22aa9ce8d20"],
7      "users": ["dac5d7aa-02d5-4b55-9aed-ddbc15bf6dc9"],
8      "categories": ["Header", "Footer", "Other"]
9  }

```

Immagine 21: Corpo della chiamata addTenant

6.3.1.3 Esempio di risposta

```

1  {
2    "Tenant": {
3      "tenantName": "addTenantTest",
4      "languages": [
5        "English",
6        "Korean"
7      ],
8      "defaultLanguage": "Italian",
9      "admins": ["a2893ce3-1db6-4677-9245-d22aa9ce8d20"],
10     "categories": [],
11     "creationDate": 1683843396.459,
12     "id": "5b516e73-0d9f-40bb-8fb8-b3dbd209939f",
13     "users": ["dac5d7aa-02d5-4b55-9aed-ddbc15bf6dc9"]
14   }
15 }

```

Immagine 22: Esempio di risposta della chiamata addTenant

6.3.2 addLanguage

- Aggiunge lingue ad un Tenant

6.3.2.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/addLanguage`

6.3.2.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.3.2.3 Esempio di richiesta

`curl -X POST https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/addLanguage`

Body

```

1  {
2    "Language": "Chinese"
3  }

```

Immagine 23: Corpo della chiamata addLanguage

6.3.2.4 Esempio di risposta

```

1  {
2      "tenant": [
3          "English",
4          "Japanese",
5          "Chinese"
6      ]
7  }
```

Immagine 24: Esempio di risposta della chiamata addLanguage

6.3.3 signUpUser

- Crea un utente associato ad un Tenant: il ruolo dipende dal campo “Group” passato nel body che può essere “admin” o “user”
- Il profilo dovrà essere verificato cliccando sul link ricevuto per email al momento della creazione

6.3.3.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/create/{TenantId}`

6.3.3.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.3.3.3 Esempio di richiesta

`curl -X POST https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/create/67faab0c-483f-44ed-a8d3-2cb9360d0f27`

Body

```

1  {
2      "Email": "testCreateUser@test.it",
3      "Password": "Test123.",
4      "Name": "TestName",
5      "Surname": "TestSurname",
6      "Group": "admin"
7  }
```

Immagine 25: Corpo della chiamata signUpUser

6.3.3.4 Esempio di risposta

```

1  {
2    "user": {
3      "Username": "81fd1abe-89ea-4c3b-beed-6a05a00deda0",
4      "UserAttributes": [
5        {
6          "Name": "custom:surname",
7          "Value": "TestSurname"
8        },
9        {
10         "Name": "sub",
11         "Value": "81fd1abe-89ea-4c3b-beed-6a05a00deda0"
12       },
13       {
14         "Name": "email_verified",
15         "Value": "false"
16       },
17       {
18         "Name": "name",
19         "Value": "TestName"
20       },
21       {
22         "Name": "email",
23         "Value": "testCreateUser@test.it"
24       }
25     ],
26     "UserCreateDate": "2023-05-11T22:38:00.394Z",
27     "UserLastModifiedDate": "2023-05-11T22:38:00.652Z",
28     "Enabled": true,
29     "UserStatus": "CONFIRMED"
30   }
31 }

```

Immagine 26: Esempio di risposta della chiamata signUpUser

6.3.4 postOriginalText

- Aggiunge un testo al Tenant

6.3.4.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/originalText`

6.3.4.2 Esempio di richiesta

`curl -X POST https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/originalText`

6.3.4.3 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

Body

```

1  {
2    "Title": "PostTextTest",
3    "Category": "Header",
4    "Text": "Text for PostTextTest",
5    "Comment": "Comment for PostTextTest",
6    "Link": "Link for PostTextTest",
7    "Languages": ["Japanese"]
8  }

```

Immagine 27: Corpo

signUpUser

della chiamata

6.3.4.4 Esempio di risposta

```

1  {
2    "result": "OK"
3  }

```

Immagine 28: Esempio di risposta della chiamata signUpUser

6.3.5 resetPassword

- Cambia la password di un utente

6.3.5.1 Endpoint

`https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/resetPassword`

6.3.5.2 Esempio di richiesta

`curl -X POST https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/resetPassword`

6.3.5.3 Parametri

Body

```

1  {
2      "Username": "adminDocAPI@sweg.it",
3      "Code": "123456",
4      "Password": "NewPassword123."
5  }
```

Immagine 29: Corpo della chiamata signUpUser

6.3.5.4 Esempio di risposta

```

1  {
2      "user": "Password resetted"
3  }
```

Immagine 29: Esempio di risposta della chiamata signUpUser

6.4 Lista di chiamate DELETE

6.4.1 deleteTenant

- Elimina un Tenant

6.4.1.1 Endpoint

```
https://iibtel10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}
```

6.4.1.2 Esempio di richiesta

```
curl -X DELETE https://iibtel10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27
```

6.4.1.3 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

6.4.1.4 Esempio di risposta

```
1 {
2   "tenants": "Tenant deleted"
3 }
```

Immagine 30: Esempio di risposta della chiamata deleteTenant

6.4.2 deleteLanguage

- Elimina una lingua dal Tenant e rispettive traduzioni

6.4.2.1 Endpoint

`https://iibtel10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/language/{Language}`

6.4.2.2 Esempio di richiesta

`curl -X DELETE https://iibtel10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/language/Chinese`

6.4.2.3 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua di traduzione dei testi associati

6.4.2.4 Esempio di risposta

```
1 {
2   "out": {
3     "tenantName": "DocAPI",
4     "languages": [
5       "English",
6       "Japanese"
7     ],
8     "defaultLanguage": "Italian",
9     "admins": [
10      "59e5680a-c781-4c92-b77e-47a00c62482e",
11      "453131f0-c061-449b-9388-bae39cfbba18"
12    ],
13    "categories": [
14      {
15        "name": "Header",
16        "id": "244de0d7-d3fb-42c6-9c04-f7e64f83f319"
17      }
18    ],
19    "creationDate": 1683821162.969,
20    "id": "67faab0c-483f-44ed-a8d3-2cb9360d0f27",
21    "users": [
22      "a216c5f4-c2cc-4db1-ad50-b70eb629f7f4",
23      "81fd1abe-89ea-4c3b-beed-6a05a00deda0"
24    ]
25  }
26 }
```


Immagine 31: Esempio di risposta della chiamata deleteLanguage

6.4.3 removeCategory

- Elimina una categoria dal Tenant e rispettivi testi

6.4.3.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/category
```

6.4.3.2 Esempio di richiesta

```
curl -X DELETE https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/67faab0c-483f-44ed-a8d3-2cb9360d0f27/d31243ce-567f-4039-9a9d-d74849451190/category
```

6.4.3.3 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Category
 - Categoria di traduzione dei testi associati

6.4.3.4 Esempio di risposta

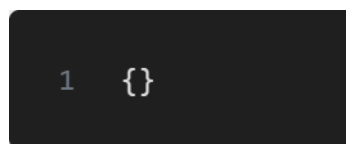


Immagine 32: Esempio di risposta della chiamata removeCategory

6.4.4 deleteUser

- Elimina un utente

6.4.4.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/{Username}/delete
```

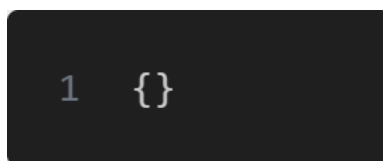
6.4.4.2 Esempio di richiesta

```
curl -X DELETE https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/user/81fd1abe-89ea-4c3b-beed-6a05a00deda0/delete
```

6.4.4.3 Parametri

- Username
 - Id utente da cancellare

6.4.4.4 Esempio di risposta



```
1 {}
```

Immagine 33: Esempio di risposta della chiamata deleteUser

6.4.5 deleteOriginalText

- Elimina un testo dal Tenant

6.4.5.1 Endpoint

```
https://iibtel0n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/category/{Category}/{Title}/originalText
```

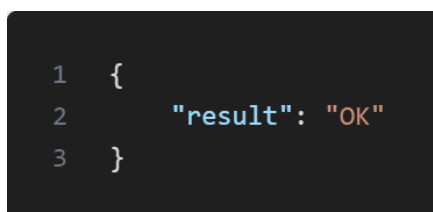
6.4.5.2 Esempio di richiesta

```
curl -X DELETE https://iibtel0n05.execute-api.eu-west-2.amazonaws.com/dev/text/67faab0c-483f-44ed-a8d3-2cb9360d0f27/category/244de0d7-d3fb-42c6-9c04-f7e64f83f319/PostTextTest/originalText
```

6.4.5.3 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Category
 - Categoria di traduzione
- Title
 - Titolo del testo

6.4.5.4 Esempio di risposta



```
1 {
2   "result": "OK"
3 }
```

Immagine 34: Esempio di

deleteOriginalText

risposta della chiamata

7 Documentazione libreria client

La seguente sezione descrive dettagliatamente la libreria client creata dal gruppo *SWEG*, che permette agli sviluppatori di un tenant di recuperare i testi gestiti nel backoffice ed utilizzarli all'interno di pagine web.

Viene fornita una panoramica completa delle funzioni disponibili, delle API implementate per permetterne il funzionamento e qualche esempio di utilizzo pratico della libreria stessa.

7.1 Struttura libreria

La libreria consiste in un singolo file .js che esporta un oggetto, denominato *Translatify*, da istanziare dovunque si voglia utilizzarla.

Questo oggetto possiede proprietà e funzioni (metodi) pubbliche e funzioni private, che non vengono esportate, che analizzeremo di seguito.

7.1.1 Funzioni non esportate

Le seguenti funzioni vengono utilizzate dai metodi dell'oggetto per svolgere alcune operazioni: essendo queste non esportate dal modulo della libreria non è possibile richiamarle, si comportano quindi come metodi definiti privati in un ambiente sviluppato con un linguaggio di programmazione ad oggetti.

7.1.1.1 findElementsMatchingIdPattern()

Questa funzione permette di trovare tutti gli elementi presenti sulla pagina web che possiedono un attributo `"id"` che contenga un determinato pattern nella forma

`"category:NomeCategoria&title:TitoloTesto"`.

Da questo pattern si ricavano il `categoryName` e il `title` di un testo, dati utilizzati in seguito per recuperare il testo corretto da inserire nell'elemento HTML.

Ritorna un array di elementi HTML (es.: `div`, `h1`, `p`, `button`,...) con id che rispettano il pattern.

```

3 //returns an array of elements that contain an id in the form of "category:XXX&title:XXX"
4 function findElementsMatchingIdPattern() {
5     // Define the regular expression pattern to match
6     const idPattern = /category:(.*?)&title:(.+)/;
7
8     // Get all HTML elements on the page
9     const allElements = document.getElementsByTagName('*');
10
11     // Create an array to store matching elements
12     const matchingElements = [];
13
14     // Loop through all elements and check if their ID matches the pattern
15     for (let i = 0; i < allElements.length; i++) {
16         const element = allElements[i];
17         const elementId = element.getAttribute('id');
18         if (elementId && elementId.match(idPattern)) {
19             const matches = elementId.match(idPattern);
20             const category = matches[1];
21             const title = matches[2];
22             matchingElements.push({ element, category, title });
23         }
24     }
25
26     // Return the array of matching elements
27     return matchingElements;
28 }

```

Immagine 35: Codice funzione findElementsMatchingIdPattern()

7.1.1.2 addTextToElement(element, text)

Questa funzione semplicemente aggiunge un testo ad un elemento HTML, entrambi passati come parametri alla funzione stessa,

Si è deciso di creare questa funzione, seppur il contenuto sia basilare, per semplificare la lettura del codice dei metodi dell'oggetto Translatify e permettere di apportare modifiche in un singolo punto del codice in futuro se necessario.

```

30 // Add some text to an HTML element
31 function addTextToMatchingElement(element, text) {
32     element.textContent = text;
33 }

```

Immagine 36: Codice funzione addTextToElement()

7.1.2 Oggetto esportato : Translatify

Si tratta dell'elemento principale della libreria: esso possiede 5 attributi, 3 metodi più un costruttore.

Gli attributi sono i seguenti:

- **tenantId** : stringa che rappresenta l'id del tenant, utilizzato per le chiamate API
- **apiKey** : stringa che contiene un valore alfanumerico da utilizzare negli headers delle chiamate API

- language: stringa che rappresenta la lingua corrente da utilizzare
- languages: array di stringhe che contiene tutte le lingue presenti all'interno del tenant indicato da tenantId
- textHolders: array di elementi HTML

Di seguito vedremo come questi valori vengono utilizzati all'interno dei vari metodi e come questi funzionano.

```

34
35  export default class Translatify{
36      tenantId;
37      apiKey;
38      language;
39      languages;
40      textHolders;
41
42  >  constructor(tenantId, apiKey){ ...
45      }
46
47      //get languages from API and sets object's languages array
48  >  async retrieveLanguages(){ ...
64      };
65
66  >  //get specified text...
69  >  getText(categoryName, title){ ...
89      };
90
91  >  //sets object language and get all texts that match the
92      //category-title combination from all the components in the page
93  >  handleLanguageChange(newLanguage){ ...
108      };
109  }

```

Immagine 37: Struttura oggetto Translatify

7.1.3 Metodi Oggetto

7.1.3.1 constructor(tenantId, apiKey)

Una volta importata la libreria sarà necessario istanziare un nuovo oggetto Translatify chiamando il costruttore, al quale è necessario fornire l'id di un tenant e una chiave API come parametri.

Lo facciamo chiamando questa funzione nel seguente modo: **new Translatify(ID_DEL_TUO_TENANT, Chiave_API)** ed assegnandolo ad una variabile.

```

42  constructor(tenantId, apiKey){
43      this.tenantId = tenantId;
44      this.apiKey = apiKey;
45  }

```

Immagine 38: Codice costruttore oggetto Translatify

7.1.3.2 async retrieveLanguages()

Questo metodo permette di ottenere dal database le lingue del tenant tramite una chiamata API e le salva all'interno della proprietà *languages* dell'oggetto stesso.

Questa funzionalità è stata separata dal costruttore per permettere agli utenti sviluppatori di avere maggiore controllo sull'oggetto Translatify.

Normalmente è immaginata per essere la prima funzione chiamata dopo aver istanziato l'oggetto.

```

47  //get languages from API and sets object's languages array
48  async retrieveLanguages(){
49      this.languages = await fetch(`${REACT_APP_API_KEY}/tenant/${this.tenantId}/languages`,
50      {
51          headers: {
52              'Content-Type': 'application/json',
53              'Accept': 'application/json',
54              'x-api-key': this.apiKey,
55          })
56      .then(res => res.json())
57      .then(res => res.languages)
58      .catch(err => {console.log('something went wrong while fetching data')});
59  };

```

Immagine 39: Codice metodo retrieveLanguages()

7.1.3.3 getText(categoryName, title)

Questo metodo è la parte centrale dell'oggetto e della libreria: passando come attributi il nome di una categoria e il titolo di un Testo, tramite chiamata API, ottiene e ritorna la stringa di testo del Testo indicato.

La lingua da utilizzare nella chiamata API è la lingua salvata all'interno della proprietà "language" dell'oggetto stesso.

La stringa ritornata potrà avere tre valori diversi:

- Se il testo esiste nella lingua selezionata, ritorna quel testo
- Se il testo non esiste nella lingua selezionata, ma esiste nella lingua di default^G del tenant, ritorna il testo nella lingua di default
- Se il testo non esiste in nessuna lingua ritorna una stringa nulla.

```

64  getText(categoryName, title){
65      if(!this.language || this.language === ''){
66          throw 'no language selected';
67      }
68      return fetch(`${REACT_APP_API_KEY}/text/${this.tenantId}/${this.language}/${categoryName}/${title}`,
69      {
70          headers: {
71              'Content-Type': 'application/json',
72              'Accept': 'application/json',
73              'x-api-key': this.apiKey,
74          })
75      .then(res => res.json())
76      .then(res => {
77          if(res.Text) return res.Text.text;
78          else return '';
79      })
80      .catch(err => {throw err});
81  };

```

Immagine 40: Codice metodo getText(categoryName, title)

7.1.3.4 handleLanguageChange(newLanguage)

Questo metodo permette di impostare la proprietà “language” dell’oggetto con un nuovo valore passato come parametro.

Fatto questo, svolge tutto il necessario per richiamare tutti i testi necessari a popolare la pagina, chiamando gli altri metodi e le altre funzioni, e popolarla.

Notiamo che la funzione “findElementsMatchingIdPattern” viene chiamata solo la prima volta per salvare all’interno della proprietà “textHolders” l’array di elementi HTML da riempire: dalla seconda in poi questo non sarà più necessario in quanto questa proprietà sarà già definita.

```

86  handleLanguageChange(newLanguage){
87      this.language = newLanguage;
88      if(!this.textHolders || this.textHolders.length === 0)
89          this.textHolders = findElementsMatchingIdPattern();
90      this.textHolders.forEach((item) => {
91          this.getText(item.category, item.title)
92          .then((text) =>{
93              addTextToElement(item.element, text);
94          })
95          .catch(err =>{
96              console.log(err);
97          })
98      });
99  };

```

Immagine 41: Codice metodo handleLanguageChange(newLanguage)

7.2 Chiamate API utilizzate

La libreria necessita di un alcune API apposite per poter svolgere le funzioni mostrate precedentemente: l'unica differenza tra queste API e quelle sviluppate per il backoffice sta nel tipo di autorizzazione utilizzata per garantire l'accesso e l'utilizzo: le API per il backoffice richiedono un idToken di un utente registrato su Amazon Cognito, mentre le seguenti API utilizzano un api-key univoca per il tenant associato fornita da AWS.

7.2.1 getLanguages

- Ritorna la lista delle lingue di un tenant specifico, che comprende sia le liste secondarie che quella di default

7.2.1.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/{TenantId}/languages
```

7.2.1.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento

7.2.1.3 Esempio di richiesta

```
curl -X GET https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/tenant/02d34632-83db-4ab6-b00a-56d3f224bb62/languages
```

7.2.1.4 Esempio di risposta

```
1  {
2      "languages": [
3          "English",
4          "Chinese",
5          "Italian"
6      ]
7  }
```

Immagine 42: Esempio di risposta della chiamata getLanguages

7.2.2 getText

- Ritorna la lista delle lingue di un tenant specifico, che comprende sia le liste secondarie che quella di default

7.2.2.1 Endpoint

```
https://iibte10n05.execute-api.eu-west-2.amazonaws.com/dev/text/{TenantId}/{Language}/{CategoryName}/{Title}
```


7.2.2.2 Parametri

- TenantId
 - Identificativo del Tenant di riferimento
- Language
 - Lingua del testo cercato
- CategoryName
 - Nome della categoria del testo cercato
- Title
 - Titolo del testo cercato

7.2.2.3 Esempio di richiesta

```
curl -X GET https://iibtel10n05.execute-api.eu-west-2.amazonaws.com/dev/text/02d34632-83db-4ab6-b00a-56d3f224bb62/English/Main/PrimoParagrafo
```

7.2.2.4 Esempio di risposta

```

1  {
2      "Text": {
3          "idTenant": "02d34632-83db-4ab6-b00a-56d3f224bb62",
4          "language": "English",
5          "category": {
6              "name": "Main",
7              "id": "9a4cccb-7eec-46ce-ae58-f29b666896bc"
8          },
9          "title": "PrimoParagrafo",
10         "text": "This is the first paragraph...",
11         "state": 3,
12         "comment": "-",
13         "link": "-",
14         "feedback": "null"
15     }
16 }
```

Immagine 43: Esempio di risposta della chiamata getText

7.3 Esempio di utilizzo

Forniamo ora un mini-guida di esempio per un modo di utilizzare la libreria.

Specifichiamo che non c'è un unico modo per utilizzarla, ma questo che presentiamo è il modo in cui noi l'abbiamo utilizzata per realizzare una nostra demo.

7.3.1 Codice HTML

È necessario che tutti gli elementi che vogliamo siano “dinamici”, ovvero il cui testo dipenda dalla lingua selezionata, posseggano un id come mostrato in figura e come spiegato precedentemente.

Si consiglia di creare questi elementi vuoti, senza nessun testo preimpostato al loro interno: questo andrebbe contro la logica dell'utilizzo della nostra libreria.

Nel codice HTML non c'è altro da aggiungere, se non importare i file di script che utilizzano la nostra libreria.

```
89 <h1 id="category:NomeCategoria&title:Titolo"></h1>
```

Immagine 44: Esempio di ID per un elemento HTML

7.3.2 Script di esempio per una pagina interattiva

```
1 import Translatify from "../Translatify-SDK/index.js";
```

Immagine 45: Importare libreria in un file JS/TS

Per dimostrare le funzionalità della libreria, possiamo utilizzare un picker creato usando una `<select>` con varie `<option>` al suo interno, ognuna che specifica una lingua.

Recuperiamo quindi l'elemento `<select>` e creiamo una funzione per gestire il cambiamento della lingua (ovvero del valore presente sulla `<select>`): per comodità salviamo l'ultimo valore scelto all'interno del `localStorage`.

```
3 //setup picker
4 const picker = document.getElementById("language-picker"); //get the picker element
5 const setLanguage = () => {
6   var selectedValue = picker.value; //get the value from the picker
7   localStorage.setItem("language", selectedValue);
8   translatify.handleLanguageChange(picker.value); //loads texts in new language
9 }
10
```

Immagine 46: Esempio di come impostare un picker ed una funzione da utilizzare per gestire il cambio lingua da utilizzare con la libreria

Creiamo un'istanza di un oggetto `Translatify` e, come spiegato precedentemente, chiamiamo subito il metodo `retrieveLanguages()` per settare le lingue tra cui scegliere in seguito.

In questo caso mostriamo anche come salvare l'istanza dell'oggetto in `localStorage`, se dovesse servire per casi particolari da noi non trattati.

```

11 //setup Translatify
12 let translatify;
13 //example on how to store and retrieve translatify from localStorage if it becomes necessary
14 if(localStorage.getItem('translatify')){
15     const localStorageData = JSON.parse(localStorage.getItem('translatify'));
16     translatify = new Translatify(localStorageData.tenantId, localStorageData.apiKey);
17 }
18 else{
19     translatify = new Translatify('YOUR_TENANT_ID', 'YOUR_API_KEY');
20     localStorage.setItem('translatify', JSON.stringify(translatify));
21 }
22 await translatify.retrieveLanguages();
23

```

Immagine 47: Esempio di come impostare Translatify e gestirlo con il localStorage

Gestiamo il salvataggio della lingua corrente separatamente dall'oggetto Translatify anch'esso in localStorage: questo permette di utilizzare lingue non presenti all'interno dell'array "languages" di Translatify.

Se nessuna lingua è presente in localStorage, viene impostata con la prima lingua presente nell'array "languages" di Translatify e poi viene gestito il caricamento di tutti i testi nella nuova lingua.

Se una lingua era già presente nel localStorage, il caricamento di tutti i testi avviene con questa.

```

25 //setup current language and load texts for the first time
26 let lang = localStorage.getItem("language");
27 if(!lang){
28     if(translatify.languages){
29         lang = translatify.languages[0];
30         translatify.handleLanguageChange(lang);
31         localStorage.setItem("language", lang);
32     }
33 }
34 else{
35     translatify.handleLanguageChange(lang);
36 }

```

Immagine 48: Esempio di come impostare la lingua da utilizzare sull'oggetto Translatify al caricamento e salvarla in localStorage

Infine non resta che impostare il picker in modo che possenga le lingue presenti nell'oggetto Translatify e che possa gestire il cambiamento della lingua selezionata per effettuare il cambio di lingua dei testi degli elementi HTML.

```

37
38 picker.addEventListener('change', setLanguage);
39
40 if(translatify.languages && translatify.languages.length > 0){
41     translatify.languages.forEach(lang => {
42         // create option using DOM
43         const newOption = document.createElement('option');
44         const optionText = document.createTextNode(lang);
45         // set option text
46         newOption.appendChild(optionText);
47         // and option value
48         newOption.setAttribute('value', lang);
49         picker.appendChild(newOption);
50     });
51 }
52
53 picker.value = lang; //setup picker
54

```

Immagine 49: Esempio di come impostare la lingua da utilizzare sul picker al caricamento e le options del picker

7.3.3 Risultato finale

Abbiamo così ottenuto una pagina web dinamica che permette all'utente di selezionare tra lingue diverse una lingua in cui visualizzare il contenuto del sito.

Di seguito mostriamo la pagina della nostra demo che visualizza i testi in Italiano, Inglese e Cinese.

Notiamo come i testi riflettono il contenuto del database, come specificato al punto 7.1.3.3: i testi mancanti nella lingua selezionata riflettono la lingua di default del tenant mentre i testi non presenti all'interno del tenant non mostrano niente .



Immagine 50: Demo con lingua impostata in Italiano

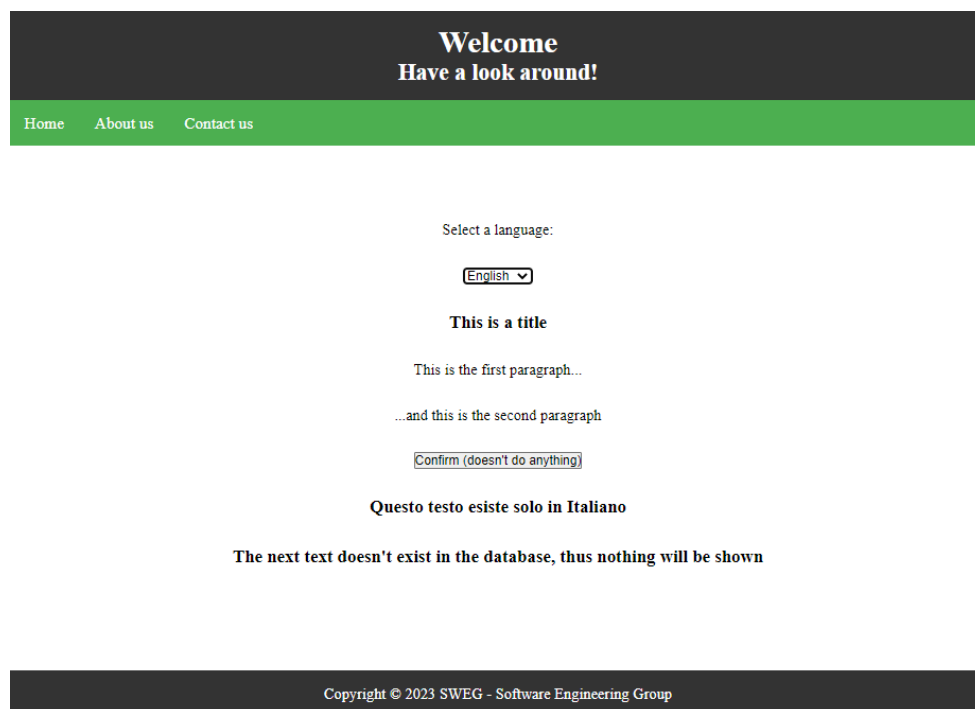


Immagine 51: Demo con lingua impostata in Inglese



Immagine 52: Demo con lingua impostata in Cinese

8 Punti di estensione

Il prodotto *Translatify* è pensato per poter essere estensibile e manutenibile, data la struttura del codice realizzato, evitando di poter attuare estensioni in futuro. Nella successiva sezione, vengono descritti i punti del codice in cui gli sviluppatori possono aggiungere funzionalità oppure modificare il comportamento del software senza dover modificare il codice sorgente esistente. I punti descritti sono progettati per essere facilmente estendibili e modulari e qui ne verrà fornita una panoramica completa.

8.1 Aggiunta di nuovi ruoli per utente

La componente grafica *UserMenu* consente di stabilire precisamente il tipo di utente presente e, in base al ruolo, permette di visualizzare voci di menù differenti nella barra di navigazione individuata. Il componente è poi incapsulato all'interno di *LayoutWrapper*, che permette la definizione delle opzioni di navigazione in base al tipo di utente riconosciuto.

Come visibile dall'immagine seguente, essa è localizzata in



```

TS UserMenu.tsx X
src > components > LayoutWrapper > UserMenu > TS UserMenu.tsx > [0] a
18   label: string;
19   to: string;
20 }
21
22 const contentUserLinks: Link[] = [
23   { label: "User", to: "/User" },
24   { label: "Tenant Texts", to: "/TenantTexts" },
25 ];
26
27 const adminLinks: Link[] = [
28   { label: "Admin", to: "/Admin" },
29   { label: "User", to: "/User" },
30   { label: "Review Texts", to: "/ReviewTexts" },
31   { label: "Tenant Texts", to: "/TenantTexts" },
32   { label: "Create User", to: "/CreateUser" },
33   { label: "Tenant Settings", to: "/TenantSettings" },
34   { label: "Tenant Categories", to: "/TenantTextCat"
35 ];
36
37 const superAdminLinks: Link[] = [
38   { label: "Super Admin", to: "/SuperAdmin" },
39   { label: "Create Tenant", to: "/CreateTenant" },
40 ];
41
42 interface UserMenuProps{
43   userType: string | null;
44 }

```

src/components/UserMenu/UserMenu.tsx.

Immagine 53: Link di navigazione forniti per utente

Nelle singole pagine, poi, una volta definiti i link di navigazione e aggiunto un nuovo tipo^G corrispondente al ruolo dell'utente, sarà sufficiente definire le aree a cui è permessa la navigazione all'utente (attraverso la componente *PrivateRoute*), permettendo la navigazione solo ad una lista di utenti specifica, come segue:

```
return (
  <PrivateRoute allowedUsers={['admin']}>
    <LayoutWrapper userType="admin">
      <Grid container spacing={3} justifyConte
        <Grid item xs={12}>
          <PageTitle title='Admin Dashboard' />
        </Grid>
      </Grid>
    </PrivateRoute>
  </Grid>
);
```

Immagine 54: Utenti a cui è permessa la navigazione

Sostituendo opportunamente il tipo di utente, sarà possibile in modo semplice e scalabile creare nuovi ruoli utente, una volta creato l'opportuno tipo di riferimento.

8.2 Aggiunta di funzionalità di ricerca e di filtro avanzate

Le funzionalità di filtro e ricerca sono realizzate rispettivamente da due componenti, cioè *SearchBox* e *Picker*. Nello specifico, per la prima delle due, disponibile al percorso *src/components/SearchBox/SearchBox.tsx*, viene offerta una funzionalità di ricerca basata su quanto è richiesto dal componente che la contiene:

```
interface SearchBoxProps {
  handleParentSearch : (query : string) => void;
}

export default function SearchBox({handleParentSearch} : SearchBoxProps) {
  const [query, setQuery] = useState<string>('');

  const handleSearch = () => {
    if(query) handleParentSearch(query);
    else handleParentSearch('');
  };

  const handleKeyPress = (event: React.KeyboardEvent<HTMLDivElement>) => {
    if (event.key === 'Enter') {
      handleSearch();
    }
  };
}
```

Immagine 55: Funzionalità di ricerca offerte

Sarebbe quindi possibile utilizzare questo componente per effettuare qualsiasi tipo di ricerca sia necessario, passando come Prop una funzione che ne specifichi il comportamento:

Per la seconda delle due componenti, disponibile al percorso `src/components/Picker/Picker.tsx`, viene offerta una funzionalità di selezione e di filtraggio della stessa basata su un evento⁶ definito in base al contesto:

```
interface PickerProps {
  id: string;
  value: string | null;
  onChange: (value : string) => void;
  choices: string[];
  onClear: () => void;
}

export default function Picker({ id, value, onChange, choices, onClear }: PickerProps) {
  const handleChange = (newValue: string | null) => {
    newValue ? onChange(newValue) : onClear();
  };

  return (
    <>
      <Autocomplete
        id={id}
        options={choices}
        value={value}
        onChange={(event, newValue: string | null) => handleChange(newValue)}
        renderInput={({params}) => <TextField {...params} label={id} fullWidth />}
      />
    </>
  );
}
```

Immagine 56: Codice del componente Picker

Nelle pagine in cui viene utilizzato, viene infatti definito uno stato che permette di definire come viene realizzato il filtro. L'esempio successivo è tratto dalla pagina *TenantTextsView*, che permette di visualizzare i testi del Tenant, filtrandoli in base al loro stato:

```
<Picker
  id={"Choose state to filter"}
  value={pickedTextState}
  onChange={handleTextStateChange}
  choices={textStates}
  onClear = {handleClearTextState}
/>
```

Immagine 57: Esempio di utilizzo del componente Picker

Come evidenziato, è quindi facilmente possibile implementare nuove funzionalità di ricerca semplicemente definendosi una funzione apposita, in base alle proprie esigenze di implementazione.