Componenti:

Generiche - Tutti gli Utenti

* Login
  + Username (campo della tabella)
  + Password (campo della tabella)
* Componente search box

SuperAdmin

* Form creazione Tenant
  + Nome
  + Admin
  + Lingua di default
* Elemento per lista singolo Tenant
  + Nome Tenant
  + Admin associati
* Lista Tenant esistenti
* Language Picker completo (vede e seleziona)
  + Menù a tendina, ogni elemento è una lingua
* Pagina dati singolo tenant

Admin

* Language Picker lingue specifiche (vede e seleziona)
  + Menù a tendina, ogni elemento è una lingua secondaria del Tenant
* Form creazione utente
  + Nome
  + Cognome
  + Username
  + Email
  + Ruolo
  + Password (temporanea, random dal sistema)
* Page creazione testo originale
  + Testo
  + Commento
  + Link esterno
  + Lingue in cui tradurlo (default: tutte; selezionabile quali scegliere)
* Pagina traduzioni da approvare
  + Approvazione
  + Rifiuto
* ~~Modal (?) creazione categoria traduzione~~
  + ~~Nome categoria (la categoria è per tutte le lingue)~~
* Lista testi originali esistenti (tutti i testi presenti)
* Elemento per lista singolo testo originale (con funzioni da admin)
* Lista traduzioni approvate
* Elemento per lista traduzione approvata (con funzioni da admin)
* Lista testi non ancora tradotti
* Elemento per lista singolo testo non tradotto (con funzioni da admin)
* Form inserimento traduzione (lo fa anche l’admin)

Dopo che i test vanno bene sui componenti, si creano le pagine.

User

* Elemento per lista singolo testo originale
* Lista testi originali esistenti (tutti i testi presenti)
* Lista traduzioni approvate
* Elemento per lista traduzione approvata
* Lista testi non ancora tradotti
* Elemento per lista singolo testo non tradotto
* Language Picker lingue specifiche (vede e seleziona, per cambiare lingua in cui tradurre)
  + Menù a tendina, ogni elemento è una lingua secondaria del Tenant
* Form inserimento traduzione

**API**

Chiamate generali:

GET:

* getAllTenants → body : [{nome:string, id:string}]
* {TenantId}/tenantInfo → tenant :Tenant
* {TenantId}/defaultLanguage → defaultLanguage : String
* {TenantId}/secondaryLanguages → languages : String[]
* {TenantId}/languages → languages : String[]
* **{TenantId}/tokens → tokens : Token[]**
* **{TenantId}/allUsersNames → users : String[]**
* {TenantId}/contentUsersNames → contentUsers : String[]
* {TenantId}/adminUsersNames → admins : String[]
* {TenantId}/{UserId}/userInfo → users : User[](tutto tranne la password)
* {TenantId}/allTexts → texts: textCategory[]
* {TenantId}/{category}/textsFromCategory → textCategory[]
* {TenantId}/{language}/textsFromLanguage → textCategory[] (ritorna solo i testi accettati)
* {TenantId}/{language}/{category}/texts → Text[]
* {TenantId}/originalTexts → textCategory[] (come old text, ma i testi nella lingua di default)
* {TenantId}/allCategories → String[]
* {TenantId}/{language}/rejectedTexts → Text[] (prima chiama questa nella pagina delle traduzioni da svolgere, e se non ritorna niente allora chiama untranslatedTexts)
* {TenantId}/{language}/untranslatedTexts → Text[]
* {TenantId}/{language}/pendingTranslations →Text[]
* {TenantId}/{language}/{id}/searchTextsById → Text[] / textGroup[] ? (un mix dei due forse)
* {TenantId}/{language}/{category}/{id}/searchTextById : → Text

DELETE:

* **{Username}/user**
* {TenantId}/tenant
* {TenantId}/{language}/language{non puoi cancellare la lingua originale}
* {TenantId}/{TextId}/text (da valutare se i testi da non tradurre NON sono presenti nel DB o se SONO presenti nel db ma con uno stato apposito; capire anche se differenziare la “delete” di un testo originale da una traduzione)//non ci sarà differenza tra originale e tradotto. la cancellazione di un TextId comporta la cancellazione del testo in TUTTE le lingue.
* {TenantId}/{TokenId}/token
* {TenantId}/{CategoryId}/category (cancella anche tutti i testi al suo interno, in tutte le lingue)

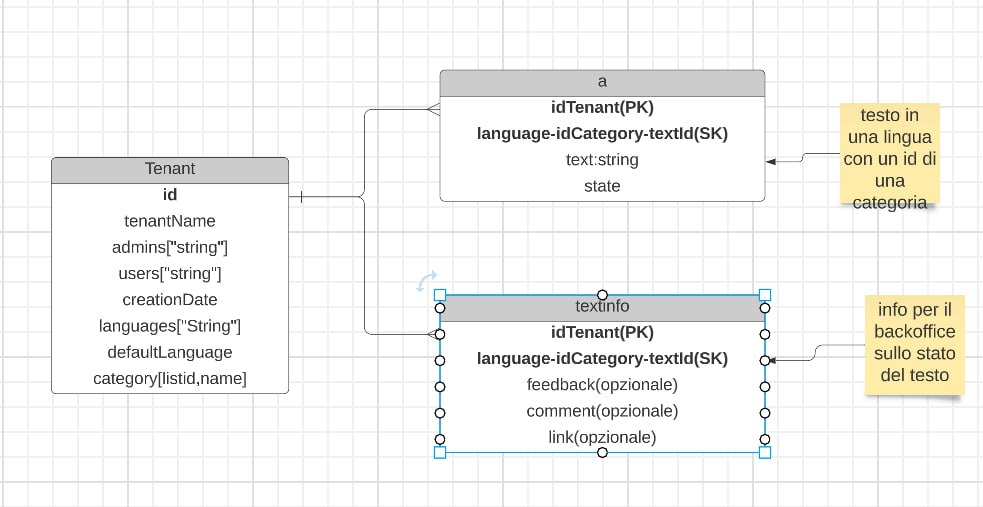
POST (/ : “body” → ritorno):

* **/user: {username, password, email, role, name, lastname} → User**
* /tenant: {tenantName, defaultLanguage, adminUsername} → Tenant
* {TenantId}/originalText: {Text, categoryId, comment, link : String} → Text//l’inserimento che causa la creazione di tutte le traduzioni deve essere una transazione? cosa deve accadere se una traduzione fallisce nella creazione?
* {TenantId}/translation: {Text, categoryId, language : String} → Text
* {TenantId}/category: {String} → categoryId : String//ha senso generare una categoria?
* {TenantId}/generateToken: null → Token

PUT:

* {TenantId}/reset (rimuove tutto dal tenant e rimane vuoto)
* /language: {language : String} → language : String (crea la lingua)
* {TenantId}/{CategoryId}/category: {Text} → categoryId : String (modifica il nome della categoria)
* {TenantId}/{TextId}/originalText : {Text, categoryId, comment, link : String} → Text
* {TenantId}/{TextId}/translation : {Text, categoryId, language, feedback, : String, feedback: Enum} → Text
* {TenantId}/{CategoryId}/{Language}/{TextId}/approveText : null → Text / null
* {TenantId}/{CategoryId}/{Language}/{TextId}/rejectText : {feedback : String} → Text / null

PLAMA key like:



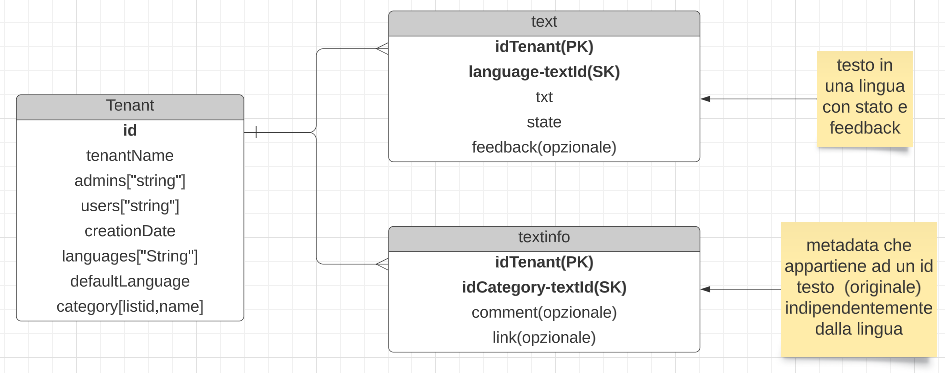
Pregi:

* separazione delle informazioni da uso normale dei testi dalle info extra necessarie solo al backend e la fase di traduzione;
* forse più facile da sviluppare;

Problema della chiave language-category-id:

* la categoria si ripete per ogni singolo testo X in ogni singola lingua Y (ridondanza dell’informazione Y, non impatta la performance) ;
* le info del testo si ripetono per ogni lingua Y anziché essere univoche in base all'ID testo X (ridondanza delle righe nella tabella textinfo Y, impatta la performance);
* un ID testo può avere più categorie assegnate, necessità di controlli aggiuntivi;
* necessario specificare la categoria quando il testo viene usato in un sito web;
* la riassegnazione della categoria ai testi è molto pesante, come nel caso della cancellazione.  
  La categoria è chiave primaria e non può essere modificata, quindi è necessario cancellare e riscrivere l’intera riga fatta manualmente.   
  La cancellazione di una categoria e ritorno di tutti i suoi testi alla categoria di default Main richiede la modifica di entrambe le tabelle textinfo e text (2), riscrivendo singola riga per riga. è necessario riscrivere ogni testo X appartenente a quella categoria per tutte le lingue Y. Costo: X\*Y\*2\*latenza\_singola\_richiesta  
    
  Es. 150Testi \* 15Lingue \* 2Tabelle \* 20ms = 90.000ms = 90s  
    
  Utilizzando BatchWrite è possibile accorpare più richieste di scrittura in una per renderla più efficiente. Tuttavia gestisce fino a 100 voci e fino a 16MB. Difficile tenere traccia del peso delle richieste data la grande e variabile dimensione dei testi. Costo: (X\*Y)/100\*2\*latenza\_batch  
    
  Stima: (150Batch \* 15Lingue)/100 \* 2 Tabelle \* 150ms(?) = 6.900ms = 6.9s;

MILO tag like:



Pregi:

* separazione delle informazioni da uso normale dei testi dalle info extra necessarie solo al backend e la fase di traduzione;
* separazione tra lingua e categoria nella chiave primaria;
* L’ID testo è univoco fra tutti i testi anziché solo all’interno di una categoria. Non è necessaria la categoria per il normale uso da sito web;
* la tabella textinfo utilizza solo una riga per testo;
* Modifica della categoria più efficiente rispetto a Plama. La riscrittura della categoria non intacca la tabella dei testi ma solo info. L'iterazione si ripete solo per l’ID dei testi.  
  Costo: X \* latenza\_singola\_richiesta.

Es. 150Testi \* 20ms = 3.000ms = 3s  
Con BatchWrite:  
Stima: 2Batch \* 150ms = 300ms = 0.3s

Problema delle chiavi language-text category-text:

* un ID testo può avere più categorie assegnate, necessità di controlli aggiuntivi;
* lo sviluppo delle funzioni potrebbe essere più difficile con maggiori controlli(?), tuttavia nello sviluppo delle mie funzioni non ho incontrato difficoltà;
* quando si devono associare la tabella testi con info tipo nella ricerca per categoria, non avendo la tabella text una categoria e non avendo la possibilità di fare una GET con una lista di ID testi, è necessario ottenere tutti i testi e manualmente iterare per tutti gli id dei testi nella categoria e ricercare fra tutti i testi l’equivalente campo info.  
  Può causare inefficienza. Tuttavia questo avviene principalmente quando i testi vengono cercati con filtro e il numero delle iterazioni dovrebbe essere limitato.

La velocità di interazione sul server con una sola chiamata dal db per ottenere tutti i dati dovrebbe essere comunque abbastanza superiore rispetto a fare tante chiamate per la scrittura al database.  
Possibile soluzione:  
aggiungere categoria nella tabella testi come campo normale anziché chiave. Crea ridondanza.