Relazione Progetto Advanced Programming Languages

Manlio Puglisi – O55000433;

Repository Progetto:

https://github.com/PManlio/AdvancedProgrammingLanguagesProject

1 - Obiettivi del Progetto

L’obiettivo del progetto consiste nello sviluppo di un servizio di supporto per psicologi e rispettivi pazienti. Una delle attività che può essere assegnata da uno psicologo al proprio paziente consiste nello scrivere una sorta di diario in cui scrivere i propri pensieri e le proprie riflessioni, per poi essere letto dallo psicologo. Nel particolare, i pazienti che sono registrati potranno quindi cercare gli psicologi di cui sono pazienti e inviare loro i testi che scrivono come se si trattasse di una sorta di blog personale (leggibile solo dallo psicologo a cui tali testi vengono inviati e dall’utente che li scrive). I testi che vengono scritti vengono poi processati da un’intelligenza artificiale che ne effettua la sentiment analysis, in modo da aiutare lo psicologo a capire in anticipo, rispetto alla terapia di presenza, se il proprio paziente stia scrivendo testi tendenzialmente positivi o negativi ed essere così in grado di migliorare le sedute in relazione a quanto letto.

2 - Linguaggi di programmazione utilizzati

- **Go**, di seguito i package esterni aggiuntivi:
  - **Gorilla/Mux** [https://github.com/gorilla/mux] che permette una maggiore flessibilità per la scrittura di applicazioni REST, in particolare per la stesura dei router e per la lettura dei campi del protocollo http;
  - **Gorilla/Handlers** [https://github.com/gorilla/handlers] per la migliore gestione dei middleware;
  - **Go-SQL-Driver** [https://github.com/go-sql-driver/mysql] che gestisce automaticamente la connection-pool al database;
  - **GoDotEnv** [https://github.com/joho/godotenv] che semplifica il parsing di file .env;

- **Python**, di seguito i package aggiuntivi:
  - **TextBlob** [https://textblob.readthedocs.io/en/dev/] per la sentiment analysis;
  - **FastAPI** [https://fastapi.tiangolo.com/] per il server REST in cui salvare i documenti;
  - **PyMongo** [https://pymongo.readthedocs.io/en/stable/] libreria per implementare un client per la comunicazione con un database MongoDB;
  - **Googletrans** [https://pypi.org/project/googletrans/] per la traduzione del testo in inglese per permettere a TextBlob una maggiore accuratezza.
  - **Python-dotenv** [https://pypi.org/project/python-dotenv/] per semplificare la lettura del file .env

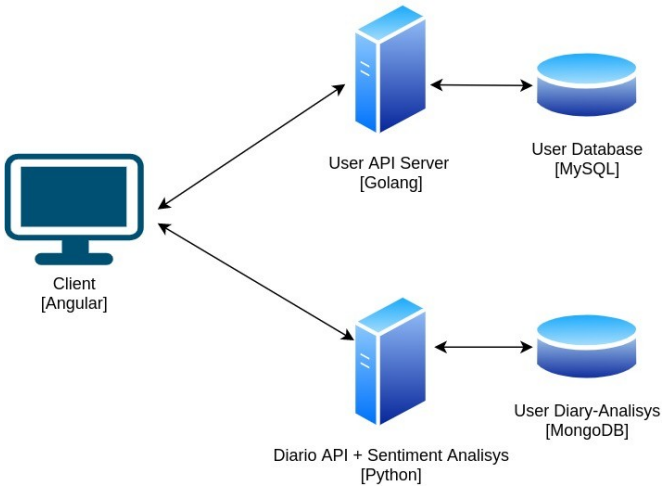
- TypeScript (per frontend, non oggetto d’esame):
  - Angular (v. 10+)

A questi si aggiunge l'utilizzo dei database MySQL per tenere traccia degli utenti (pazienti e psicologi) e MongoDB per salvare i documenti scritti dai pazienti, di cui bisognerà effettuare la sentiment analysis.

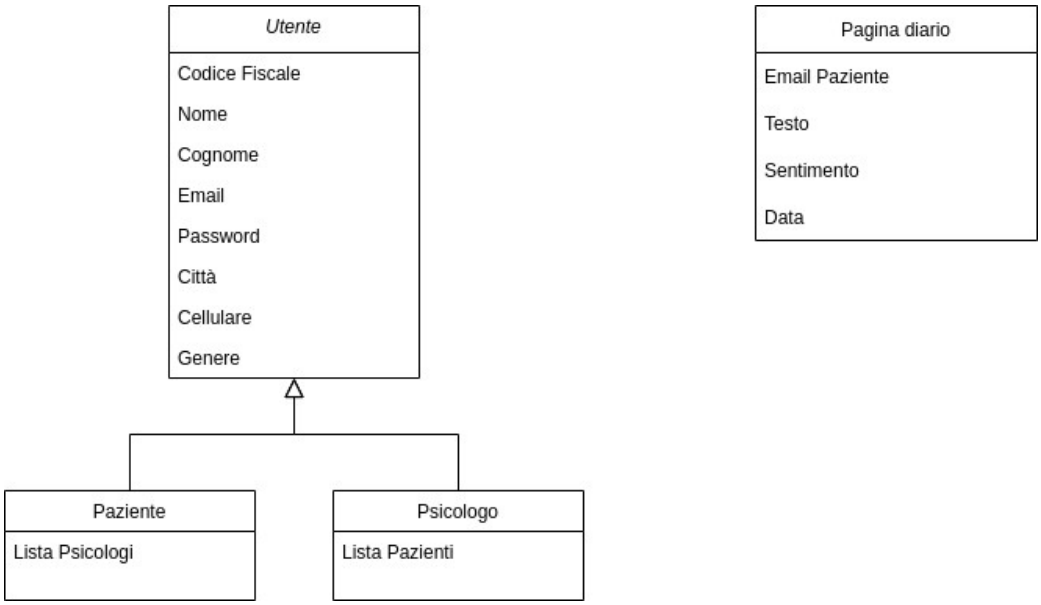
Nota: si segua il README.md all'interno della cartella di progetto per installare e mandare in esecuzione il progetto.

3 - Architettura dell'Applicazione

L'applicazione prevede l'utilizzo di due server REST, uno in in Go - per i casi d'uso CRUD riguardanti pazienti e psicologi – e uno in Python - per salvare i testi del diario su database e per effettuare la sentiment analysis prima del salvataggio.



3.1 – Diagramma delle Classi



Le principali strutture dati sono quattro, di cui due – Paziente e Psicologo – derivanti da Utente.

Pazienti e Psicologi infatti condividono gran parte delle informazioni, ma nello specifico implementano due liste differenti. I Pazienti infatti possiedono una lista degli psicologi di cui sono, appunto, pazienti mentre gli Psicologi hanno una lista di pazienti che hanno in cura.

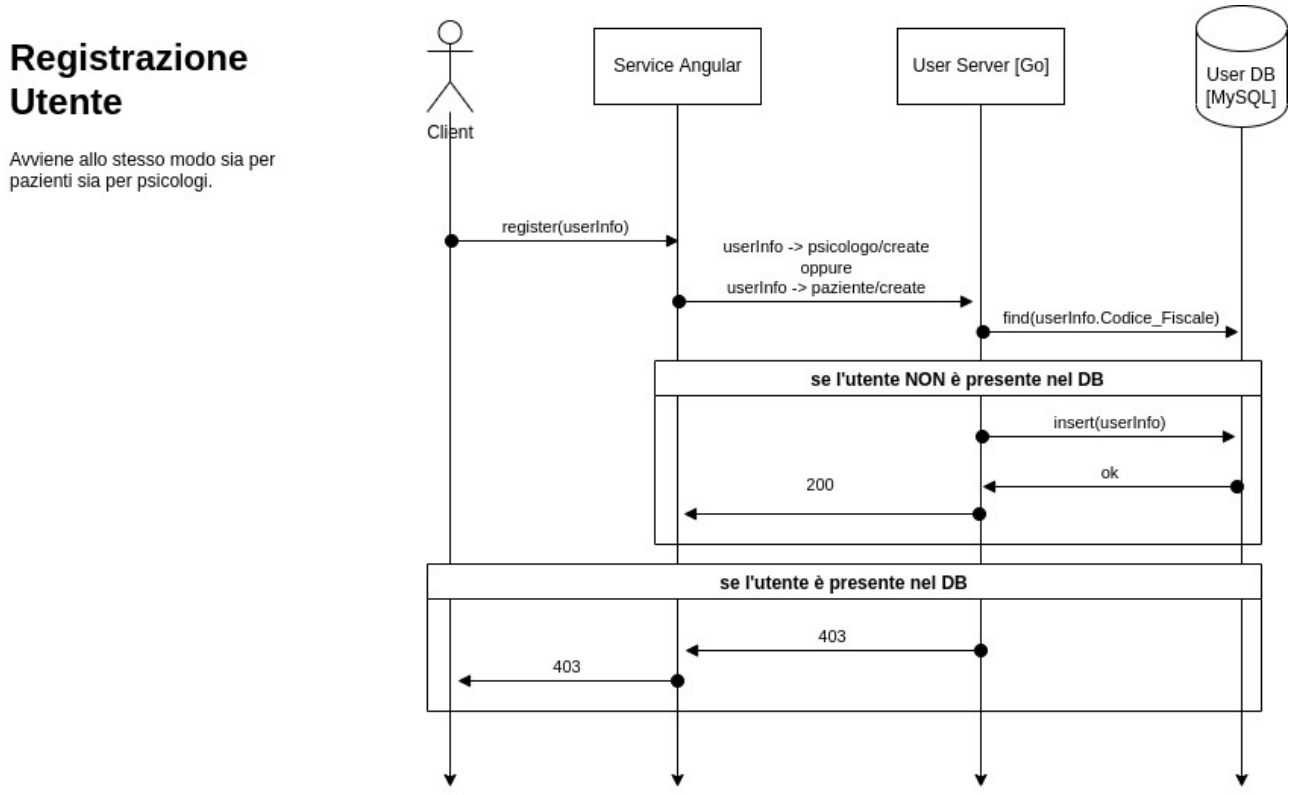
Il motivo per cui i pazienti hanno una lista di psicologi è spiegato dal fatto che un paziente possa essere preso in cura da più psicologi.

L'ultima struttura è la Pagina Diario, che rappresenta quanti e quali sono gli argomenti che vengono salvati – previa opportuna elaborazione – per indicare una *singola* pagina di diario.
A tale struttura, una volta salvata nel database Mongo, verrà aggiunto automaticamente da Mongo, un campo *__id*, che è un identificativo univoco del documento.

3.2 – Chiamate a User Server

- Registrazione Utente

La Registrazione dell'Utente ha un formato pressoché identico sia per i Pazienti sia per gli Psicologi, le entità e le loro azioni sono rappresentate nel seguente grafico:



Quindi il Client, psicologo o paziente, inserisce le proprie *userInfo* (ossia le informazioni della classe *Utente* vista sopra nel diagramma delle classi) all'interno del form ed effettua una post ad un opportuna URL:

- **/psicologo/create** se la richiesta viene lanciata dall'End Point dello psicologo. Nel database verranno aggiunti due record, uno nella tabella Utenti, contenente tutte le informazioni del diagramma, e uno nella tabella

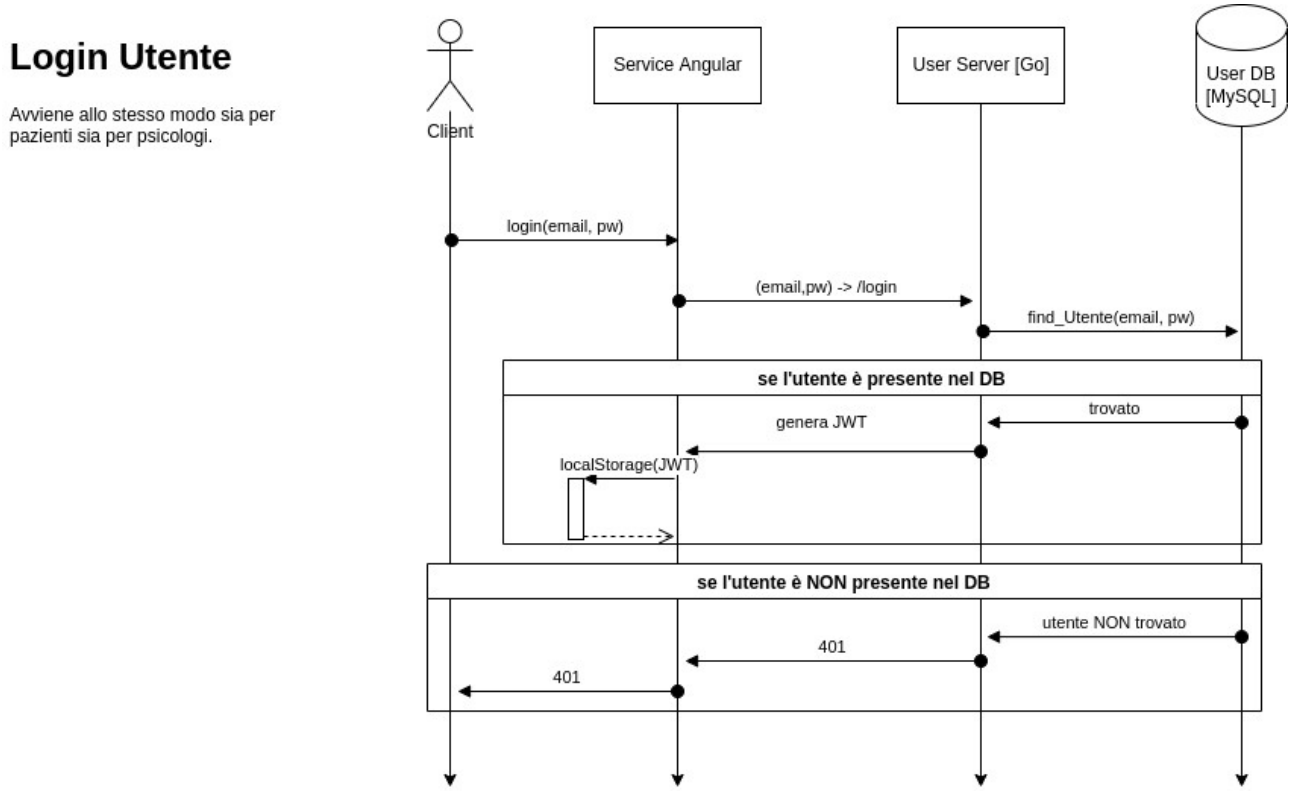
Psicologi, i cui campi sono il codice fiscale dello psicologo e una stringa (inizializzata come vuota) che conterrà l'elenco delle email dei pazienti.

- **/paziente/create** se la richiesta viene lanciata dall'End Point del paziente. Il comportamento della creazione del paziente è analogo a quello della creazione dello psicologo, solo che le tabelle in gioco saranno Utenti e Pazienti. In quest'ultima gli elementi della tabella sono costituiti dal codice fiscale del paziente e dalla lista degli psicologi di cui l'utente ne è paziente.

Il controllo di esistenza dell'utente viene effettuato sul codice fiscale.

- Login

Inserendo email e password, il client riceverà un Json Web Token (JWT) che verrà salvato nel local storage del browser ed utilizzato per evitare di dover eseguire il login ad ogni volta che si vuole usufruire del servizio:



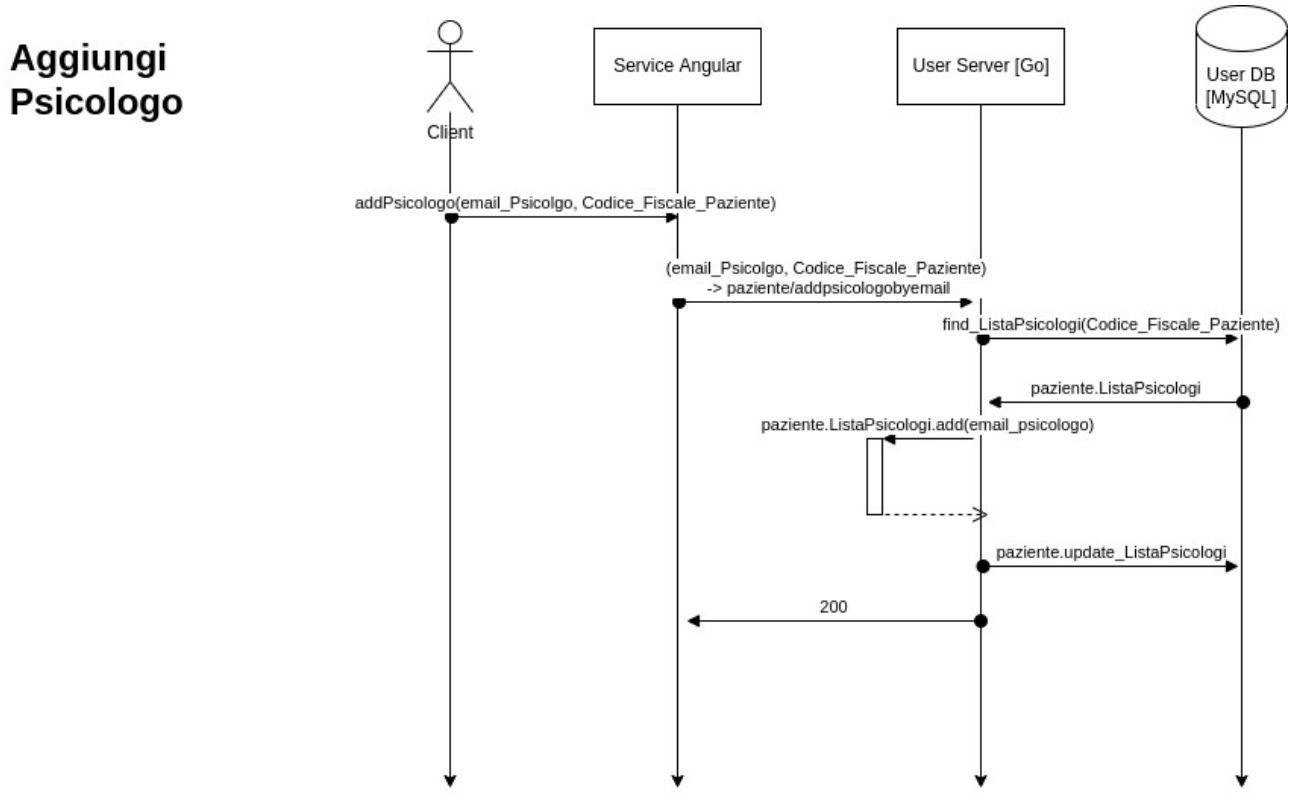
Oltre al token JWT, verranno salvate nel browser anche alcune informazioni sull'utente, quali nome, email e codice fiscale – che verranno utilizzate per altre chiamate.

- Aggiungi Psicologo (stesso procedimento per Aggiungi Paziente)

Nel grafico rappresentato sotto si descrive lo scenario di successo per l'aggiunta di uno psicologo nella *Lista Psicologi* di un paziente.

Il paziente inserisce in un apposito form l'email dello psicologo che sta cercando di aggiungere. Verrà lanciata una chiamata che contiene l'email dello psicologo che si

vuole aggiungere e, prelevandolo dalle informazioni salvate nel local storage, il codice fiscale del paziente che sta effettuando la richiesta:



Se lo psicologo che si sta provando ad aggiungere è già presente in lista, nel frontend verrà renderizzato un pannello che, al posto di permettere al paziente di effettuare una richiesta di aggiunta, permetterà al paziente di rimuovere lo psicologo dal proprio elenco.

Nota importante: per semplicità di sviluppo, si è scelto di implementare questa chiamata in maniera “bidirezionale” tra pazienti e psicologi: se un paziente aggiunge uno psicologo alla sua *Lista Psicologi*, il paziente verrà automaticamente aggiunto alla *Lista Pazienti* dello psicologo, e viceversa. Discorso analogo per la rimozione di un paziente/psicologo dalla rispettiva lista.

Le chiamate per la rimozione di uno psicologo dalla propria *Lista Psicologi*, così come anche le chiamate per l’aggiunta e la rimozione di un Paziente dalla *Lista Pazienti* di uno psicologo seguono esattamente lo stesso pattern di questo caso d’uso, quindi non sono state inserite.

3.3 – Chiamate a Text Analysis

Text Analysis è il secondo server, dedito all’interfacciamento col database MongoDB in cui vengono salvate le *pagine del diario*, ossia gli oggetti definiti con la classe *Pagina Diario* nel diagramma delle classi.

Pazienti e Psicologi hanno ruoli differenti: i primi scrivono il proprio diario mentre i secondi leggono i diari dei pazienti.

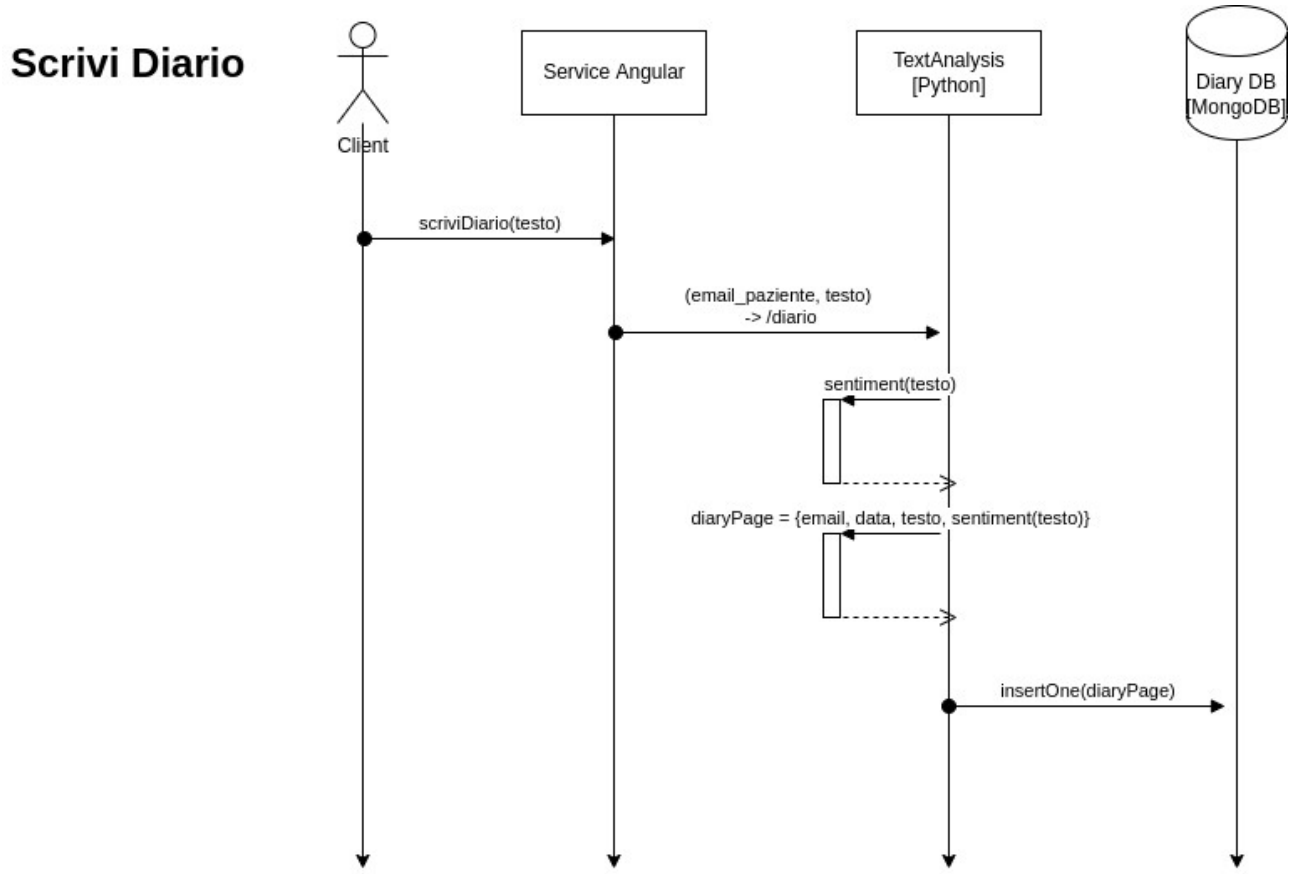
- [Paziente] Scrivi Diario

Dopo aver selezionato uno psicologo, da un opportuno elenco renderizzato nel frontend, il paziente può scrivere il proprio diario, da un input testuale.

Questo testo viene inviato al server Python insieme all'email del paziente, recuperata dalle informazioni precedentemente salvate nel local storage del browser. Il testo quindi viene prima tradotto in inglese con la libreria Googletrans e poi passato a TextBlob.

Al momento dell'arrivo della richiesta, viene effettuato un timestamp.

Tutte le informazioni in gioco, ossia l'email del paziente, il timestamp, il sentiment e il testo originale, vengono raggruppate e salvate nel database MongoDB.

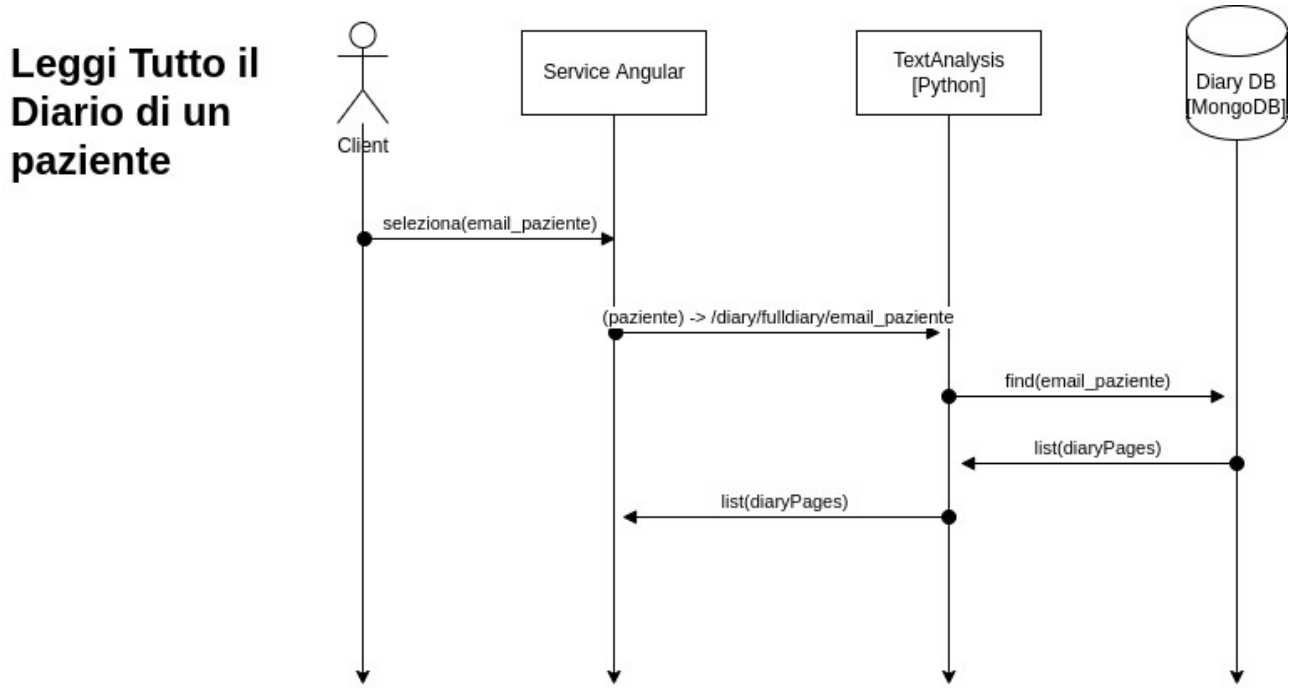


- [Psicologo] Leggi Tutto il Diario di un Paziente

Lo psicologo avrà renderizzata a schermo la propria *Lista Pazienti*. Selezionando uno di questi, verrà lanciata una chiamata a */diary/fulldiary/{email_paziente}* – dove *email_pazient* sarà la mail del paziente selezionato.

Il Server risponderà con una lista di pagine di diario, trovate cercando tutti i documenti nel database che hanno quella determinata email.

Verrà renderizzata una tabella in cui ogni riga è una pagina di diario, contenente le informazioni del diario.



- [Psicologo] Sentimento Medio del Paziente

Nella lista di pazienti dello psicologo, oltre ad essere renderizzate le funzionalità di visione del diario e di rimozione del paziente, è renderizzato anche il sentimento medio del paziente. Per ogni paziente nella lista, ne viene presa l'email e si lancia una chiamata a */paziente/metrics/meansentiment/{email_paziente}* e per ognuno di loro si ottiene un valore: