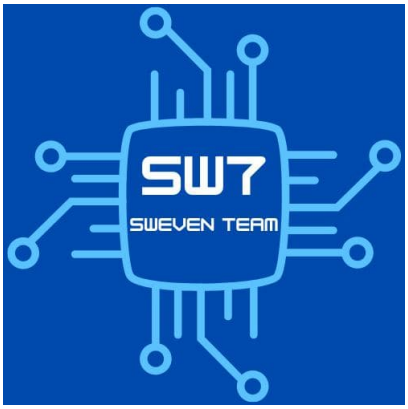
Specifica Architetturale



Sweven Team
swe7.team@gmail.com

Informazioni sul documento

Versione	0.0.0
Uso	Esterno
Destinatari	Gruppo Sweven Team Prof. Tullio Vardanega Prof. Riccardo Cardin Azienda Imola Informatica
Stato	in lavorazione
Redattori	
Verificatori	
Approvatori	

Sintesi

Specifica architetturale e delle tecnologie per la realizzazione del *Chatbot*$_G$.

Diario delle modifiche

Versione	Data	Descrizione	Ruolo	Autore	Verificatore
	2022-08-29	Completamento $2 e $3	Progettista	Irene Benetazzo	
	2022-08-27	Modifiche $2	Progettista	Irene Benetazzo	
	2022-08-22	Scrittura $2.2 e $2.3	Progettista	Irene Benetazzo	
	2022-08-09	Scrittura $3	Progettista	Irene Benetazzo	
	2022-08-08	Scrittura $1	Amministratore	Irene Benetazzo	
	2022-07-21	Creazione documento	Amministratore	Irene Benetazzo	

Indice

1 Introduzione

1.1 Scopo del Documento

La Specifica Architetturale ha lo scopo di descrivere le scelte architetturali e tecnologiche attuate per la realizzazione del *Chatbot*$_G$.

1.2 Scopo del Capitolato

Lo scopo di tale progetto è quello di sviluppare un Chatbot che interfacciandosi con software aziendali spesso complessi e dispersivi, semplifichi i compiti che i dipendenti devono svolgere. In particolare vengono individuate le seguenti operazioni:

- Tracciamento della presenza in sede (**EMT**$_G$)
- Rendiconto attività svolte quotidianamente (**EMT**$_G$)
- Apertura del cancello aziendale (**MQTT**$_G$)
- Creazione di una riunione in un servizio esterno
- Servizio di ricerca documentale (**CMIS**$_G$)
- Creazione e tracciamento di bug (**Redmine**$_G$)

1.3 Glossario

Per assicurare la massima fruibilità e leggibilità del documento, il team SWEven ha deciso di creare un documento denominato *Glossario* il cui scopo sarà quello di contenere le definizioni dei termini ambigui o specifici del progetto. Sarà possibile riconoscere i termini presenti al suo interno in quanto terminanti con la lettera *G* posta come pedice della parola stessa.

1.4 Riferimenti

1.4.1 Normativi

- Norme di Progetto *v1.0.0*

1.4.2 Informativi

- Capitolato di appalto C1 - BOT4ME
- Slide del corso - Diagrammi dei casi d'uso
- Slide del corso - Diagrammi di sequenza
- Slide del corso - I pattern architetturali

2 Architettura

L'architettura del prodotto è suddivisa tra Client e Server, inoltre si utilizzano le *API Rest*$_G$ messe a disposizione dall'azienda Imola Informatica.

2.1 Diagramma delle classi

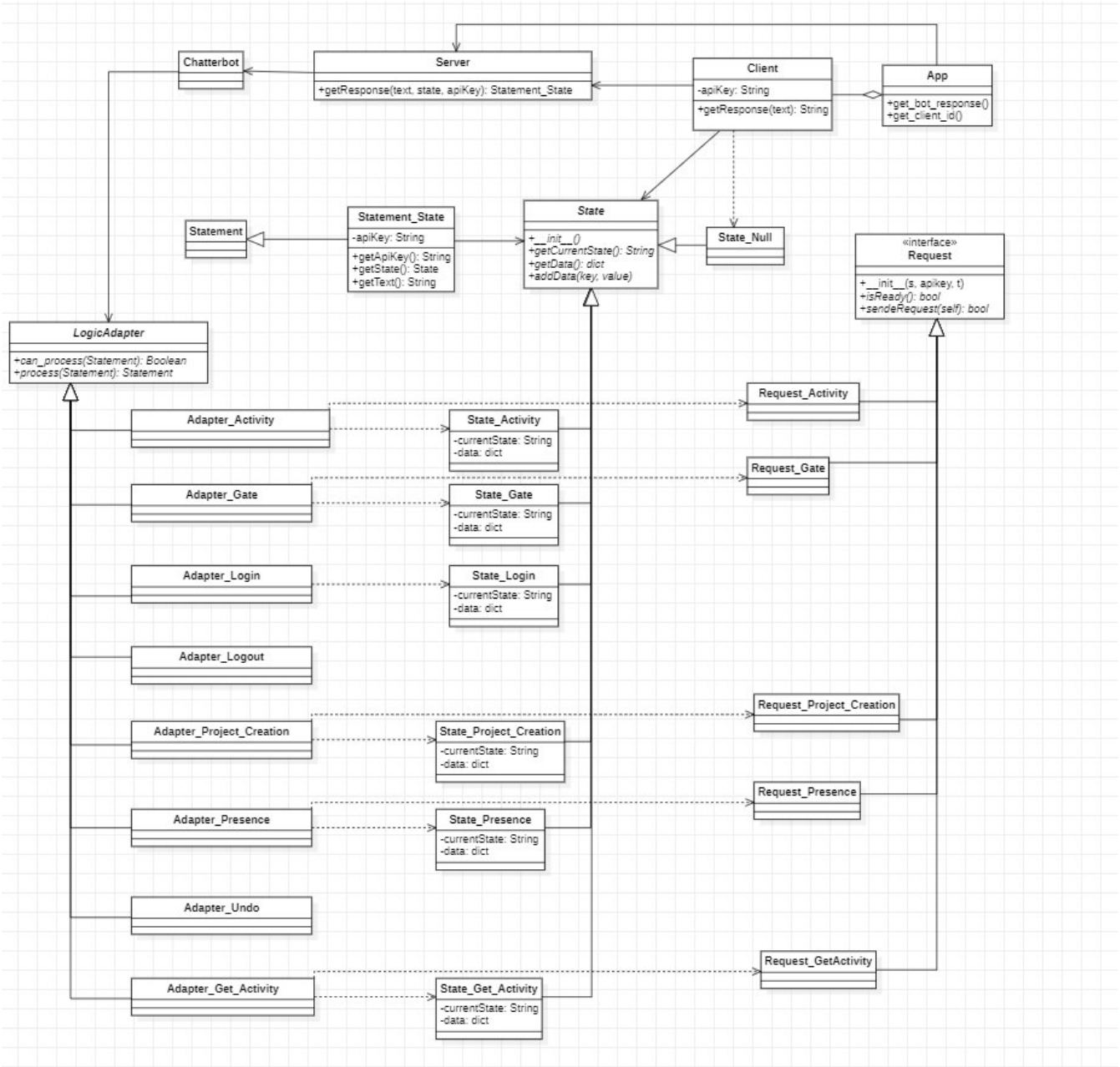


Figure 1: Diagramma UML delle classi

2.2 Client

Il client è statless cioè non contiene stati ma viene tutto totalmente gestito dal server. Ad ogni avvio dell'applicazione viene assegnato un nuovo clientID che viene salvato nel localstorage e ad ogni messaggio inviato dall'utente viene aggiunto l'attributo del *clientID* oltre all'*api-key* che è considerata come precondizione. Anche solo l'aggiornamento della pagina nel browser comporta l'assegnazione di un nuovo clientID ma se non si è effettuato il logout l'api-key risulterà già inserita e basterà solo inviarla nuovamente.

2.3 Server

2.3.1 App

Classe in cui vengono gestiti gli utenti e si interfaccia direttamente con l'utente, inoltre assegna il nuovo client id ai nuovi utenti.

2.3.2 Client

Classe che rappresenta la minisessione di ogni utente nel server e ha tutti i dati salvati.

2.3.3 Server

Classe che contiene la logica utilizzata dal chatbot per poter rispondere correttamente all'input del client tramite l'utilizzo di *Statement_State*.

2.3.4 Chatterbot

Classe della libreria esterna scritta in *Python*$_G$. La classe *Chatterbot* e le seguenti *Statement*, *Adapter* fanno parte della libreria.

2.3.5 Statement

Classe fornita dalla libreria *Chatterbot*$_G$ che rappresenta una singola entità, parola o frase che qualcuno può dire.

2.3.6 LogicAdapter

Classe astratta fornita dalla libreria *Chatterbot*$_G$ che permette al programmatore esterno di scrivere nuovi adapter. Dispone dei due metodi base di cui verrà fatto l'*overriding*$_G$:

- can_process: metodo booleano che controlla tutte le varie condizioni e se tutto okay fa procedere il metodo *process*.
- process: controlla ed elabora tutti i dati forniti così da produrre una risposta.

2.3.7 State

Interfaccia che definisce il contratto di tutti i vari stati e come dato privato si salva l'attuale stato corrente e pubblicamente dispone anche di un metodo per aggiungere informazioni necessarie per completare la richiesta in corso.

State_Null Sottoclasse concreta di *Stato* che simula uno stato nullo, utilizzato quando l'utente non ha effettuato nessuna richiesta.

2.3.8 Statement_State

Sottoclasse di Statement, cioè adatta l'adapter alla libreria chatterbot, in più ha lo stato attuale dell'utente, e l'api-key che dimostra l'autenticazione dell'utente che funge come input di ogni adapter.

2.3.9 Request

Interfaccia che riceve i dati pronti verificandone la completezza e in base all'*adapter* invia la richiesta $HTTP_G$ alle $API\ Rest_G$ di Imola per interagire con i loro servizi e soddisfare la richiesta dell'utente e infine ritorna ad adapter una risposta.

2.3.10 Login

Classi *Adapter_Login*, *State_Login* permettono di effettuare il login

2.3.11 Logout

Classe *Adapter_Logout* permette di effettuare il logout.

2.3.12 Activity

Classi *Adapter_Activity*, *State_Activity* e *Request_Activity* per la funzionalità di consuntivare le ore dedicate ad un progetto compreso le eventuali ore di viaggio.

2.3.13 Gate

Classi *Adapter_Gate*, *State_Gate* e *Request_Gate* per la funzionalità di apertura cancello

2.3.14 Project_Creation

Classi *Adapter_Project_Creation*, *State_Project_Creation* e *Request_Project_Creation*

2.3.15 Presence

Classi *Adapter_Presence*, *State_Presence* e *Request_Presence* per la funzionalità di registrazione della presenza

2.3.16 Undo

Classe *Adapter_Undo* permette di annullare l'operazione in corso e di ricominciare la stessa o un'altra operazione dall'inizio.

2.4 API Rest Imola Informatica

L'azienda ha fornito delle *API Rest*$_G$ che permettono al *chatbot*$_G$ di interagire con i loro sistemi aziendali. Sono facilmente consultabili a questo link.

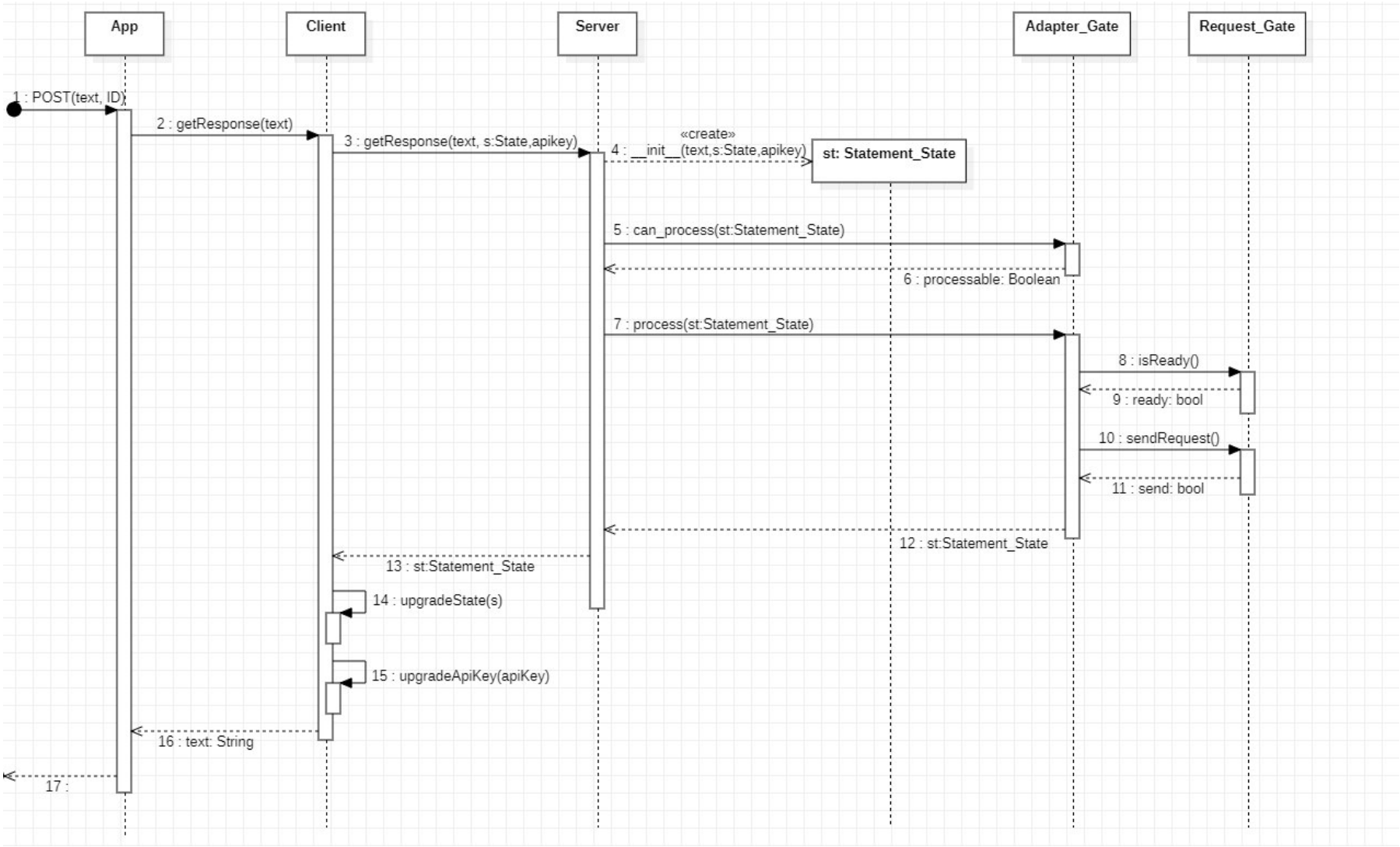


Figure 2: Diagramma di sequenza Server

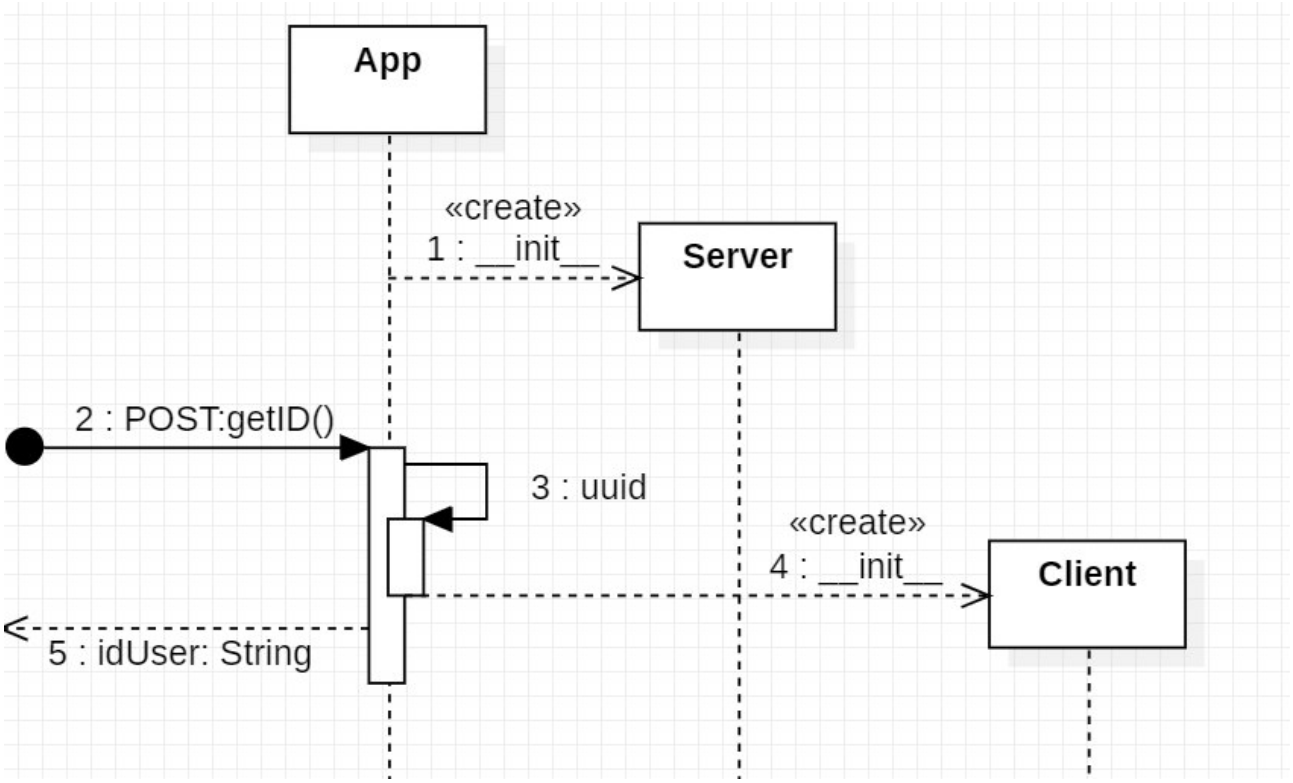


Figure 3: Diagramma di sequenza Client

2.5 Design Pattern

Front Controller pattern è un pattern architetturale per la progettazione di applicazioni web fornendo un punto di ingresso centralizzato per la gestione delle richieste. Viene usato il server per gestire le operazioni comuni e poi in base alle specifiche richieste si chiama il rispettivo adapter e state.

3 Tecnologie

3.1 API Rest

Un'API REST è un'interfaccia di programmazione delle applicazioni conforme ai vincoli dello stile architetturale REST, che consente l'interazione con servizi web RESTful.
Il termine REST è l'acronimo di REpresentational State Transfer. REST è un insieme di vincoli architetturali, non un protocollo né uno standard. Quando una richiesta client viene inviata tramite un'API RESTful, questa trasferisce al richiedente o all'endpoint uno stato rappresentativo della risorsa. L'informazione viene consegnata in HTTP in un formato JSON, HTML, Python o txt.

3.2 Server

3.2.1 Python

Linguaggio di programmazione ad alto livello, adatto alla programmazione orientata agli oggetti. E' stato utilizzato per sviluppare il back-end insieme alla libreria esterna Chatterbot.

3.2.1.1 Chatterbot Libreria esterna in *Python*$_G$ che utilizza algoritmi di intelligenza artificiale per trovare la migliore risposta per emulare il comportamento di un *chatbot*$_G$ nel server. Grazie alla sua flessibilità si sono implementati degli adapter che modellano e gestiscono le varie richieste dell'utente.
Durante l'esecuzione Chatterbot crea un adapter che gli consente di connettersi ad un database *SQLite*$_G$.

3.2.2 Flask

Framework Python per lo sviluppo di applicazioni web. Flask contiene tutte le classi e le funzioni necessarie per la costruzione di una web app, e ha agevolato l'organizzazione e la gestione del *chatbot*$_G$.

3.2.3 Cors

Cross-Origin Resource Sharing regola la cooperazione tra sito web e il caricamento dati da un server esterno garantendone la sicurezza attraverso un'intestazione *HTTP*$_G$.

3.2.4 UUID

Universally Unique IDentifier, cioè identificativo univoco universale, è usato nelle infrastrutture software per avere l'unicità pratica senza garanzia in un sistema distribuito semplificando il mantenimento dell'identità degli oggetti in ambienti disconnessi.

3.3 Client

3.3.1 React

React è una libreria JavaScript per costruire l'interfaccia utente caratterizzata dal fatto che è dichiarativa, efficiente e flessibile. E' stato utilizzato per creare l'applicazione lato client.

3.3.2 HTML

Linguaggio di markup, in standard W3C, per documenti visualizzabili attraverso un web browser.

3.3.3 CSS

Linguaggio di formattazione per i documenti HTML.

3.3.4 Regex

Libreria esterna che controlla le espressioni regolari per la validazione dell'*api-key*$_G$.

3.3.5 API AssemblyAI

Servizio che permette di tradurre automaticamente l'audio in testo richiamando l'api Speech-to-text. Link