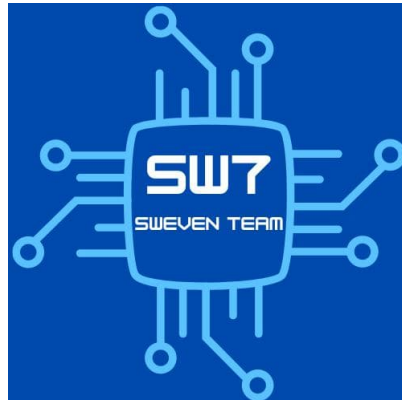


# SPECIFICA ARCHITETTURALE



**SWEVEN TEAM**

swe7.team@gmail.com

## INFORMAZIONI SUL DOCUMENTO

<b>Versione</b>	0.0.0
<b>Uso</b>	Esterno
<b>Destinatari</b>	Gruppo Sweven Team Prof. Tullio Vardanega Prof. Riccardo Cardin Azienda Imola Informatica
<b>Stato</b>	in lavorazione
<b>Redattori</b>	Irene Benetazzo Samuele Rizzato
<b>Verificatori</b>	
<b>Approvatori</b>	

## Sintesi

Specifica architetturale e delle tecnologie per la realizzazione del *Chatbot<sub>G</sub>*.

## Diario delle modifiche

Versione	Data	Descrizione	Ruolo	Autore	Verificatore
	2022-08-30	Aggiornato \$2.4 e \$2.1	Progettista	Samuele Rizzato	
	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

<b>1</b>	<b>Introduzione</b>	<b>5</b>
1.1	Scopo del Documento . . . . .	5
1.2	Scopo del Capitolato . . . . .	5
1.3	Glossario . . . . .	5
1.4	Riferimenti . . . . .	5
1.4.1	Normativi . . . . .	5
1.4.2	Informativi . . . . .	5
<b>2</b>	<b>Architettura</b>	<b>6</b>
2.1	Diagramma delle classi . . . . .	6
2.2	Client . . . . .	6
2.3	Server . . . . .	7
2.3.1	App . . . . .	7
2.3.2	Client . . . . .	7
2.3.3	Server . . . . .	7
2.3.4	Chatterbot . . . . .	7
2.3.5	Statement . . . . .	7
2.3.6	LogicAdapter . . . . .	7
2.3.7	State . . . . .	7
2.3.8	Statement_State . . . . .	8
2.3.9	Request . . . . .	8
2.3.10	Login . . . . .	8
2.3.11	Logout . . . . .	8
2.3.12	Activity . . . . .	8
2.3.13	Gate . . . . .	8
2.3.14	Project_Creation . . . . .	8
2.3.15	Presence . . . . .	8
2.3.16	Undo . . . . .	8
2.4	API Rest Imola Informatica . . . . .	9
2.4.1	Consuntivazione di un attività . . . . .	9
2.4.1.1	API Url . . . . .	9
2.4.1.2	Metodo di richiesta <i>HTTP<sub>G</sub></i> . . . . .	9
2.4.1.3	Headers <i>HTTP<sub>G</sub></i> . . . . .	9
2.4.1.4	Parametri URL . . . . .	9
2.4.1.5	Parametri Body . . . . .	9
2.4.1.6	Risposte . . . . .	10
2.4.2	Apertura del cancello . . . . .	10
2.4.2.1	API Url . . . . .	10
2.4.2.2	Metodo di richiesta <i>HTTP<sub>G</sub></i> . . . . .	10
2.4.2.3	Headers <i>HTTP<sub>G</sub></i> . . . . .	10
2.4.2.4	Parametri URL . . . . .	10
2.4.2.5	Parametri Body . . . . .	10
2.4.2.6	Risposte . . . . .	11
2.4.3	Registrazione della presenza . . . . .	11

2.4.3.1	API Url . . . . .	11
2.4.3.2	Metodo di richiesta <i>HTTP<sub>G</sub></i> . . . . .	11
2.4.3.3	Headers <i>HTTP<sub>G</sub></i> . . . . .	11
2.4.3.4	Parametri URL . . . . .	11
2.4.3.5	Risposte . . . . .	11
2.4.4	Creazione di un nuovo progetto . . . . .	12
2.4.4.1	API Url . . . . .	12
2.4.4.2	Metodo di richiesta <i>HTTP<sub>G</sub></i> . . . . .	12
2.4.4.3	Headers <i>HTTP<sub>G</sub></i> . . . . .	12
2.4.4.4	Parametri Body . . . . .	12
2.4.4.5	Risposte . . . . .	12
2.4.5	Recupero delle sedi . . . . .	13
2.4.5.1	API Url . . . . .	13
2.4.5.2	Metodo di richiesta <i>HTTP<sub>G</sub></i> . . . . .	13
2.4.5.3	Headers <i>HTTP<sub>G</sub></i> . . . . .	13
2.4.5.4	Risposte . . . . .	13
2.4.6	Recupero della rendicontazione per il progetto corrente . . . . .	13
2.4.6.1	API Url . . . . .	13
2.4.6.2	Metodo di richiesta <i>HTTP<sub>G</sub></i> . . . . .	13
2.4.6.3	Headers <i>HTTP<sub>G</sub></i> . . . . .	13
2.4.6.4	Parametri URL . . . . .	13
2.4.6.5	Risposte . . . . .	14
2.5	Design Pattern . . . . .	16
<b>3</b>	<b>Tecnologie</b> . . . . .	<b>17</b>
3.1	API Rest . . . . .	17
3.2	Server . . . . .	17
3.2.1	Python . . . . .	17
3.2.1.1	Chatterbot . . . . .	17
3.2.2	Flask . . . . .	17
3.2.3	Cors . . . . .	17
3.2.4	UUID . . . . .	17
3.3	Client . . . . .	18
3.3.1	React . . . . .	18
3.3.2	HTML . . . . .	18
3.3.3	CSS . . . . .	18
3.3.4	Regex . . . . .	18
3.3.5	API AssemblyAI . . . . .	18

# 1 Introduzione

## 1.1 Scopo del Documento

La Specifica Architettuale ha lo scopo di descrivere le scelte architettureali e tecnologiche attuate per la realizzazione del *Chatbot<sub>G</sub>*.

## 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<sub>G</sub>**)
- Rendiconto attività svolte quotidianamente (**EMT<sub>G</sub>**)
- Apertura del cancello aziendale (**MQTT<sub>G</sub>**)
- Creazione di una riunione in un servizio esterno
- Servizio di ricerca documentale (**CMIS<sub>G</sub>**)
- Creazione e tracciamento di bug (**Redmine<sub>G</sub>**)

## 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](#)



e ad ogni messaggio inviato dall'utente viene aggiunto l'attributo del *clientID* oltre all'*api-key* che è considerata come preconditione. 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<sub>G</sub>*. La classe *Chatterbot* e le seguenti *Statement*, *Adapter* fanno parte della libreria.

### 2.3.5 Statement

Classe fornita dalla libreria *Chatterbot<sub>G</sub>* che rappresenta una singola entità, parola o frase che qualcuno può dire.

### 2.3.6 LogicAdapter

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

- *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<sub>G</sub>* alle *API Rest<sub>G</sub>* 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<sub>G</sub>* che permettono al *chatbot<sub>G</sub>* di interagire con i loro sistemi aziendali. Sono facilmente consultabili a questo [link](#).

### 2.4.1 Consuntivazione di un attività

#### 2.4.1.1 API Url

/projects/{code}/activities/me

#### 2.4.1.2 Metodo di richiesta *HTTP<sub>G</sub>*

POST

#### 2.4.1.3 Headers *HTTP<sub>G</sub>*

- **accept:** application/json;
- **api\_key:** api\_key, per autorizzare le richieste;
- **Content-Type:** application/json.

#### 2.4.1.4 Parametri URL

Nome	Tipo	Descrizione
code	string	rappresenta il codice del progetto di cui fa parte l'attività da rendicontare.

#### 2.4.1.5 Parametri Body

Nel body della richiesta viene passato un array contenente un json con questi parametri

Nome	Tipo	Descrizione
date	string	data in cui si è svolta l'attività.
billableHours	int	ore fatturabili dell'attività.
travelHours	int	ore di viaggio spese per l'attività.
billableTravelHours	int	ore di viaggio fatturabili per l'attività.
location	string	nome della sede in cui si è svolta l'attività.
billable	bool	indica se l'attività è fatturabile.
note	string	descrizione dell'attività.

### 2.4.1.6 Risposte

Status code <i>HTTP<sub>G</sub></i>	Descrizione
204	indica che la richiesta di consuntivazione è andata a buon fine
404	indica che il codice specificato del progetto non è corretto
401	indica che l'api key non è stata specificata o quella utilizzata non è valida.

## 2.4.2 Apertura del cancello

### 2.4.2.1 API Url

/locations/{location\_name}/devices/{device}/status

### 2.4.2.2 Metodo di richiesta *HTTP<sub>G</sub>*

PUT

### 2.4.2.3 Headers *HTTP<sub>G</sub>*

- **accept:** application/json;
- **api\_key:** api\_key, per autorizzare le richieste;
- **Content-Type:** application/json.

### 2.4.2.4 Parametri URL

Nome	Tipo	Descrizione
location_name	string	sede di cui aprire il cancello.
device	string	nome del dispositivo da utilizzare, in questo caso il cancello.

### 2.4.2.5 Parametri Body

Nel body della richiesta viene passato un json contenente questi parametri

Nome	Tipo	Descrizione
status	string	rappresenta il nuovo stato del dispositivo.

### 2.4.2.6 Risposte

Status code <i>HTTP<sub>G</sub></i>	Descrizione
204	indica che la richiesta di apertura del cancello è andata a buon fine.
404	indica che la sede di cui aprire il cancello non è valida.
401	indica che l'api key non è stata specificata o quella utilizzata non è valida.

### 2.4.3 Registrazione della presenza

#### 2.4.3.1 API Url

/locations/{location\_name}/presence

#### 2.4.3.2 Metodo di richiesta *HTTP<sub>G</sub>*

POST

#### 2.4.3.3 Headers *HTTP<sub>G</sub>*

- **accept:** application/json;
- **api\_key:** api\_key, per autorizzare le richieste;
- **Content-Type:** application/json.

#### 2.4.3.4 Parametri URL

Nome	Tipo	Descrizione
location_name	string	sede in cui registrare la presenza.

#### 2.4.3.5 Risposte

Status code <i>HTTP<sub>G</sub></i>	Descrizione
200	indica che la registrazione della presenza è stata effettuata con successo.
404	indica che la sede specificata non è valida.
401	indica che l'api key non è stata specificata o quella utilizzata non è valida.

## 2.4.4 Creazione di un nuovo progetto

### 2.4.4.1 API Url

/projects

### 2.4.4.2 Metodo di richiesta *HTTP<sub>G</sub>*

POST

### 2.4.4.3 Headers *HTTP<sub>G</sub>*

- **accept:** application/json;
- **api\_key:** api\_key, per autorizzare le richieste;

### 2.4.4.4 Parametri Body

Nel body della richiesta viene passato un json contenente questi parametri

Nome	Tipo	Descrizione
code	string	codice del progetto da creare.
detail	string	descrizione del progetto da creare.
customer	string	cliente per cui viene creato il progetto.
manager	string	manager del progetto da creare.
status	string	stato del progetto.
area	string	sede in cui svolgere il progetto.
startDate	string	data di inizio del progetto.
endDate	string	data di fine del progetto.

### 2.4.4.5 Risposte

Status code <i>HTTP<sub>G</sub></i>	Descrizione
204	indica che è l'operazione di creazione del progetto è andata a buon fine.
400	indica che almeno uno dei parametri non è stato specificato.
401	indica che l'api key non è stata specificata o quella utilizzata non è valida.

## 2.4.5 Recupero delle sedi

### 2.4.5.1 API Url

/locations

### 2.4.5.2 Metodo di richiesta *HTTP<sub>G</sub>*

GET

### 2.4.5.3 Headers *HTTP<sub>G</sub>*

- **accept:** application/json;
- **api\_key:** api\_key, per autorizzare le richieste;

### 2.4.5.4 Risposte

Status code <i>HTTP<sub>G</sub></i>	Descrizione
200	ritorna una lista contenente le informazioni delle sedi.
401	indica che l'api key non è stata specificata o quella utilizzata non è valida.

## 2.4.6 Recupero della rendicontazione per il progetto corrente

### 2.4.6.1 API Url

/projects/{code}

### 2.4.6.2 Metodo di richiesta *HTTP<sub>G</sub>*

GET

### 2.4.6.3 Headers *HTTP<sub>G</sub>*

- **accept:** application/json;
- **api\_key:** api\_key, per autorizzare le richieste;

### 2.4.6.4 Parametri URL

Nome	Tipo	Descrizione
code	string	codice del progetto corrente.

#### 2.4.6.5 Risposte

Status code <i>HTTP<sub>G</sub></i>	Descrizione
200	ritorna un json contenente le informazioni per il progetto corrente.
401	indica che l'api key non è stata specificata o quella utilizzata non è valida.

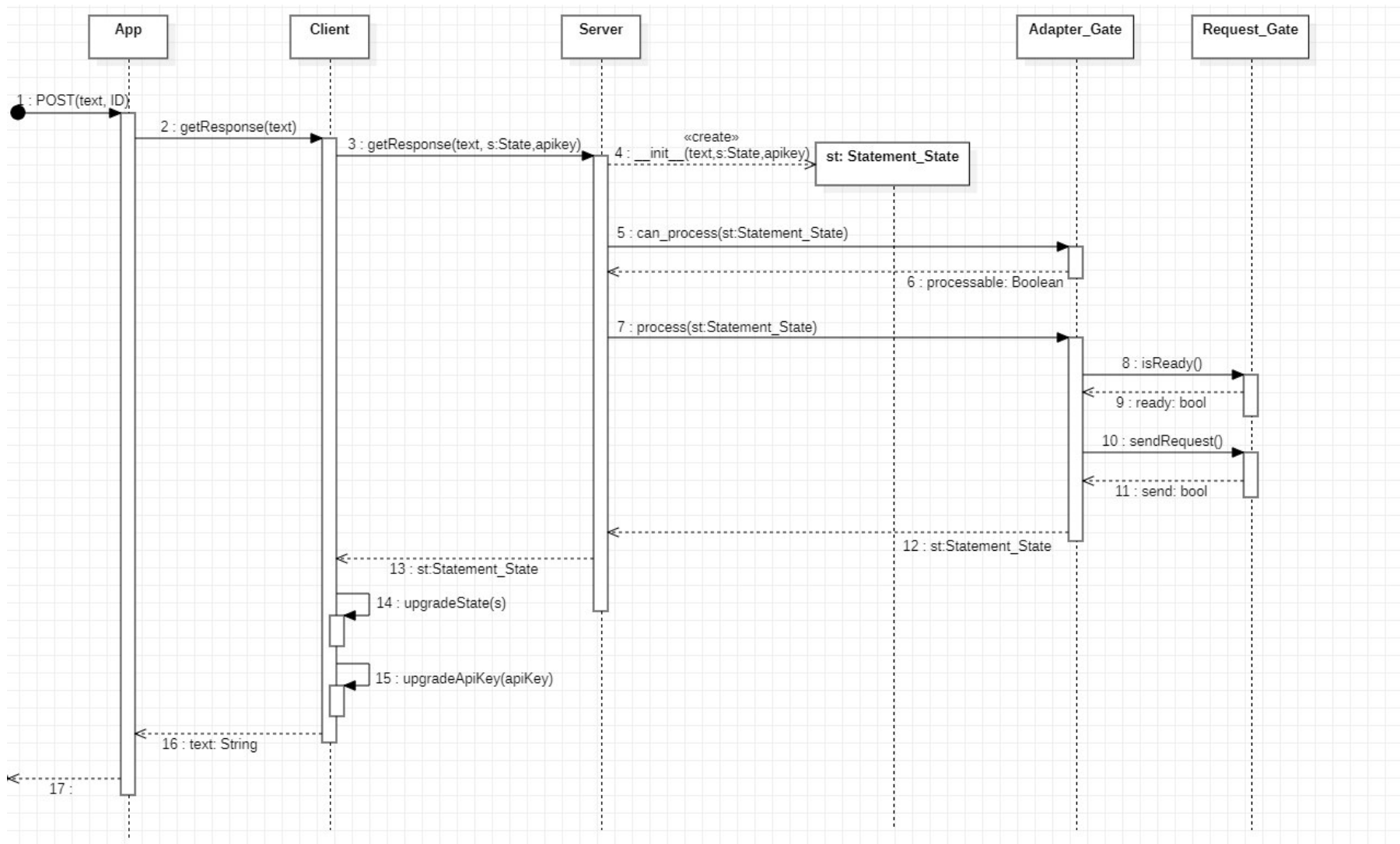


Figure 2: Diagramma di sequenza Server

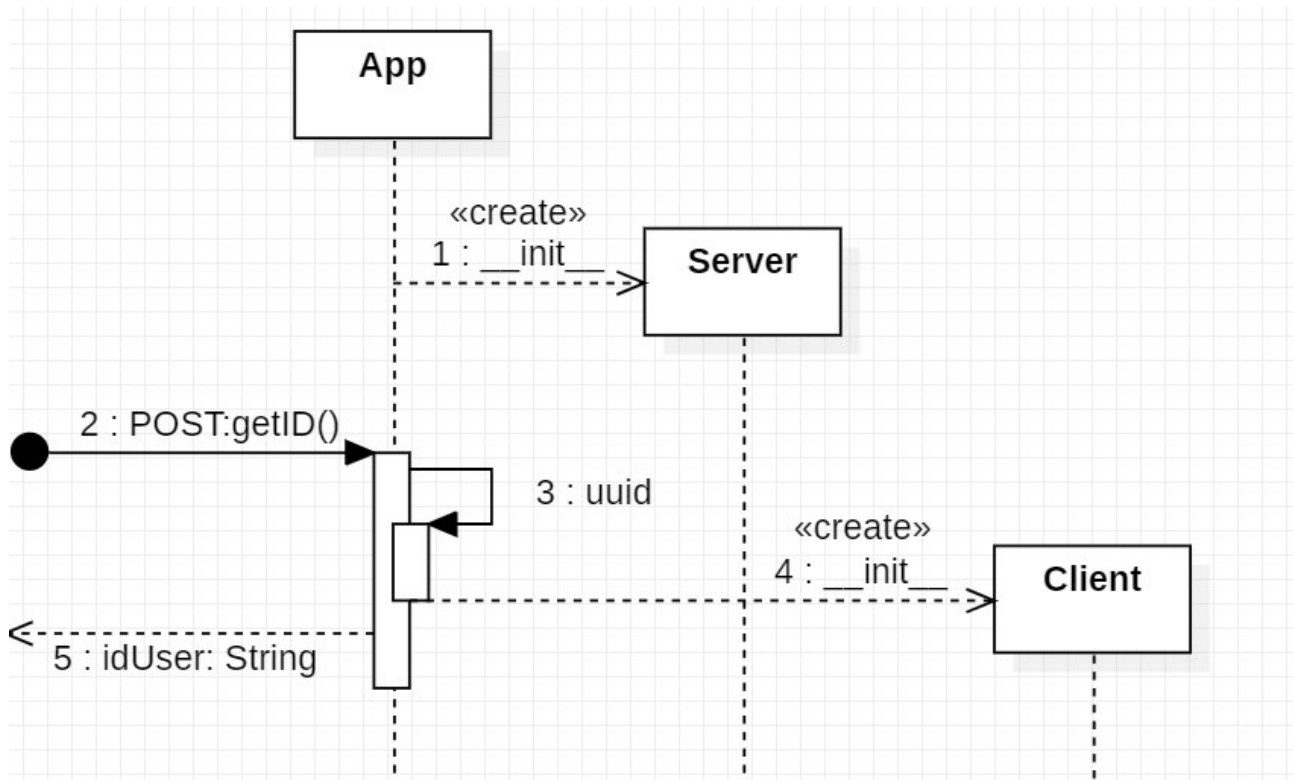


Figure 3: Diagramma di sequenza Client

## 2.5 Design Pattern

Front Controller pattern è un pattern architetture 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 architettuale 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 architettureali, 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<sub>G</sub>* che utilizza algoritmi di intelligenza artificiale per trovare la migliore risposta per emulare il comportamento di un *chatbot<sub>G</sub>* 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<sub>G</sub>*.

#### 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<sub>G</sub>*.

#### 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<sub>G</sub>*.

#### 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<sub>G</sub>*.

#### 3.3.5 API AssemblyAI

Servizio che permette di tradurre automaticamente l'audio in testo richiamando l'api Speech-to-text. [Link](#)