

UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE



Sviluppo di un ChatBot per la prenotazione di stanze d'hotel basato su Rasa

Studenti:

Bedetta Alessandro
Ascani Christian

Docenti:

Ursino Domenico
Marchetti Michele

Sommario

1. Chatbot	2
2. RASA.....	3
2.1 Obiettivo del progetto.....	4
3. Strumenti utilizzati.....	5
3.1 XAMPP	5
3.2 API Wikipedia.....	6
3.3 Ngrok.....	6
4. Implementazione.....	7
4.1 Nome e cognome	8
4.2 Attrazioni	8
4.3 FAQ.....	10
4.4 Dettagli sulle stanze	10
4.5 Prenotazione	11
5. Test.....	16
6. Conclusioni e sviluppi futuri	24

1. Chatbot

Col termine chatbot, come è possibile intuire dal nome, si indica un software finalizzato ad automatizzare il processo di comunicazione con l'utente.

Questi programmi sono in grado di simulare, in maniera più o meno realistica, una conversazione con l'utente, basata sul linguaggio naturale.

Le aziende odierne fanno largo uso di questi sistemi, in quanto permettono di automatizzare i processi che prevedono un interfacciamento con l'utente, garantendo un servizio attivo 24 ore su 24, 7 giorni su 7.

Nella letteratura si tende a distinguere fra due tipologie principali di bot:

- **TASK ORIENTED:** Si tratta di assistenti mirati ad un task specifico; puntano a fornire aiuto all'utente, facilitandolo nell'esecuzione di determinate operazioni. Questo tipo di software attingono da set di dati predefiniti (es. Database, API, ecc.) per restituire all'utente risposte inerenti alle richieste di informazioni.
- **CHATBOT CONVERSAZIONALI:** Si tratta di assistenti virtuali in tutto e per tutto, e quindi non puntano ad avere una conoscenza limitata ad un ambito preciso, ma devono essere in grado di sostenere conversazioni in merito ad ogni topic possibile.

Per lo svolgimento di questo progetto ci limiteremo allo sviluppo di un bot task oriented capace di fornire all'utente informazioni utili alla prenotazione di una stanza d'hotel. Infatti, per soddisfare la domanda del mercato in continua crescita e attirare sempre più clienti, i grandi competitor nel settore alberghiero continuano a migliorare i loro servizi e cercano nuovi modi per rendere la loro offerta più conveniente. I chatbot svolgono un ruolo sempre più importante in tutte le operazioni alberghiere, dall'indirizzare le prenotazioni dirette all'aumento della conversione del sito web, dall'aiutare gli albergatori, come il concierge e gli agenti della reception, ad automatizzare le richieste ripetitive degli ospiti.

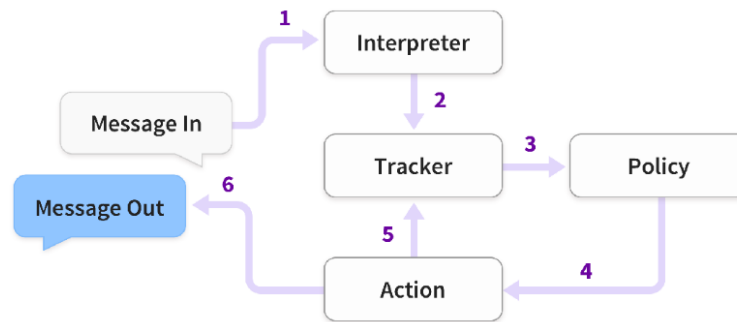
2. RASA

Rasa è un framework open source finalizzato allo sviluppo di chatbot che automatizzano il processo di conversazione basata su testo e voce. Gli assistenti vocali sviluppati in rasa possono essere in grado di comunicare attraverso svariate piattaforme come Facebook Messenger, Slack, Google Hangouts, Telegram e molti altri. Rasa fornisce anche dei moduli che permettono ai software di comunicare con dispositivi come Amazon Alexa e Google Home.

Il framework RASA è identificabile come l'unione di due sottocomponenti, ognuna delle quali aventi scopi precisi:

- **RASA NLU** (Natural Language Understanding): La componente di RASA che mira ad interpretare il linguaggio naturale dato come input (intent) ed estrarre da esso la volontà e i comandi dati dall'utente.
- **RASA CORE**: La componente che, una volta interpretati gli intent da parte dell'NLU, si occupa di andare a gestire le response, ovvero pianificare azioni di risposta a quelli che sono stati gli input da parte dell'utente.

La Rasa NLU necessita, per essere calata nel contesto operativo, di una fase di training supervisionato; ciò vuol dire che bisognerà allestire un dataset atto ad addestrare l'unità NLU per interpretare correttamente ogni possibile input dato al bot. Per generare una risposta esaustiva all'utente, si deve seguire una sequenza ben definita. Innanzitutto, i messaggi dati in input dall'utente vengono classificati in una serie di intent ed analizzati dall'interprete, il quale si occupa di estrarre le entità (un oggetto specifico, che si può riferire ad una persona, luogo, cosa, ecc.), qual ora fossero presenti. Successivamente, il messaggio di input è convertito in un dizionario che include il testo originale, l'intent e tutte le entità trovate. Il risultato del modulo precedente viene memorizzato nel tracker, il quale consente di avere un controllo sulla memoria del bot, potendo anche recuperare informazioni passate. L'interprete si basa su criteri determinati (potendo specificare certe policy) al fine di decidere quali azioni si devono intraprendere in ogni fase. In seguito, l'azione scelta viene immagazzinata nel tracker, che, come prima accennato, conserva i dati sullo stato della conversazione e all'occorrenza può utilizzare informazioni necessarie in base alla cronologia del dialogo (quindi, anche slot e risultati). In seguito a ciò, viene generata una risposta per l'utente.



2.1 Obiettivo del progetto

Il chatbot sviluppato nel corso di questo progetto ha lo scopo di assistere l'utente durante la prenotazione di stanze d'hotel, andando a fornire una serie di informazioni di supporto alla scelta. A livello di base, un chatbot di questo tipo funziona interpretando o comprendendo le interazioni con i clienti e fornendo risposte pertinenti alla richiesta fatta. Inoltre, esso dovrà essere in grado di offrire una user experience piacevole senza mancare i suoi obiettivi primari, in termini di chiarezza e precisione delle informazioni fornite.

Tramite l'utilizzo della chat testuale e alcuni bottoni, l'utente sarà in grado di ottenere informazioni di varia natura, come le attrazioni e i luoghi di maggiore interesse legati ad una destinazione specifica, o le caratteristiche di determinate stanze d'hotel. Infine, il bot darà al cliente anche la possibilità di effettuare la prenotazione, validando automaticamente la richiesta fatta dall'utente.

3. Strumenti utilizzati

3.1 XAMPP

Dopo aver individuato con chiarezza le varie features da implementare nel chatbot, siamo passati alla definizione delle strutture finalizzate alla persistenza delle informazioni. Inizialmente si voleva procedere consultando delle API online legate a noti servizi di prenotazione di alloggi e camere (Booking.com, AirB&B, ecc.), ma ciò è risultato non praticabile a causa dei costi legati all'utilizzo di tali servizi web. Perciò, è stato creato un Database sfruttando le funzionalità offerte dal software open-source XAMPP.

Di conseguenza, considerando le feature del bot, sono state incluse nella base dati le seguenti colonne:

- **room_id**: Identificativo (Primary Key) intero della singola stanza di hotel
- **room_type**: Campo testuale, indicante il tipo della stanza in questione (Standard, Deluxe, Presidential)
- **reservation**: Campo testuale, contenente una stringa in formato JSON, necessaria per tener traccia delle varie prenotazioni effettuate dagli utenti. Ogni elemento dell'array è caratterizzato dai campi: check_in, check_out, reference (nome completo dell'utente che ha effettuato la prenotazione), kids (numero di bambini prenotati) e adults (numero di adulti prenotati).

room_id	room_type	reservations
1	Standard	<pre>{ "res": [{ "check_in": "06/12/2023", "check_out": "10/12/2023", "reference": "Patrick Bateman", "kids": 0, "adults": 2 }, { "check_in": "21/12/2023", "check_out": "27/12/2023", "reference": "Beatrix Kiddo", "kids": 2, "adults": 2 }, { "check_in": "30/12/2023", "check_out": "03/01/2024", "reference": "Alessandro Bedetta", "kids": 0, "adults": 1 }] }</pre>

Per stabilire la connessione a questa base dati, è stata sfruttata la libreria MySQL Connector per Python. Quest'ultima fornisce una serie di metodi utili per interagire con il database selezionato; in particolare, nel nostro caso, sono stati sfruttati i metodi e gli oggetti:

- **connect():** Metodo finalizzato ad istaurare la connessione con la base dati una volta prese in input le credenziali di quest'ultima.
- **cursor:** Si tratta di un costrutto che permette di gestire le interrogazioni al database, fornendo un modo per iterare i record delle tabelle prodotte dalle query.
- **commit():** Metodo necessario per eseguire materialmente le modifiche sulle tabelle del DB.
- **close():** Una volta stabilita la connessione, questo metodo permette di terminarla.

```
# Connection to the database
def db_connect():
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database='db_rasa_cb',
                                             user='root',
                                             port=3306)

        if connection.is_connected():
            print("DB is connected!")
            cursor = connection.cursor(buffered=True)
            cursor.execute("select database();")
            query = ("SELECT * FROM room")
            cursor.execute(query)

            return connection, cursor

    except Error:
        print("Some problem occurred while connecting to the DataBase")
```

3.2 API Wikipedia

Poiché l'utente può richiedere al chatbot informazioni su attrazioni o località, viene utilizzata una [libreria Python wrapper dell'API Wikimedia](#) così che il chatbot possa recuperare i dettagli associati da Wikipedia. Infatti, gli elementi considerati sono: il titolo della pagina, una parte della descrizione e un'immagine.

3.3 Ngrok

Ngrok è un servizio cross platform che fornisce agli utenti la possibilità di esporre su un server internet un ambiente di sviluppo locale, con minimo sforzo. Il software permette di mascherare il proprio indirizzo IP, andando ad hostare il servizio su un subdominio di ngrok.com. Nel nostro caso, il software è stato utilizzato per collegare il backend rasa con un bot Telegram.

4. Implementazione

Il progetto è stato strutturato in modo da rispettare la suddivisione logica in aree tematiche, legate all'ambito applicativo del bot. In particolare, le componenti del progetto sulle quali è stato concentrato il nostro lavoro sono:

- **nlu.yml**: questo file contiene gli intent necessari alla fase di training, atta a far sì che il chatbot sia in grado di comprendere le richieste, indipendentemente dal modo con cui vengono formulate dall'utente.
- **domain.yml**: questo file presenta al suo interno tutte le nozioni legate al dominio operativo, necessari al corretto funzionamento del bot.

Questi concetti vengono rappresentati tramite:

1. *Intents*, i quali sono i medesimi presenti nel file nlu.yml;
2. *Entities*, cioè porzioni di informazioni strutturate atte a rappresentare concetti presenti nel messaggio dell'utente. Le entità possono essere singole o composte, per esempio "date" che presenta al suo interno due ruoli distinti, "checkin" e "checkout".
3. *Slots*, rappresentano le aree di memoria del bot e vengono sfruttate da quest'ultimo al fine di tener traccia di quanto detto durante la conversazione.
4. *Forms*, le quali sono un modo per raccogliere i valori da assegnare agli slot in base agli input dell'utente, secondo una struttura definita.
5. *Responses*, che identificano le risposte possibili del bot.
6. *Actions*, ovvero una lista delle azioni che il bot può eseguire (ad esempio le validazioni e il processamento di informazioni).

```
entities:
- attraction
- people:
  roles:
    - adults
    - kids
- date:
  roles:
    - checkin
    - checkout
- contacts:
  roles:
    - email
    - phno
- room

forms:
  book_room_form:
    required_slots:
      - adults
      - kids
      - checkin
      - checkout
      - email
      - phno
      - room
  attraction_form:
    required_slots:
      - attraction
  name_form:
    required_slots:
      - first_name
      - last_name

actions:
- utter_faq_prompt
- utter_goodbye
- utter_greet
- utter_default
- action_nearest_attractions
- action_check_rooms
- utter_deluxe_details
- utter_standard_details
- utter_presidential_details
- utter_breakfastmenu
- utter_cancellationpolicy
- utter_check_in
- utter_check_out
- utter_childrenage
- utter_roomcount
- utter_breakfasttime
- utter_refundable
- utter_receptiontime
- utter_petpolicy
- utter_reservationconfirm
- utter_contactnumber
- utter_faq_prompt
- utter_ask_continue
- utter_ask_attraction
- utter_ask_first_name
- utter_ask_last_name
- utter_ask_adults
- utter_ask_kids
- utter_ask_checkin
- utter_ask_checkout
- utter_ask_email
- utter_ask_phno
- utter_submit_name_form
- utter_submit_book_room_form
- action_book_room_details
- action_reset
- action_ask_room
- validate_book_room_form
- validate_attraction_form
- validate_name_form
```

- **stories.yml**: file contenente i dati di training necessari per l'addestramento del modello e alla gestione della conversazione. Esse possono essere sfruttate in modo che il bot riesca a generalizzare diversi pattern di dialogo.

- **rules.yml**: in maniera simile alle storie, queste regole devono essere rispettate in qualunque scenario a prescindere dal tipo di conversazione (ad esempio, all'inizio vengono sempre richieste le generalità dell'utente).
- **actions.py**: file contenente le classi e i metodi utili all'elaborazione delle informazioni, identificati dagli stessi nomi presenti nel domain.

Una volta considerato il design dell'applicazione, si può procedere con l'analisi dei diversi ambiti applicativi. Per ognuna di queste aree tematiche, sono previsti specifici intent, entità e azioni.

4.1 Nome e cognome

Appena avviato il chatbot, alla prima interazione, vengono richiesti il nome ed il cognome all'utente, il quale, oltre a fornire tali informazioni, dovrà anche confermare in via definitiva la correttezza delle generalità. Per implementare tale funzionalità, è prevista una form con due slot diversi, uno per il nome ed uno per il cognome. La validazione delle generalità avviene tramite un'azione specifica, che esegue un controllo sul numero dei caratteri inseriti.

```
def validate_first_name(
    self,
    slot_value: Text,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Text, Any]:
    """Validate `first_name` value."""
    name=slot_value
    if len(name)<2:
        dispatcher.utter_message(text=f"You may have done a typo! Retry.")
        return {"first_name": None}
    else:
        return {"first_name": name}
```

Eseguito questo step, il chatbot presenta le proprie funzionalità all'utente, le quali consistono in:

- Informazioni relative a paesi e attrazioni vicine
- FAQ (policy di rimborso, orari reception, ecc.)
- Informazioni dettagliate sulle stanze
- Prenotazione di una stanza
- Riepilogo della prenotazione effettuata

4.2 Attrazioni

L'utente può richiedere maggiori dettagli riguardo alle attrazioni di maggior rilievo presenti nell'area circostante l'hotel; a tale richiesta, dal chatbot viene presentato un elenco, comprensivo di immagini e descrizioni, delle località maggiormente apprezzate dai turisti. Infatti, è previsto un database statico che contiene solo 6 Paesi, ovvero Loreto, Sirolo, Recanati, Ancona,

Castelfidardo, Porto Recanati, e, per ognuno di questi, si ricavano le descrizioni e la relativa immagine dalla pagina di Wikipedia. In dettaglio, è stato implementato un metodo che consente di consultare le API di Wikipedia, specificando per ogni località un certo numero di caratteri e l'immagine, che devono essere considerati.

```
# using the wiki api, some info are stored in a json
def api_wiki(PAGE,end_content,indx):
    wikipedia.set_lang("en")
    city = wikipedia.page(title=PAGE)
    title = city.title
    content = city.content[:end_content]
    image = city.images[indx]
    json_data = {"title":title, "content":content, "image":image}
    return json_data
```

```
def run(self,dispatcher,tracker,domain):
    loreto = api_wiki("Loreto, Marche",1072, 1)
    sirolo = api_wiki("Sirolo", 1284, 3)
    recanati = api_wiki("Recanati", 1193, 2)
    ancona = api_wiki("Ancona", 562, 1)
    castelfidardo = api_wiki("Castelfidardo", 737, 0)
    porto_recnati = api_wiki("Porto Recanati", 798, 3)
    attractions = [loreto,sirolo,recanati,ancona,castelfidardo,portorecanati]

    for att in attractions:
        msg="\n\n"+att["title"]+"\n"+att["content"]+"\n\n"
        dispatcher.utter_message(text=msg,image=att["image"])
    return []
```

A seguito di ciò, l'utente può richiedere maggiori informazioni riguardanti una singola attrazione e si procede con la validazione del valore inserito. A tal punto, se questo è ammesso, allora viene restituito un link relativo del sito della Riviera del Conero. Naturalmente, è possibile ottenere le informazioni su diverse attrazioni, in maniera sequenziale; invece, se la richiesta riguarda un luogo non ammesso, viene tornato un messaggio di default, insieme ad un link generico della Riviera del Conero.

```
@staticmethod
def attraction_db() -> List[Text]:
    """Database of supported attractions"""

    return ["loreto","recanati","sirolo","porto recanati","castelfidardo","ancona"]
```

```

def validate_attraction(
    self,
    slot_value: Any,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Text, Any]:
    """Validate `attraction` value."""

    if slot_value.lower() not in self.attraction_db():
        dispatcher.utter_message(text=f"I didn't find anything. \nBut you can find more info")
        # return {"attraction": None}
    else:
        if slot_value.lower() == self.attraction_db()[0]:
            website = "https://www.rivieradelconero.info/en/loreto-a-christian-pilgrimage-des"
        elif slot_value.lower() == self.attraction_db()[1]:
            website = "https://www.rivieradelconero.info/en/item/recanati.php"
        elif slot_value.lower() == self.attraction_db()[2]:
            website = "https://www.rivieradelconero.info/en/sirollo-a-seaside-town-in-the-cone"
        elif slot_value.lower() == self.attraction_db()[3]:
            website = "https://www.rivieradelconero.info/en/porto-recanati-a-seaside-town-in-"
        elif slot_value.lower() == self.attraction_db()[4]:
            website = "https://www.rivieradelconero.info/en/music-and-culture-in-castelfidard"
        elif slot_value.lower() == self.attraction_db()[5]:
            website = "https://www.rivieradelconero.info/en/ancona-a-seaside-town-with-a-rich"
        dispatcher.utter_message(text=f"OK! Here some info about {slot_value}:\n" + website)
    return {"attraction": None}

```

4.3 FAQ

In un chatbot di questo tipo, certamente non può mancare la componente di gestione delle domande maggiormente richieste. In primo luogo, l'utente può richiedere al chatbot le FAQ, in modo da ottenere un elenco di ciò che si può domandare, come: visione delle stanze, prenotazione della stanza, conferma della prenotazione, limiti sulla prenotazione, orari della reception, contatto telefonico dell'hotel, policy sulla cancellazione della prenotazione e sul rimborso, orari e menù della colazione, check-in e check-out, la policy riguardante gli animali domestici, la regolamentazione sui minori.

4.4 Dettagli sulle stanze

L'utente può chiedere al chatbot di mostrare in maniera più generica le tipologie di stanze presenti in hotel e, grazie ad un'azione specifica, vengono ritornate le immagini delle camere, con i relativi nomi. Una volta che l'utente ha visionato l'elenco fornito, è possibile richiedere maggiori dettagli su una particolare stanza (ovviamente, una volta ottenute le informazioni aggiuntive, si potrà ripetere il procedimento per una diversa camera).

```

def run(self, dispatcher, tracker, domain):
    deluxe = {"title": "Deluxe", "image": "https://www.miasaigon.com/wp-content/uplo
    standard = {"title": "Standard", "image": "https://images.pexels.com/photos/1402
    presidential = {"title": "Presidential", "image": "https://th.bing.com/th/id/R.7
    room_type = [deluxe, standard, presidential]

    for r in room_type:
        msg = "\n\n" + r["title"] + "\n\n"
        dispatcher.utter_message(text=msg, image=r["image"])

    dispatcher.utter_message(text="If you desire any further information, " +
                                "feel free to ask for more details.")
    return []

```

4.5 Prenotazione

La caratteristica più importante del chatbot è rappresentata dalla possibilità di eseguire una prenotazione di una stanza in maniera autonoma ed automatica. Quando l'utente è pronto per procedere alla prenotazione, viene avviata una form che richiede 7 slot, mappati con le corrispondenti entità: numero di adulti, numero di bambini, data di check-in, data di check-out, e-mail, numero telefonico, stanza desiderata.

```

@staticmethod
def required_slots(tracker: Tracker) -> List[Text]:
    """A list of required slots that the form has to fill"""
    return ["adults", "kids", "checkin", "checkout", "email", "phno", "room"]

```

In dettaglio, il processo di prenotazione si compone di 4 passi fondamentali.

- Viene richiesto all'utente di inserire sia il numero degli adulti sia il numero di bambini che alloggeranno nell'hotel; per ogni stanza, sono ammessi al massimo 8 adulti e 6 bambini (sotto i 12 anni di età), i quali devono essere necessariamente accompagnati (perciò, non è possibile prenotare per un singolo bambino). Nella validazione di questi due slot, si verificano i vincoli di numero.

```

def validate_adults(
    self,
    slot_value: Any,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Any, Any]:
    """Validate `adults` value."""
    try:
        n_adults = int(slot_value)
        if (n_adults < 1 or n_adults > 8):
            dispatcher.utter_message(text=f"Number of adults allowed: 1 - 8! Please, retry.")
            return {"adults": None}
        else:
            return {"adults": n_adults}
    except:
        dispatcher.utter_message(text=f"This value (" + str(slot_value) +
                                     ") is not correct! Please, retry.")
        return {"adults": None}

```

```

def validate_kids(
    self,
    slot_value: Any,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Any, Any]:
    """Validate `kids` value."""
    try:
        if slot_value.lower() in ["no", "none", "any"]:
            return {"kids": 0}
        elif (int(slot_value) >= 0 and int(slot_value) < 7):
            n_kids = int(slot_value)
            return {"kids": n_kids}
        else:
            dispatcher.utter_message(text=f"Number of kids allowed: 0 - 6! Please, retry.")
            return {"kids": None}
    except:
        dispatcher.utter_message(text=f"This value (' + str(slot_value) +
                                     ") is not correct! Please, retry.")
        return {"kids": None}

```

- L'utente deve scegliere la data di check-in e quella di check-out, secondo uno dei quattro formati ammessi. Il processo di validazione termina con successo se le date inserite non sono riferite al giorno corrente o a quello passato, se il check-in è antecedente al check-out e se non ci sono errori di battitura.

```

def validate_checkout(
    self,
    slot_value: Text,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Text, Any]:
    """Validate `checkout` value."""
    date_str = slot_value
    present = datetime.now().date()
    try:
        check_in_date = str(tracker.get_slot("checkin"))
        check_in_date = datetime.strptime(check_in_date, '%d/%m/%Y').date()
        for fmt in ('%d/%m/%Y', '%d-%m-%Y', '%Y/%m/%d', '%Y-%m-%d'):
            try:
                date_obj = datetime.strptime(date_str, fmt).date()
                if date_obj < present:
                    dispatcher.utter_message(text=f"Check-out date cannot be past date! Please, retry.")
                    return {"checkout": None}
                elif date_obj < check_in_date:
                    dispatcher.utter_message(text=f"Check-out date cannot be prior to check-in date! Please, retry.")
                    return {"checkout": None}
                else:
                    date_obj = date_obj.strftime('%d/%m/%Y')
                    return {"checkout": date_obj}
            except ValueError:
                pass
    except:
        dispatcher.utter_message(text=f"Check-out date is not correct! Please, retry.")
        return {"checkout": None}

```

- A questo punto, il chatbot richiede i contatti dell'utente, ovvero l'indirizzo di posta elettronica ed un recapito telefonico (preferibilmente con prefisso); in tal caso, il processo di validazione sfrutta una funzione di controllo dell'e-mail (per esempio, nomeutente@dominio.it) e verifica la lunghezza del numero fornito.

```

def validate_email(
    self,
    slot_value: Text,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Text, Any]:
    """Validate `email` value."""
    mail = slot_value
    if check_mail(mail) == False:
        dispatcher.utter_message(text=f"You may have done a typo! Retry.")
        return {"email": None}
    else:
        return {"email": mail}

```

```

def validate_phno(
    self,
    slot_value: Text,
    dispatcher: CollectingDispatcher,
    tracker: Tracker,
    domain: DomainDict,
) -> Dict[Text, Any]:
    """Validate `phno` value."""
    number = slot_value
    if (len(number)<6 or len(number)>15):
        dispatcher.utter_message(text=f"Your number is not correct! Retry.")
        return {"phno": None}
    else:
        return {"phno": number}

```

- Avendo a disposizione tutte le informazioni necessarie, il chatbot sfrutta un'azione particolare per collegarsi al database; una volta stabilita la connessione, viene eseguita una query al fine di verificare quali sono le stanze libere nell'intervallo di tempo, deciso in precedenza (tenendo conto che il check-in avviene durante il pomeriggio, mentre il check-out in mattinata). Per facilitare l'esperienza dell'utente, vengono restituiti dei bottoni che indicano le stanze disponibili (*Standard*, *Deluxe*, *Presidential*) così che il cliente possa scegliere in maniera semplice ed intuitiva.

```

# Iterating over the rows, each room is checked
for(room_id,room_type,reservations) in cursor:
    # only reservation field is read
    json_res = json.loads(reservations)
    # flag used to check whether the room is available
    flag = True
    for res in json_res["res"]:
        date_string = res["check_in"]
        check_in_date = datetime.strptime(date_string, "%d/%m/%Y").date()
        date_string = res["check_out"]
        check_out_date = datetime.strptime(date_string, "%d/%m/%Y").date()

        check_in_new_date = datetime.strptime(new_check_in, "%d/%m/%Y").date()
        check_out_new_date = datetime.strptime(new_check_out, "%d/%m/%Y").date()

        # by this condition, the room's availability is verified
        if((check_in_new_date>=check_in_date and check_in_new_date<check_out_date)
        or(check_out_new_date<=check_out_date and check_out_new_date>check_in_date)):
            # setting the flag to false means that the room cannot be booked
            flag = False
            break

```

```
# checking if this room is free during that period
if flag:
    button = {"title": room_type, "payload": '/inform_room{"room":"' + room_type + '"}'}
    # room type only is shown to the user
    if (button not in rooms_available):
        rooms_available.append(button)
```

Tramite il processo di validazione, si verifica che il valore inserito sia coerente. Contemporaneamente, nel database si esegue un aggiornamento delle prenotazioni, all'interno del campo "reservations" (check-in, check-out, utente, numero di adulti, numero di bambini), corrispondente alla tipologia di stanza scelta. L'attività a livello del database viene validata in modo da non generare incoerenze ed errori, come perdite di informazioni, parsing del JSON errato ed ambiguità.

```
# by this condition, the room's availability is verified
if((check_in_new_date>=check_in_date and check_in_new_date<check_out_date)
or(check_out_new_date<=check_out_date and check_out_new_date>check_in_date)):
    # setting the flag to false means that the room cannot be booked
    flag = False
    break

# checking if this room is free during that period
if flag:
    try:
        new_res = {
            "check_in": new_check_in,
            "check_out": new_check_out,
            "reference": name,
            "kids": kids,
            "adults": adults
        }

        json_res["res"].append(new_res)
        updt_res = json.dumps(json_res, indent=3)
        # this query allows to update the reservation field relative to a particular id
        updt = ("UPDATE room SET reservations = %s WHERE room_id = %s")
        v = (updt_res, room_id)
        # Execute the SQL command
        cursor.execute(updt,v)
        # Commit your changes in the database
        connection.commit()
```

Infine, una volta che l'utente ha terminato la procedura, viene restituito un riepilogo della prenotazione, comprensivo del numero della stanza e del prezzo, grazie ad un metodo che calcola l'importo totale a partire dal numero di adulti, dal numero dei bambini e dalla tipologia di stanza scelta.

```
def unitPrice(room,n_adults, n_kids):

    if room == "Deluxe":
        price = ((n_adults*500)+(n_kids*100))
    elif room == "Standard":
        price = ((n_adults*50)+(n_kids*10))
    elif room == "Presidential":
        price = ((n_adults*1000)+(n_kids*200))
    return price
```

```

check_in=str(tracker.get_slot("checkin"))
check_out=str(tracker.get_slot("checkout"))
adults=str(tracker.get_slot("adults"))
kids=str(tracker.get_slot("kids"))
room=str(tracker.get_slot("room"))
first_name=str(tracker.get_slot("first_name"))
last_name=str(tracker.get_slot("last_name"))
phno=str(tracker.get_slot("phno"))
email=str(tracker.get_slot("email"))
name = str(first_name) + " " + str(last_name)
unit_price = unitPrice(room,int(adults),int(kids))
n_days = (datetime.strptime(check_out, '%d/%m/%Y').date() - datetime.strptime(check_in, '%d/%m/%Y').date()).days
price = str(n_days*unit_price)

```

```

# by this condition, the room's availability is verified
if((check_in_new_date == check_in_date and check_out_new_date==check_out_date)
and(name == res["reference"])):
    room = room + " n° " + str(room_id)

```

```

message="BOOKING DETAILS"+"\\n"+...
"Name: "+name+"\\n"+...
"Check-in Date: "+check_in+"\\n"+...
"Check-out Date: "+check_out+"\\n"+...
"No. of Adults: "+adults+"\\n"+...
"No. of Kids: "+kids+"\\n"+...
"Room: "+room+"\\n"+...
"Phone Number: "+phno+"\\n"+...
>Email: "+email
dispatcher.utter_message(message)
dispatcher.utter_message(text="Total price: € " + price)

```

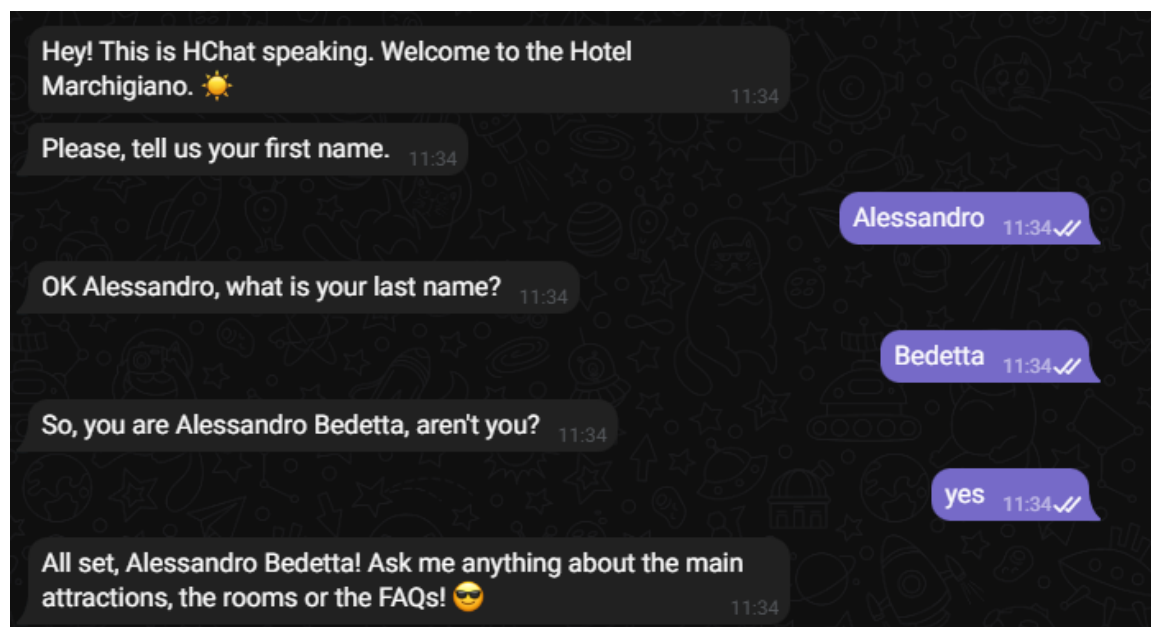

5. Test

Per eseguire i test e verificare il funzionamento corretto, è stato deciso di usare Telegram. In questo modo è possibile rendere il chatbot disponibile agli utenti attraverso un'interfaccia più familiare, i quali possono accedere semplicemente dall'applicazione per smartphone o per desktop. La scelta di Telegram è dettata sostanzialmente dalla facilità di caricamento del chatbot, mediante una procedura interna all'applicazione stessa; inoltre, sfruttando il servizio ngrok, grazie al quale si genera un url che consente di comunicare attraverso la porta 5005, si rende possibile la comunicazione con il bot. Naturalmente, bisogna impostare le credenziali all'interno del file *credentials.yml* in cui si inserisce il token di accesso (oscurato per motivi di sicurezza), l'host (url precedentemente creata) e il nome del bot.

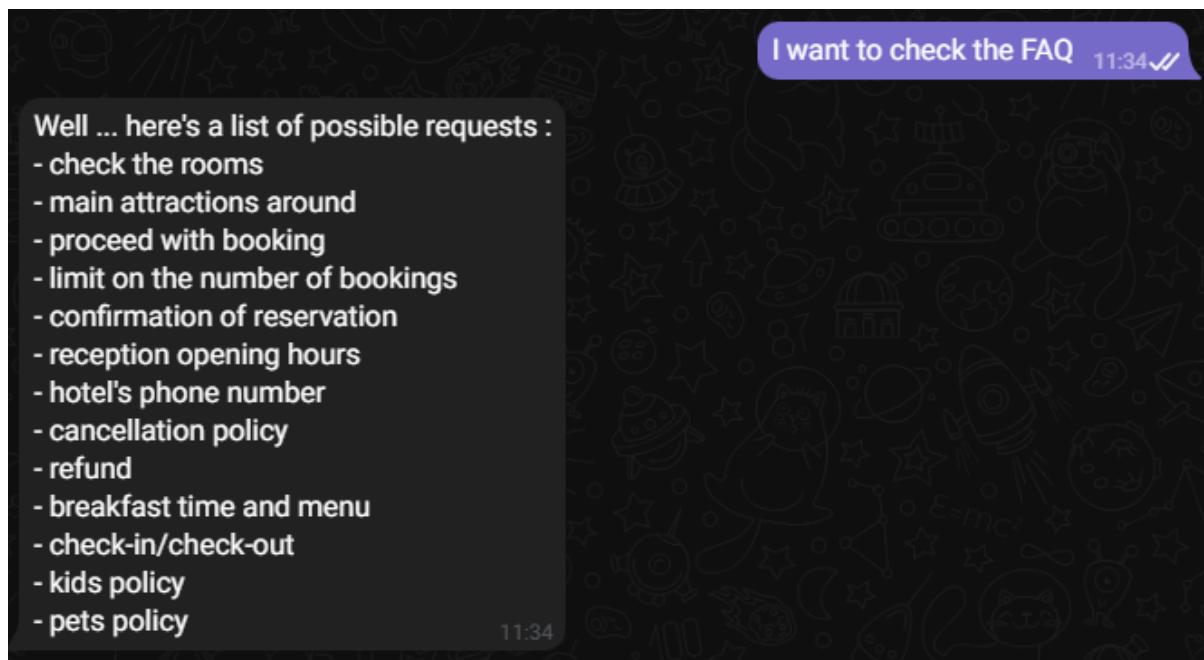
```
telegram:  
  access_token: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
  verify: HotelChat_bot  
  webhook_url: "https://195a-79-37-71-17.eu.ngrok.io/webhooks/telegram/webhook"
```

A questo punto si può procedere alla fasi di test della conversazione andando ad analizzare i diversi scenari possibili.

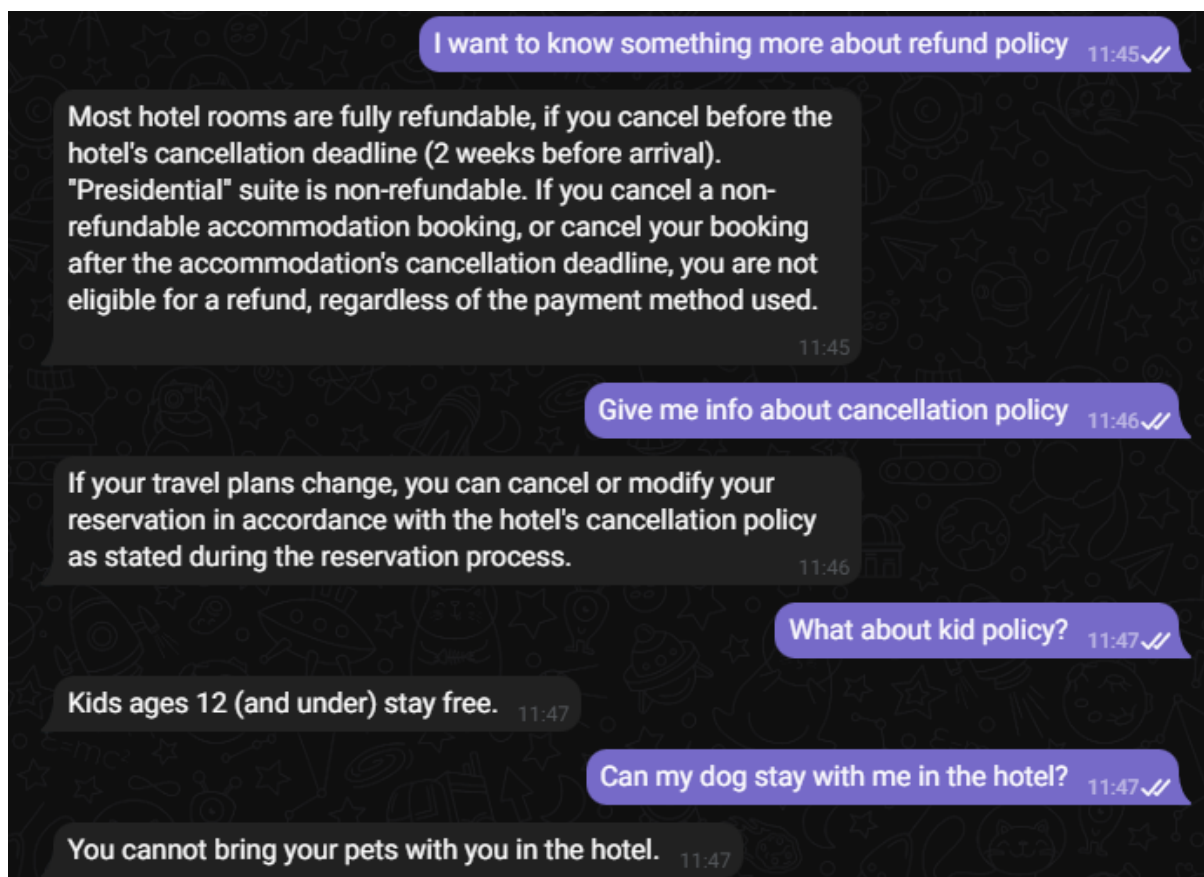
In un primo test, all'utente viene richiesto di identificarsi tramite il proprio nome e cognome; successivamente egli dovrà dare conferma delle generalità appena fornite.



Fatto ciò, il cliente, per avere una panoramica su ciò che il bot è in grado di fare, può richiedere quali sono le domande più frequentemente poste. A tal richiesta, il bot fornirà un elenco contenente le varie funzionalità.



L'utente può domandare alcune informazioni ausiliarie, come quelle riguardanti il rimborso, la cancellazione di una prenotazione, la policy sui minori e quella sugli animali.



Il cliente potrebbe anche essere interessato alle località turistiche e alle attrazioni nella zona circostante l'hotel.

I want to know what are the main attractions around 11:50 ✓

Loreto, Marche

Loreto (la-RET-oh, US also la-RAY-toh, Italian: [lo're.to]) is a hill town and comune of the Italian province of Ancona, in the Marche. It is most commonly known as the seat of the Basilica della Santa Casa, a popular Catholic pilgrimage site. 11:50

== Location ==

Loreto is located 127 meters (417 feet) above sea level on the right bank of the Musone river and 22 kilometers (14 miles) by rail south-southeast of Ancona; like many places in the Marche, it provides good views from the Apennines to the Adriatic. 11:50

== Main sights ==

The city's main monuments occupy the four sides of the piazza: the college of the Jesuits; the Palazzo Comunale (formerly the Palazzo Apostolico), designed by Bramante, that houses an art gallery with works of Lorenzo Lotto, Vouet and Annibale Carracci as well as a collection of maiolica, and the Shrine of the Holy House (Santuario della Santa Casa). It also boasts a massive line of walls, designed by the architect (and military engineer) Antonio da Sangallo the Younger, which were erected from 1518 and reinforced in the 17th century. 11:50



Sirolo

Sirolo is a comune (municipality) in the Province of Ancona in the Italian region of Marche. As of 28 February 2009 it had a population of 3,747 inhabitants and an area of 16.68 square kilometres (6.44 sq mi). The town is situated near Monte Conero and in medieval times it was a castle which belonged to the Conti Cortesi family. It is a tourist destination, especially in summer times and it has continuously been awarded one of the eleven blue flags in the region of Marche since 1994. It features a restored medieval town centre, which culminates with a particular little square with view on the sea and on the Monte Conero. As tourist destination, Sirolo has many restaurants and bars. From the main square, there is a 10-minute walk to several beaches. 11:50

== Beaches ==

Spiaggia Urbani is located just above the centre and offers amenities such as sunbeds, umbrellas and kayaks rentals, bars and restaurant. 11:50

Spiaggia San Michele is located just next to Urbani and, while still offering the same amenities, it has a wilder look. 11:50

Spiaggia delle Due Sorelle is a beach of white round pebbles which can be seen through the crystal clear water. The name of this beach, which means "two sisters" is due to the twin sea stack raising from the water, which look like two nuns praying. 11:50



Recanati

Recanati (Italian pronunciation: [reka'na.ti]) is a town and comune in the Province of Macerata, in the Marche region of Italy. Recanati was founded around 1150 AD from three pre-existing castles. In 1290 it proclaimed itself an independent republic and, in the 15th century, was famous for its international fair. In March 1798 it was conquered by Napoleon Bonaparte. 11:50

The elongated historic center extends from one end to the other for over 200 metres and occupies an area of about 35 hectares. Its linear structure distinguishes it from most of the neighboring centers with a concentric plan, in which the inhabited area has extended from a central square. Along the margins of the central road, connecting the ancient housing clusters, there are numerous aristocratic buildings, for the most part on three floors, built by merchants or landowners. 11:50

It is the hometown of the tenor Beniamino Gigli and the poet Giacomo Leopardi, which is why the town is known to some as "the city of poetry". Famous medieval Ashkenazi Kabbalist rabbi Menahem Recanati flourished here in the 13th century. Teatro Persiani named after Giuseppe Persiani an opera composer, born in 1799, is located in the town. 11:50



Ancona

Ancona (, also US: , Italian: [an'ko:na] (listen)) is a city and a seaport in the Marche region in central Italy, with a population of around 101,997 as of 2015. Ancona is the capital of the province of Ancona and of the region. The city is located 280 km (170 mi) northeast of Rome, on the Adriatic Sea, between the slopes of the two extremities of the promontory of Monte Conero, Monte Astagno and Monte Guasco. Ancona is one of the main ports on the Adriatic Sea, especially for passenger traffic, and is the main economic and demographic centre of the region. 11:50



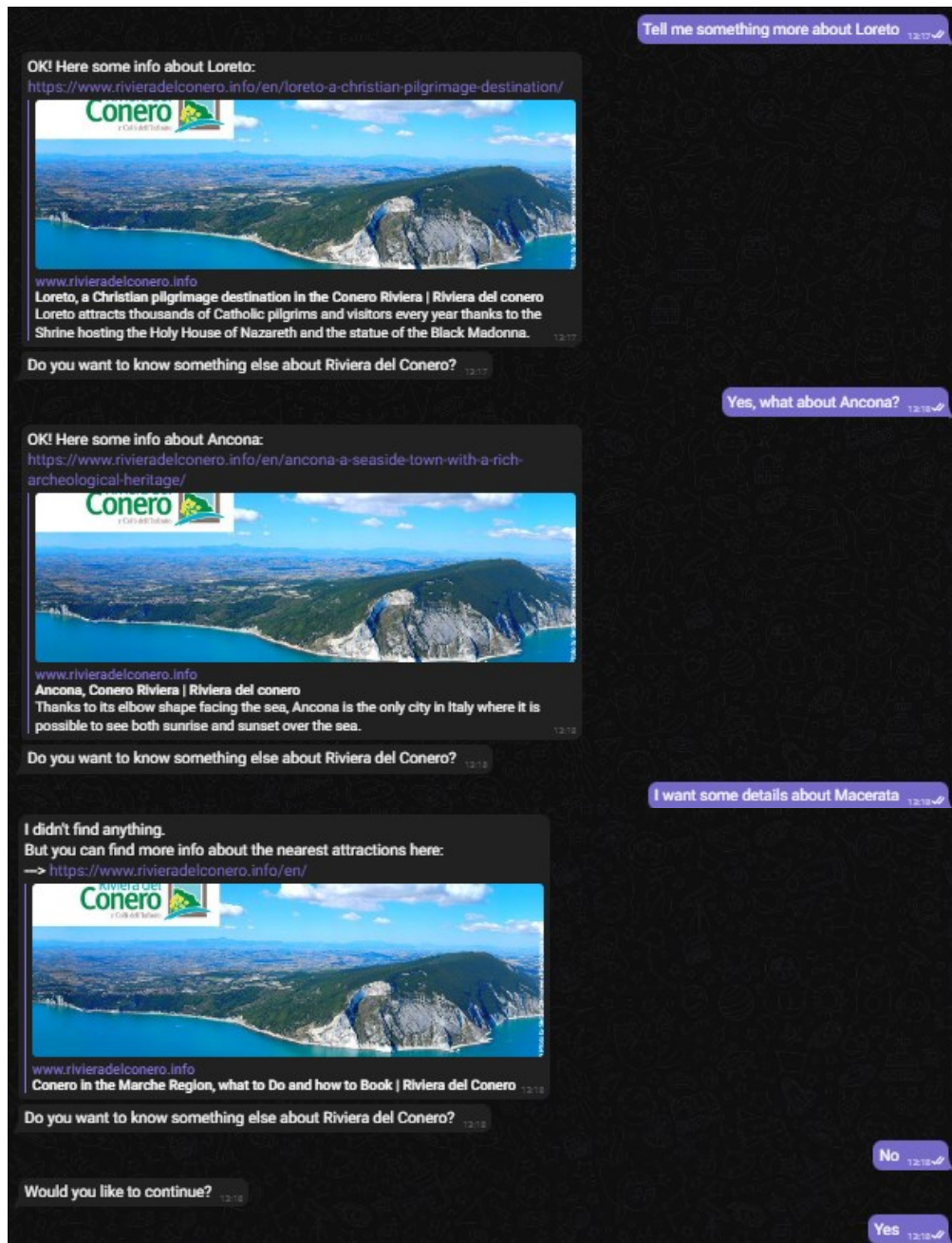
Castelfidardo

Castelfidardo (Marchigiano: Castello) is a town and comune in the province of Ancona, in the Marche region of central-eastern Italy. 11:50

It is remembered for a Piedmontese victory over an army composed of foreign volunteers defending the Papal States, on September 18, 1860. The town's Museum of the Risorgimento, in the palazzo Mordini, commemorates the battle and places it in the wider context of the Risorgimento as a whole. It houses artifacts and documents of the period, including around 130 loans from private collections or other museums. In addition, Castelfidardo is home to a number of Renaissance-era buildings, including 'il Palombarone' a 1580 construction which underwent extensive renovations in the early 21st century. 11:50



Se il cliente è particolarmente colpito da una delle attrazioni mostrate dal bot, potrà richiedere maggiori dettagli in merito ad una o più località. Nel caso in cui si selezioni un Paese non presente nell'elenco precedente, viene tornato un link che si riferisce al sito web della Riviera del Conero. A tal punto, l'utente può decidere di continuare direttamente la conversazione o specificare la volontà di non voler avere altre informazioni sulle attrazioni.



L'utente potrebbe voler sapere quali sono le tipologie di stanze che l'hotel offre, perciò il chatbot mostra i nomi delle tre stanze (*Standard*, *Deluxe*, *Presidential*) e le relative immagini. In aggiunta, si può richiedere maggiori dettagli su un alloggio in particolare.

I would like to see what the rooms look like 1:4:20 ✓

Deluxe 1:4:20



Standard 1:4:20



Presidential 1:4:20



If you desire any further information, feel free to ask for more details. 1:4:20

I want to know something more about deluxe 1:5:38 ✓

Step into our deluxe hotel room and experience the ultimate in luxury and comfort. From the moment you step inside, you'll be greeted with an air of elegance and sophistication. Our deluxe room is the perfect retreat for travelers who seek a little extra indulgence during their stay. The room is tastefully decorated with modern furnishings and luxurious finishes, making it a stylish and comfortable space for guests to relax and recharge. The large windows offer breathtaking views of the surrounding city or landscape, depending on the location of the hotel. The room features a large flat-screen TV with cable channels, high-speed internet access, a mini-fridge, and a coffee maker. The spacious marble bathroom is stocked with plush towels, complimentary toiletries, and a luxurious bathtub to soak in after a long day. Access to the boardroom, and Grand Club serving breakfast and evening refreshments. Complimentary bottled mineral water.

Price:

Adults: 500 €/night

Kids: 100 €/night 1:5:38

More info about Standard room 1:5:55 ✓

A standard hotel room is the quintessential accommodation option for travelers seeking a comfortable and convenient stay away from home. Upon entering a standard hotel room, guests are greeted with a cozy and inviting atmosphere. The room is usually decorated with tasteful furnishings and soothing colors, creating a warm and inviting ambiance. In addition to the sleeping area, the room typically includes a seating area with a comfortable armchair or sofa, a work desk and chair, and a flat-screen television with cable or satellite channels. Guests will also find a well-stocked mini-bar or refrigerator, providing a variety of refreshments and snacks for their convenience. For added convenience, the room may also feature an iron and ironing board, a safe, and free Wi-Fi.

Price:

Adults: 50 €/night

Kids: 10 €/night 1:5:55

Tell me more about Presidential suite 1:6:01 ✓

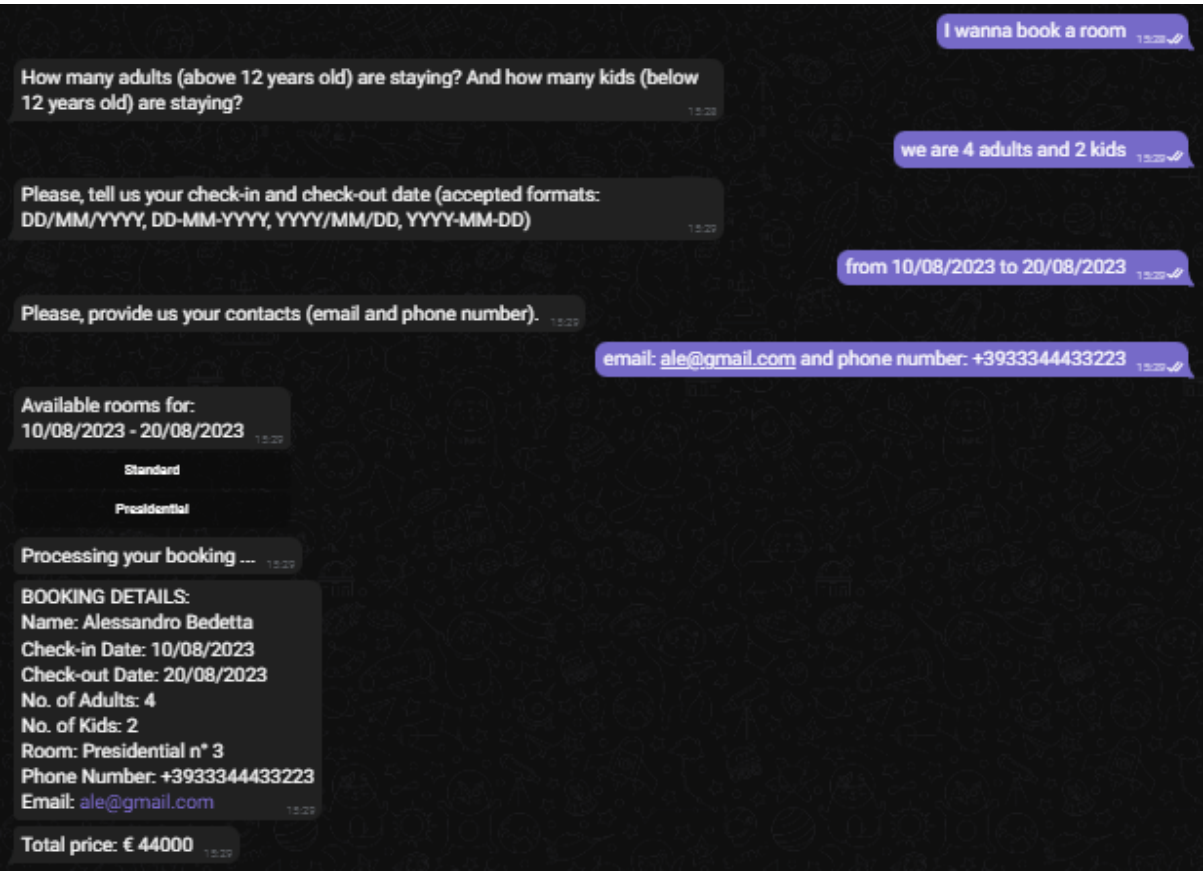
An exclusive 500 sqft room that redefines luxury. As you enter the suite, you are immediately struck by the spaciousness of the room. The living area is tastefully furnished with plush sofas, armchairs, and a grand piano. Large windows bathe the space in natural light, adding to the already inviting atmosphere. The master bedroom is equally impressive, featuring a king-sized bed dressed in the finest linens and a stunning en-suite bathroom with a Jacuzzi tub and a separate shower. The room also boasts a walk-in closet, ensuring ample storage space for your belongings. The Presidential Hotel Suite comes equipped with all the latest technology, including high-speed Wi-Fi, flat-screen televisions, and state-of-the-art sound systems. You can relax and unwind while enjoying your favorite movies or music from the comfort of your luxurious surroundings. In addition to the superb amenities, the suite provides a range of personalized services to cater to your every need.

Price:

Adults: 1000 €/night

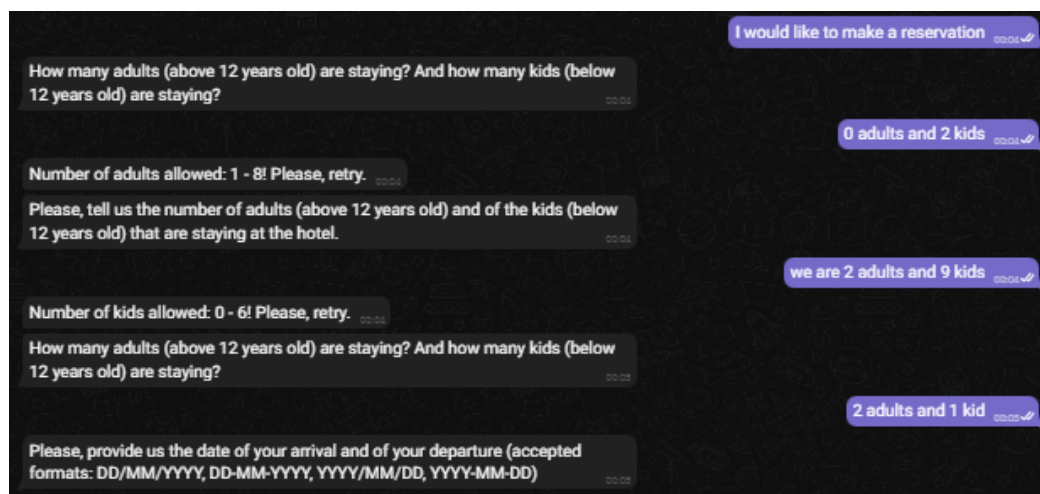
Kids: 200 €/night 1:6:01

Quando l'utente è pronto, si può procedere con la prenotazione di una stanza; come precedentemente descritto, il chatbot richiede le informazioni necessarie, provvedendo a rispondere in maniera coerente, sia quando viene superata correttamente la fase di validazione sia quando si hanno gli errori.

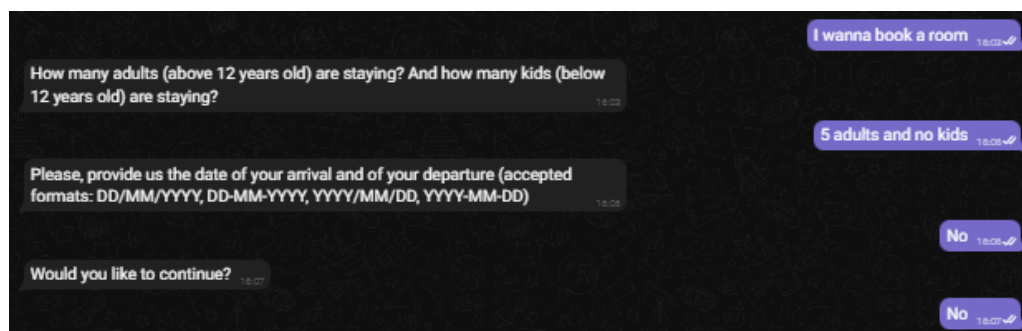


room_id	room_type	reservations
3	Presidential	<pre>{ "res": [{ "check_in": "15/07/2023", "check_out": "24/07/2023", "reference": "Jack Sparrow", "kids": 2, "adults": 1 }, { "check_in": "20/12/2023", "check_out": "06/01/2024", "reference": "Stefano Pioli", "kids": 0, "adults": 4 }] }</pre>
3	Presidential	<pre>{ "res": [{ "check_in": "15/07/2023", "check_out": "24/07/2023", "reference": "Jack Sparrow", "kids": 2, "adults": 1 }, { "check_in": "20/12/2023", "check_out": "06/01/2024", "reference": "Stefano Pioli", "kids": 0, "adults": 4 }, { "check_in": "10/08/2023", "check_out": "20/08/2023", "reference": "Alessandro Bedetta", "kids": 2, "adults": 4 }] }</pre>

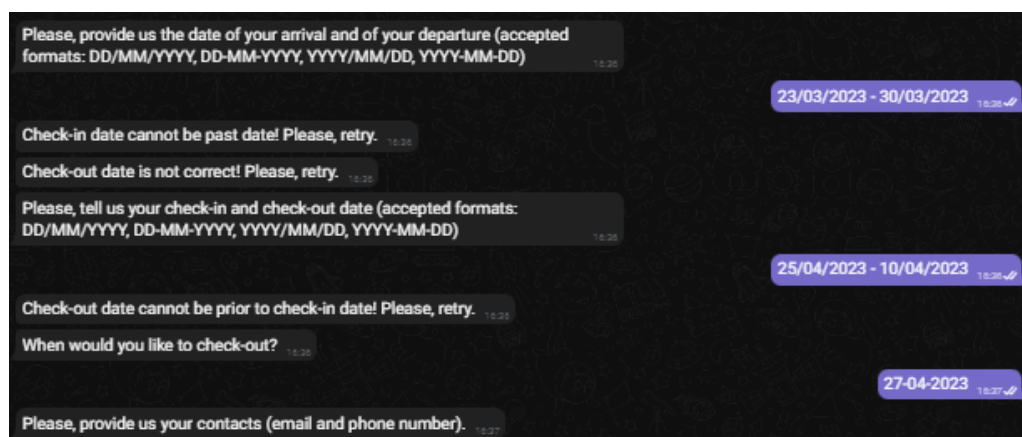
Quando il chatbot richiede il numero di adulti e di bambini, se viene inserito un numero non corretto (quantità negative o fuori dal limite consentito), viene ritornato un messaggio con il relativo errore, riprendendo la form.



L'utente può anche scegliere di interrompere la prenotazione, però, prima di poter effettivamente interagire nuovamente con il chatbot, deve confermare la propria decisione o continuare con la form.



Infine, se le date di check-in e check-out sono errate, non coerenti o riferite al passato, il chatbot comunica l'errore e chiede di nuovo tale informazione.



I wanna book a room

Please, tell us the number of adults (above 12 years old) and of the kids (below 12 years old) that are staying at the hotel.

1 adult

Please, tell us how many kids (below 12 years old) are staying at the hotel.

0

Please, provide us the date of your arrival and of your departure (accepted formats: DD/MM/YYYY, DD-MM-YYYY, YYYY/MM/DD, YYYY-MM-DD)

i will be there from 2023-04-15 to 2023-03-17

Check-out date cannot be past date! Please, retry.

When would you like to check-out?

2023-04-17

Please, provide us your contacts (email and phone number).

email: patrick_bateman@gmail.com and phone: [+445544334454](tel:+445544334454)

Available rooms for:
15/04/2023 - 17/04/2023

Standard

Deluxe

Presidential

Processing your booking ...

BOOKING DETAILS:
Name: Patrick Bateman
Check-in Date: 15/04/2023
Check-out Date: 17/04/2023
No. of Adults: 1
No. of Kids: 0
Room: Deluxe n° 2
Phone Number: [+445544334454](tel:+445544334454)
Email: patrick_bateman@gmail.com

Total price: € 1000

6. Conclusioni e sviluppi futuri

Il progetto ha permesso di approfondire diversi aspetti legati agli assistenti virtuali, basati sul framework RASA, soprattutto nell'ambito del customer service. Questo tipo di soluzioni, in futuro, potrebbe rappresentare una possibilità aggiuntiva per migliorare la customer experience del cliente e, di conseguenza, sarà sempre più necessaria un'attività di manutenzione ad essa legata.

Per quanto riguarda possibili sviluppi futuri relativi ad un chatbot di questo tipo, si potrebbero apportare miglioramenti a diversi aspetti. Essendo fondamentale per la gestione dei dati, il database dovrebbe essere progettato in un modo più efficiente, per esempio mediante l'uso di indici, soprattutto in ottica enterprise. Infatti, un sistema di questo tipo potrebbe essere scalato facilmente, includendo più database facenti capo a diverse strutture alberghiere. Inoltre, si potrebbero memorizzare anche informazioni relative ai clienti, in ottica analytics, e al fine di evitare incoerenze, ad esempio fornendo codici univoci ad ogni utente. Riguardo alla prenotazione delle stanze, poiché nella realtà spesso la capienza non è fissa, si potrebbe considerare un approccio più dinamico sui limiti delle camere, per esempio stabilendo un numero massimo e un numero minimo di adulti/bambini per ognuna di esse. Sempre in merito alle funzionalità del database, si potrebbe pensare di rendere persistente anche la gestione delle informazioni relative alle attrazioni, inserendo opportune tabelle all'interno del DB (immagini, descrizioni, link). Dato che durante la fase iniziale vengono richieste le generalità del cliente, si potrebbe implementare un meccanismo di login in grado di autenticare l'utente, a partire da credenziali collegate ad un account social o di posta elettronica.

Infine, come ulteriore miglioramento al chatbot, potrebbero essere modificati i dati di training in modo da garantire comportamenti più vicini all'uomo e anche distinguere diversi scenari per diversi clienti.