ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA

CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

Chatbot ed Elaborazione Naturale del Linguaggio.

Progettazione e realizzazione di un assistente sanitario

Elaborato in

Sistemi Embedded E Internet-of-things

Relatore Presentata da

Prof. Alessandro Ricci Andrea Bondanini

Correlatore

Dmytro Vlasenko

Sessione III

Anno Accademico 2017 - 2018

# Introduzione

Negli ultimi anni, la forte crescita dell’utilizzo di applicazioni inerenti alla messaggistica istantanea unita a sorprendenti miglioramenti attuati nel campo dell’intelligenza artificiale[[1]](#footnote-1), hanno portato ad un importante incremento riguardante lo sviluppo di chatbot. Un chatbot è un software sviluppato per emulare una conversazione con una persona, il quale può essere progettato per rispondere a semplici input tramite una scansione delle parole chiave, oppure implementando un sistema intelligente in grado di analizzare, comprendere e trarre il significato dalle frasi che un essere umano utilizza nella comunicazione di tutti i giorni. Nonostante l’ambiguità che caratterizza il linguaggio umano, gli avanzati algoritmi di *Natural Language Processing* permettono, tramite l’elaborazione di un’espressione in linguaggio naturale, di eseguire compiti come la traduzione automatica, correzione sintattica, estrazione dei concetti principali fino all’analisi dei sentimenti.

I possibili ambiti di impiego sono molteplici e diametralmente opposti, partendo da assistenti virtuali che trovano spazio all’interno di dispositivi domotici, fino a sistemi per la gestione del *Customer Relationship Management*. La digitalizzazione di vari aspetti inerenti alla nostra vita ha apportato una radicale modifica in innumerevoli settori, da quelli intrinsecamente propensi all’integrazione con agenti intelligenti a quelli più impermeabili all’innovazione, come la sfera dell’assistenza sanitaria. L’utilizzo di un *HealthBot* in campo sanitario non si pone certamente come obiettivo la sostituzione della terapia umana, poiché raggiungere le conoscenze di un medico in “carne e ossa” e ricreare il rapporto empatico tra medico e paziente sarebbe utopistico, tuttavia il suo impiego potrebbe portare vantaggi su due fronti; dal lato del professionista, per offrirgli strumenti e tecnologie in grado di supportarlo in una diagnosi e permettergli una gestione amministrativa pratica, dall’altra parte agevolare il paziente all’accesso di prestazioni mediche, il tutto comunicando in modo naturale attraverso uno smartphone o un PC [1].

Il progetto in esame si pone come obiettivo lo studio della fattibilità di implementazione di un assistente sanitario intelligente, analizzando i vantaggi apportati e le criticità; nel dettaglio è stato realizzato un prototipo in grado di comunicare con il paziente, per concedere ad esso la possibilità di interagire con un assistente virtuale 24 ore su 24, senza bisogno di intervento umano e predisponendo un interfacciamento con i canali di comunicazione più utilizzati a fronte di un’eventuale futura messa in produzione (*Telegram*, *Facebook Messenger*, *Skype)*. In particolare viene implementato un servizio che vuole, in primis, facilitare tutti i processi di prenotazione per le prestazioni specialistiche in regime di SSN[[2]](#footnote-2), e in secondo luogo implementare un servizio FAQ[[3]](#footnote-3), in grado di rispondere ad eventuali dubbi e dispensare consigli in riferimento ad una eventuale prestazione fissata.

La tesi è redatta nella seguente modalità: nel primo capitolo viene effettuata un’introduzione del problema, prendendo in considerazione tutti i casi d’uso che sarebbero oggetto di miglioria a fronte di una possibile implementazione del progetto in esame. In particolare vengono elencate le possibilità attuali d’accesso alle prestazioni mediche del paziente mettendone in luce gli ostacoli; successivamente si espone l’approccio al problema dell’agente conversazionale.

Nel secondo capitolo si descrivono le materie affrontate, partendo dal tema del chatbot ed alcuni riferimenti storici, fino ad arrivare alle tecniche di elaborazione del linguaggio, riportando il flusso di sviluppo dell’analisi testuale. Vengono riportati i tipi di chatbot esaminando la struttura dei componenti fondamentali di un agente intelligente, in seguito all’applicazione diNLP.

Nel capitolo 3 viene descritto il progetto realizzato delineando il contesto aziendale e le finalità da ottemperare. Si entra quindi nel dettaglio delle tecnologie utilizzate, per comprendere i concetti principali di *Microsoft Bot Framework SDK*. Successivamente viene mostrato l’approccio con cui l’utente comunica con il bot tramite delle conversazioni d’esempio, mettendo in luce i punti chiave del dialogo. Oltre all’analisi dei requisiti si discute sulle criticità intrinseche di natura progettuale e di utilizzo.

La fase di progettazione e sviluppo viene introdotta nel capitolo 4, dove si descrivono i task eseguiti e si espongono i concetti principali sui quali viene costruito l’agente conversazionale.

Nel finale viene esaminata la fase relativa ai test effettuati, monitorando il raggiungimento dei requisiti preposti e riportando eventuali sviluppi futuri.

Indice

[Introduzione i](#_Toc2903467)

1 [Introduzione al problema 1](#_Toc2903468)

[1.1 Prenotazioni prestazioni mediche in regime SSN 2](#_Toc2903469)

[1.1.1 Prenotazione allo sportello 2](#_Toc2903470)

[1.1.2 Prenotazione tramite call center 3](#_Toc2903471)

[1.1.3 Prenotazione online 3](#_Toc2903472)

[1.2 Chatbot come soluzione? 3](#_Toc2903473)

[1.2.1 Usabilità 3](#_Toc2903474)

[1.2.2 Efficienza 4](#_Toc2903475)

[1.2.3 Sviluppo tecnologico 4](#_Toc2903476)

2 [Tecnologie Abilitanti 5](#_Toc2903477)

[2.1 Chatbot 5](#_Toc2903479)

[2.1.1 Primi esperimenti 6](#_Toc2903480)

[2.2 Natural Language Processing 6](#_Toc2903481)

[2.2.1 Processo di analisi testuale 7](#_Toc2903482)

[2.2.2 Applicare NLP per agenti conversazionali 12](#_Toc2903483)

[2.2.3 Elementi fondamentali di un agente conversazionale 14](#_Toc2903484)

3 [Progetto Aziendale 17](#_Toc2903485)

[3.1 Contesto aziendale di riferimento 17](#_Toc2903487)

[3.2 Descrizione del prototipo 18](#_Toc2903488)

[3.2.1 Processo di prenotazione di un appuntamento 18](#_Toc2903489)

[3.2.2 Integrazione con l’applicativo On.Health 20](#_Toc2903490)

[3.3 Microsoft Bot Framework SDK 20](#_Toc2903491)

[3.3.1 Bot Framework Service 20](#_Toc2903492)

[3.3.2 Luis.ai 21](#_Toc2903493)

[3.3.3 Contesto di turno ed elaborazione delle attività 21](#_Toc2903494)

[3.4 Analisi dei requisiti 23](#_Toc2903495)

[3.4.1 Requisiti obbligatori 24](#_Toc2903496)

[3.4.2 Requisiti desiderabili 27](#_Toc2903497)

[3.5 Scenari di utilizzo 28](#_Toc2903498)

[3.5.1 Scenario di prenotazione 29](#_Toc2903499)

[3.5.2 Scenario di richiesta informativa 32](#_Toc2903500)

[3.6 Criticità progettuali 33](#_Toc2903501)

4 [Progettazione e Sviluppo 35](#_Toc2903502)

[4.1 Comparazione delle soluzioni progettuali 35](#_Toc2903504)

[4.2 Implementazione applicazioni Luis.ai 37](#_Toc2903505)

[4.3 Implementazione del Dialog Manager 40](#_Toc2903506)

[4.4 Studio delle On.Health API 44](#_Toc2903507)

5 [Testing 45](#_Toc2903508)

6 [Conclusioni 49](#_Toc2903509)

Capitolo 1

# Introduzione al problema

Il problema rilevante per qualsiasi persona che gestisce un esercizio, la cui attività è regolata su appuntamenti programmati con i clienti, consiste nella gestione della propria agenda. Generalmente questo compito spetta direttamente al gestore, in riferimento ad un’attività privata di piccole dimensioni, mentre nelle grandi aziende e negli enti pubblici la gestione viene quasi sempre demandata ad un servizio di call center o presso strutture apposite. Un fattore di estrema importanza in economia aziendale è il *Customer Relationship Management*, spesso chiamato con l’acronimo CRM, che le grandi aziende utilizzano per la gestione dei rapporti con i loro clienti potenziali ed esistenti; è una strategia di marketing centrata sul cliente, che pone come obiettivo l’instaurazione di relazioni profittevoli e personalizzate con il cliente, offrendo servizi di assistenza, d’informazione e diminuendo i costi operativi [2].

Come qualsiasi altra attività, anche per la pratica medica è necessaria un’evoluzione per essere al passo con le preferenze della propria clientela. Al giorno d’oggi le strutture sanitarie hanno la necessità di modernizzare i propri processi, servizi e strumenti, in modo tale da facilitare le relazioni con il paziente permettendogli di essere più informato e facilitato durante l’accesso alle prestazioni mediche. Proprio per questo motivo, negli ultimi anni, si tratta sempre più spesso il concetto di *Patient Relationship Management* [3], ovvero una particolare declinazione del CRM, legato al concetto di fidelizzazione del cliente, che in questo caso assume il ruolo di paziente. Un approccio personalizzato e proattivo è la chiave per un modello di assistenza socio-sanitaria rivolto alla salute e al benessere del paziente per guidarlo durante tutto il continuo assistenziale.

## Prenotazioni prestazioni mediche in regime SSN

Attualmente la difficoltà maggiore di un cittadino che deve essere sottoposto a pratiche sanitarie, è rappresentata dalla possibilità d’accesso ad esse, dovuta, in particolar modo, alle metodologie di prenotazioni obsolete o poco efficienti. Ciò può comportare un’esperienza spiacevole a cui il paziente viene sottoposto, che lo vede districarsi tra chiamate a call center spesso irraggiungibili oppure sopportare interminabili code nelle strutture dedicate, con inutili perdite di tempo. In Italia per richiedere una prestazione medica del SSN è necessario rivolgersi al *Centro Unico di Prenotazione (CUP)* che accentra le prenotazioni delle prestazioni ambulatoriali presso le strutture pubbliche di una Asl[[4]](#footnote-4), città, provincia o Regione. È possibile farlo in vari modi: telefonando al numero verde per la comunicazione con un call center, recandosi di persona agli sportelli nei centri CUP o nelle farmacie associate oppure registrandosi attraverso il CUP online, un portale messo a disposizione per alcune Regioni.

L’Agenzia di tutela della Salute di Milano ha analizzato le prenotazioni dei cittadini italiani tramite il SSN ed è emerso che, nel 2018, meno del 20% delle visite mediche viene prenotato con il centralino unico di prenotazioni online, mentre la restante parte viene fissata presentandosi di persona.

### Prenotazione allo sportello

La modalità per prenotare una visita presentandosi di persona ad uno sportello apposito è la più datata e scomoda ma, a causa della mancanza di alternative valide ed efficienti, è la più utilizzata dai cittadini italiani. Ciò comporta diverse problematiche al paziente, il quale è spesso costretto a sacrificare buona parte della giornata ed eventualmente a richiedere un permesso lavorativo per mettersi in fila ed attendere il proprio turno. In aggiunta è necessario considerare che uno sportello di prenotazione è spesso disponibile solo in certi giorni della settimana e in un numero limitato di ore al giorno.

### Prenotazione tramite call center

Pur risolvendo il problema principale di una prenotazione eseguita di persona, anche chiamare un call center può presentare diversi grattacapi ad un paziente. L’esperienza insegna che spesso questi servizi non sono immediati come dovrebbero, facendo attendere il chiamante per svariati minuti ascoltando un’orecchiabile canzone, senza riuscire a comunicare con un operatore, quindi è costretto a ripetere l’operazione svariate volte; anche i call center sono raggiungibili ad intervalli temporali limitati.

### Prenotazione online

Come accennato in precedenza è possibile prenotare le proprie prestazioni mediche attraverso i portali messi a disposizione delle varie Regioni italiane, le quali però non garantiscono un’omogeneità nell’utilizzo. Per le varie Regioni che offrono tale servizio sono presenti molteplici siti web, caratterizzati da nominativi differente che presentano modalità di registrazione e uso non affini tra loro, per cui tale sistema potrebbe non essere adeguato per le tipologie di paziente.

## Chatbot come soluzione?

Dopo aver considerato le problematiche riportate in precedenza, si procede con un’analisi sull’impiego e sviluppo di un chatbot come soluzione, esponendone i vantaggi derivati.

### Usabilità

L’usabilità è sicuramente il fattore che trova maggior giovamento nell’implementazione di questo modello risolutivo. La facilità e praticità di comunicazione, garantita dall’elaborazione del linguaggio naturale, ne permette l’immediato utilizzo ad ogni categoria d’utente, poiché non è richiesta nessuna conoscenza tecnologica. Inoltre il supporto e l’integrazione con i canali di messaggistica attualmente più scaricati sugli smartphone, come *Telegram, Facebook Messenger* o *Skype,* evita al paziente di dover scaricare sul proprio device ulteriori applicazioni. Per comunicare con il chatbot sarà necessario emulare i passaggi effettuati quando si manda un messaggio ad un amico o parente.

### Efficienza

Anche l’efficienza subirebbe considerevoli miglioramenti potendo interagire con un servizio sempre funzionante, in ogni ora del giorno e per tutti i giorni dell’anno. La durata media di una comunicazione per la prenotazione di una prestazione si estende per una manciata di minuti, consentendo al paziente di eseguire l’operazione in qualsiasi momento della giornata.

### Sviluppo tecnologico

I passi avanti raggiunti nel campo dell’intelligenza artificiale nell’ultimo periodo, stanno indirizzando le grandi aziende ad investire sempre più massivamente sugli assistenti virtuali, mettendo a disposizione agli sviluppatori un numero crescente di framework per ogni necessità in relazione alla tipologia di applicativo che si vuole realizzare; nel futuro prossimo si potrà disporre di ambienti di sviluppo sempre più pratici, personalizzabili, servizi facilmente mantenibili ed aggiornati.

Capitolo 2

# Tecnologie Abilitanti



## Chatbot

Un chatbot è un software progettato per simulare una conversazione attraverso comandi vocali o messaggi testuali, il quale è in grado di comunicare con l’utente tramite il linguaggio naturale, ovvero il modo con cui le persone comunicano normalmente tra di loro. L’obiettivo di un chatbot è proprio quello di astrarre ad un livello superiore il concetto classico del software, dando all’utente la percezione di comunicare con un agente pensante, che non si limita a ricoprire un ruolo passivo nell’interazione ma anche proattivo. Poiché il termine è una composizione di due nomi, ovvero *Chat*, che in inglese significa “chiacchierare” e *Bot* che è un diminutivo di “Robot”, stiamo descrivendo un robot a tutti gli effetti, che riceve i nostri messaggi inviati su un canale di comunicazione e fornisce, a seguito di un’elaborazione dell’input, risposte pertinenti.

Il continuo sviluppo nell’ambito dell’AI e la flessibilità che caratterizza i bots, ne permettono l’utilizzo in moltissimi settori, motivo per il quale le aziende si stanno spingendo sempre più verso questo tipo di soluzione, progredendo sia nel rapporto azienda-cliente, sia per quanto riguarda l’ottimizzazione della gestione del personale interno. Van Baker, vicepresidente di *Gartner inc*., ha stimato che nel 2020 più del 50% delle aziende medio-grandi offriranno i propri prodotti e servizi attraverso l’impiego di assistenti conversazionali [4].

Un esempio tangibile è il successo di “Vera” [5], un assistente virtuale che si occupa della gestione e recruiting del personale, lanciato sul mercato nel 2016 da una startup chiamata *Statfory*, ed attualmente utilizzato da importanti aziende a livello mondiale come *Pepsi, Ikea, Auchan* e altre. Secondo gli sviluppatori l’adozione di Vera consente un risparmio di tempo e costi pari ad un terzo per la ricerca delle figure professionali, poiché analizza gli annunci di lavoro da diversi siti, si occupa di contattare i candidati ed infine sostenere un’intervista ai profili più interessanti.



Figura 2.1 L’assistente virtuale Vera durante un colloquio.

### Primi esperimenti

In realtà l’idea che supporta il modello di un chatbot non è poi così innovativa come si può pensare, infatti il termine è stato coniato da Micheal Mauldin nel 1994 per descrivere questa tipologia di programmi conversazionali, mentre i primi esperimenti risalgono addirittura al 1950, quando Alan Turing pubblica l’articolo *“Computing Machinery and Intelligence”*, in cui elabora un criterio per determinare se una macchina sia in grado di pensare, oggi definito come “*Test di Touring”*. Nel 1966 Joseph Weizenbaum realizza il primo prototipo a cui diede il nome “Eliza”, un programma progettato per illudere gli utenti convincendoli del fatto che a sostenere la conversazione ci fosse un’altra persona. L’algoritmo alla base di Eliza era in grado di rispondere alle parole e frasi in ingresso, facendo progredire nel tempo la conversazione. Tuttavia lo stesso Weizenbaum, sostenne che il suo archetipo non fosse realmente intelligente, ma si limitava a replicare con risposte vaghe e mai specifiche [6].

## Natural Language Processing

I Computer sono strumenti che funzionano perfettamente quando è necessario manipolare informazioni strutturate, come file o tabelle di un database. Riuscire ad istruire una macchina alla comprensione di informazioni non strutturate come il linguaggio umano è diventato uno dei principali oggetti di studio in ambito informatico, tanto è vero che esiste una branca dell’Intelligenza Artificiale chiamata Natural Language Processingo brevemente NLP, che ha esattamente questo scopo. Gli algoritmi fondamentali di NLP permettono di eseguire compiti altamente complessi, come scrivere un riassunto estraendo i concetti più significativi da un articolo, correzione grammaticale, traduzione del linguaggio parlato in un testo e viceversa, ricerca automatica sul Web tramite interfacciamento con i motori di ricerca, analisi dei sentimenti etc.

Quando conversiamo con un altro essere umano eseguiamo inconsciamente due processi in modo sequenziale ed alternato: quando ascoltiamo, il nostro cervello assimila le informazioni, le elabora ricavandone il significato generale e le parti principali, di conseguenza viene formulata una risposta correlata. Queste due azioni vengono eseguite dai due elementi che compongono il NLP:

* **Natural Language Understanding (NLU)**: tramite una sequenza di operazioni viene eseguita una conversione del linguaggio naturale in linguaggio artificiale, alla quale sarà collegata una rispettiva azione da eseguire.
* **Natural Language Generation (NLG)**: il processo opposto al precedente,

dove il sistema converte i dati rappresentati in forma complessa in frasi di senso compiuto che permettono una facile comprensione all’utente.

### Processo di analisi testuale

Il processo di analisi di un testo è un concetto particolarmente complicato e l’approccio che viene utilizzato può essere visto come una serie di task che hanno la funzione di scomporre il macro-problema, in problemi più semplici. Di seguito si studiano alcuni modelli risolutivi nella libreria open source[[5]](#footnote-5) *Apache OpenNLP* [9].

**Segmentazione**

Il primo passo consiste nella “rottura” del testo, per analizzare una ad una le varie frasi che compongono un paragrafo. Un modello basilare può essere rappresentato da una funzione che divide il testo in base all’inserimento della punteggiatura.

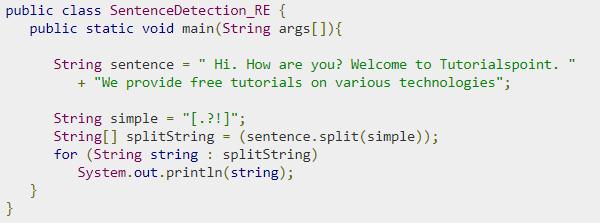


Figura 2.2 Codice della libreria OpenNLP per Sentence Detection

I moderni algoritmi di NLP utilizzano tecniche più complesse che riescono ad analizzare e comprendere anche i documenti non formattati correttamente.

**Tokenizzazione delle parole**

Dopo aver suddiviso il documento nelle frasi principali ognuna di esse viene analizzata separandone i lessemi, generalmente chiamati *Token,* in base alla categoria. Questa operazione è denominata *Tokenizzazione* ed è propedeutica ad ulteriori operazioni che vengono eseguite successivamente, come la traduzione, la determinazione delle finalità, estrazione delle entità etc.

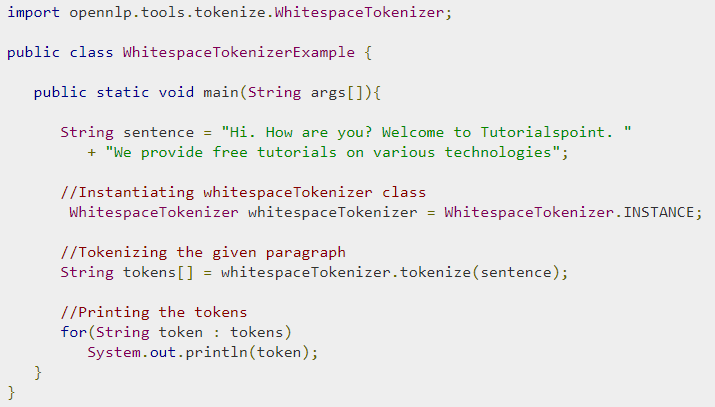


Figura 2.3 Codice della libreria OpenNLP per tokenizzazione

**Parti del discorso**

In questa fase si passano in rassegna i vari *Token*, per comprendere il ruolo di ognuno di essi, ed è evidente come il NLP sia fortemente plasmato dalla lingua oggetto di studio. Basti pensare alle differenze d’approccio necessarie tra analisi di un testo in italiano e in inglese, dove il posizionamento degli elementi grammaticali è diverso, o addirittura per lingue che adottano un altro alfabeto. Di seguito viene riportata una classificazione standard delle parti del discorso applicabile alla quasi totalità delle lingue indoeuropee.

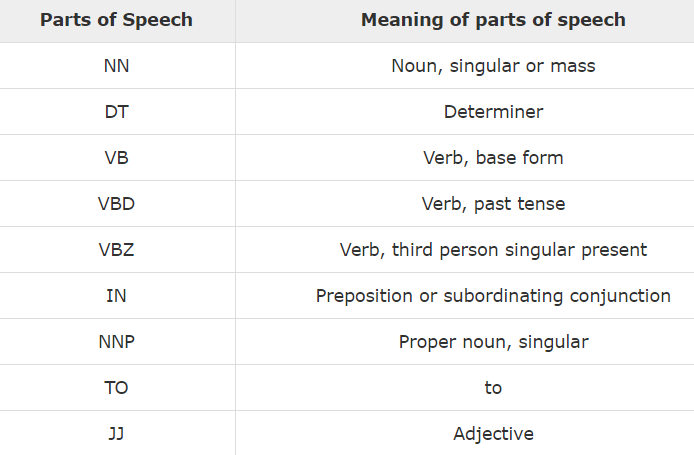


Figura 2.4 Tabella rappresentante le parti del discorso delle lingue indoeuropee



Figura 2.5 Codice della libreria OpenNLP per la catalogazione delle parti del discorso

**Lemmatizzazione**

È il processo che trasforma una parola in forma flessa nella rispettiva forma canonica, eliminando qualsiasi variazione morfologica non interpretabile dal sistema, come può essere un verbo coniugato in un tempo verbale nella sua forma infinita oppure un nome dal plurale al singolare. La lemmatizzazione viene eseguita attraverso la consultazione di una struttura dati apposita chiamata “look-up table”, dove vengono riportate le correlazioni di una parola in tutte le sue forme possibili [7].

**Identificazione delle *stop words***

Con stop words intendiamo tutte quelle parole che sono ritenute poco significative a causa della loro altra frequenza in una lingua, come articoli, congiunzioni e preposizioni.

**Traduzione delle dipendenze**

Una volta estratte le parole con significati rilevanti è necessario capire come siano relazionate le une con le altre. Una possibile soluzione consiste nel creare una struttura ad albero, in modo tale che i nodi intermedi abbiano solo un nodo genitore; la radice della struttura è rappresentata dal verbo principale dell’asserzione. La struttura può essere ulteriormente dettagliata aggiungendo la tipologia di relazione per ogni nodo e il rispettivo nodo padre. Di seguito viene riportata una rappresentazione grafica della struttura in riferimento alla frase *“London is the capital and most populous city”*:

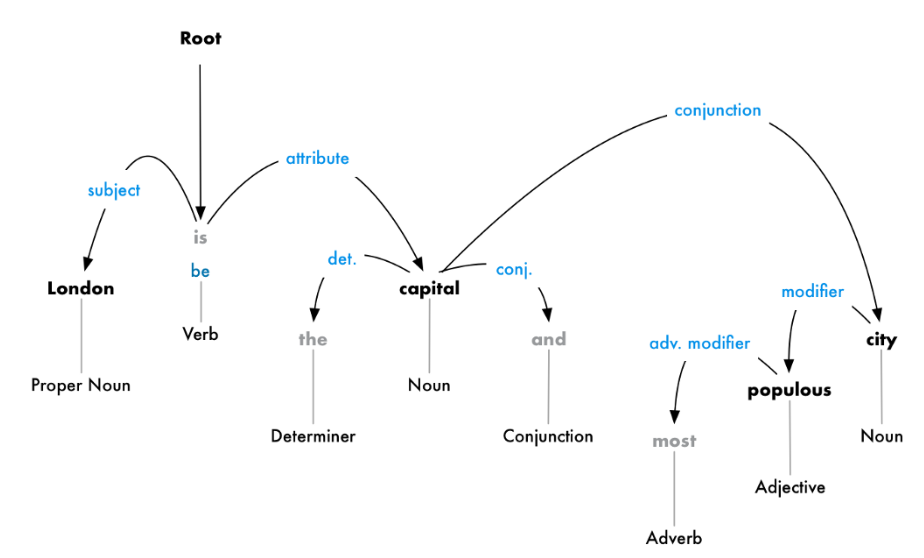


Figura 2.6 Rappresentazione grafica di una struttura che rappresenta le dipendenze funzionali di una frase

La difficoltà prevalente nella risoluzione delle dipendenze è spesso rappresentata dall’ambiguità delle varie proposizioni, le quali possono avere una struttura sintattica diversa e contenere al proprio interno parole con significati multipli a seconda del contesto di riferimento; tutto ciò può causare problematiche con conseguente esito errato da parte del modello. È quindi evidente come l’apprendimento automatico, comunemente noto come *Machine Learning*,sia estremamente importante per il miglioramento progressivo del sistema, che sarà in grado di imparare in modo “adattivo” dagli input forniti.

**Estrazione delle entità**

Il processo che si occupa di trovare nomi di cose e persone, luoghi, prodotti ed altri elementi, viene chiamato *NER* dal nome completo *Named Entity Recognition*. Per adempiere a questo requisito, solitamente vengono usati dei modelli predefiniti che si occupano di estrarre le rispettive entità in una stringa di testo. Le entità vengono etichettate in base ad un elevato numero di tipi di oggetto, ognuno dei quali contiene una lista di record [8]. Di seguito vengono alcuni esempi:

* Nomi di persone
* Nomi di società
* Aree geografiche
* Nomi di prodotti
* Tipologie di valuta
* Numerazioni
* Unità di misura

Nell’esempio successivo viene riportato il codice sorgente di una funzione che si occupa di ricavare i nomi propri di persona e mostrarli in output con le rispettive posizioni.



Figura 2.7 Codice della libreria OpenNLP che estrae i nomi propri di persona mostrandoli a video con le rispettive posizioni

### Applicare NLP per agenti conversazionali

Le tipologie di chatbot esistenti si distinguono in base alla complessità dei compiti che devono assolvere, di conseguenza la loro struttura può essere più o meno complessa. Nativamente un chatbot è un sistema abbastanza banale, non smart che riceve degli input testuali inseriti dall’utente senza la possibilità di attribuire ad essi alcun significato. L’elaborazione di NLP unita alla duttilità garantita dall’infrastruttura di un chatbot, consentono l’esistenza di agenti conversazionali utili in tantissime situazioni e contesti. A seconda della profondità conversazionale di cui sono muniti, possono essere così classificati:

**Scripted Bot**

Sono i chatbot in cui l’aspetto conversazionale è nullo pensati per funzionare tramite l’invio di comandi specifici ai quali vengono associate delle azioni. Non prevedono l’utilizzo di contesti di conversazione, offrendo un modo di dialogare detto *“one-step”*, dove ogni messaggio è completamente indipendente dagli altri. L’utente deve essere a conoscenza delle parole esatte che permettono di abilitare i vari “trigger” predisposti dallo sviluppatore del bot, azzerando le possibilità di funzionamento nel caso in cui l’utente commetta degli errori sintattici. Come mostrato in *Figura 2.8,* quando viene progettato questo tipo di soluzione, si fa largo uso di pulsanti per indirizzare l’user correttamente.

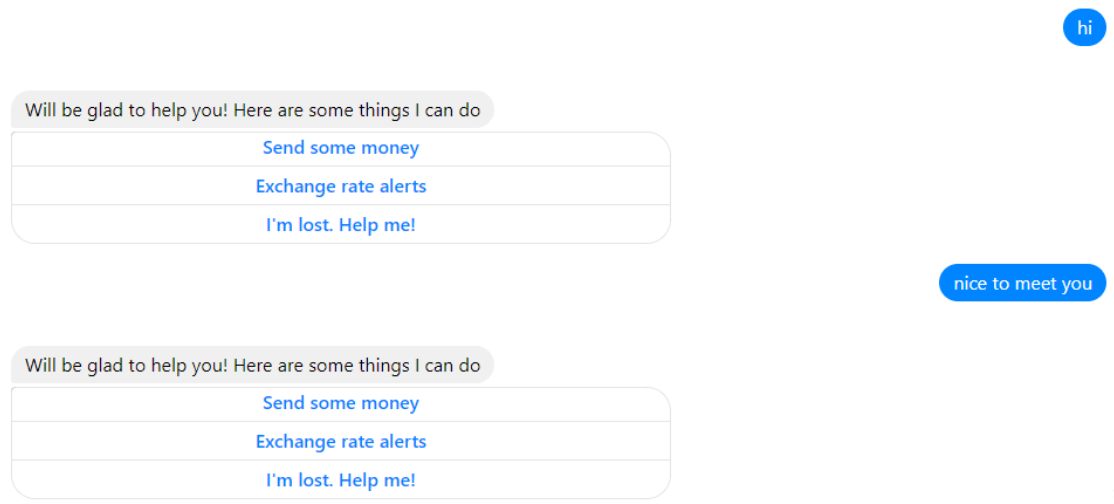


Figura 2.8 Il bot ignora gli input testuali inseriti dall’utente riproponendo lo stesso messaggio con 3 opzioni

Vengono spesso utilizzati per rispondere alle FAQ o attuare compiti semplici.

**Agenti conversazionali**

Se il dominio del problema richiede un approccio più performante dove è necessario mantenere un contesto di conversazione e si intende migliorare la User Experience[[6]](#footnote-6) si può applicare l’AI, ottenendo un agente conversazionale. Oltre a tutti i vantaggi di NLP descritti nel paragrafo precedente, sarà possibile mantenere una correlazione tra messaggi, evitando che il sistema “resetti la memoria” ad ogni nuovo invio. Grazie a questo è possibile conservare informazioni per ogni utente; per esempio, per un sistema che ha l’obiettivo di prendere ordini, sarà possibile, mantenere le informazioni relative all’indirizzo e i metodi di pagamento preferiti, facilitando e velocizzando il processo, oppure mandare messaggi di notifica mostrando prodotti filtrati in base all’interesse ad hoc. Questi agenti migliorano in percentuale la capacità di identificazione della volontà dell’utente, genericamente chiamata “hit ratio”, perfezionandosi con l’esperienza e con i dati che immagazzinano nel tempo.

### Elementi fondamentali di un agente conversazionale

Lo sviluppo di un agente conversazionale richiede al progettista di porre particolare attenzione agli elementi portanti del modello, i quali sono “Intents”, “Entities”, “Actions” e “Context”. La loro comprensione è imprescindibile per comprendere il capitolo 4, dove viene trattata la progettazione del prototipo e lo studio dei frameworks:

* **Intents**: rappresenta un’attività o una finalità che l’utente desidera eseguire e che deve essere captata dal software. Durante la progettazione si devono predisporre un set di intent, ognuno dei quali verrà individuato dall’*engine* a fronte di una richiesta dell’utente, cercando di capire le modalità di espressione che verranno poste dagli utilizzatori futuri. Inoltre è importante mantenere un numero bilanciato di intent, evitando da una parte di crearne un numero esagerato che potrebbe portare il sistema a non individuare l’intent giusto, e dall’altra predisponendone un numero limitato con conseguenti finalità troppo poco specifiche. Un intent è formato da due sezioni principali:
* **User says**:sequenza di modi con cui una frase può essere scritta o enunciata. Ovviamente non è richiesto fornire come esempio ogni possibile forma sintattica, tuttavia più il set è fornito e tanto più sarà alto l’*hit ratio*. Per esempio una richiesta metereologica si può scrivere *“Mostrami il tempo a Cesena”*, oppure *“Come sono le previsioni per Cesena?”*.

Quando viene elaborata un’espressione, oltre all’intent correlato viene indicato un punteggio, che indica il grado di attendibilità della stima. Se il punteggio è medio-basso è possibile programmare la logica del sistema per chiedere in input parametri aggiuntivi, per una maggiore correttezza di funzionamento.

* **Response**: è possibile definire le risposte correlate ad ogni intent, che vengono inviate ad ogni richiesta della finalità.
* **Actions**: sono azioni che l’agente deve eseguire quando viene invocato l’intent ad esse collegato. Per esempio, un agente che controlla i dispositivi di una *smart home* si interfaccerà con gli oggetti distribuiti per la casa, oppure una semplice chiamata *http* se è necessario la connessione con una API[[7]](#footnote-7) esterna.
* **Entities**: sono strumenti molto importanti per l’estrazione delle parole chiave all’interno di una proposizione e vengono richiesti per eseguire una *action*. I framework solitamente dispongono di set precostruiti di entità, le quali possono essere viste come classi di oggetti generici (luoghi, nomi di persone, orari, colori…), oppure sarà lo sviluppatore a creare entità personalizzate. Dall’espressione *“Ordina 3 biglietti per Londra”* viene estratto il *3* come istanza della classe *Numeri*, e *Londra* come istanza della classe *Luoghi*.
* **Contesto**: ad ogni richiesta effettuata l’utente si trova in un contesto di riferimento, il quale può essere influenzato dalla posizione geografica o dalle sue preferenze che aiutano l’agente a rispondere correttamente a richieste vaghe.
* **Ruoli**: sono sottotipi contestuali denominati di un’entità, e descrivono la collocazione dell’entità nella frase. Nell’espressione *“prenotare un volo da Milano a Madrid”*, sia Milano che Madrid sono città, tuttavia è abbastanza evidente come assumano un significato differente, poiché la prima assume il ruolo di *origine,* mentre la seconda di *destinazione.*

Capitolo 3

# Progetto Aziendale

In questo capitolo si descrivono funzionalità e caratteristiche del progetto realizzato, ponendo particolare attenzione all’analisi dei requisiti e agli scenari di utilizzo. Il prototipo oggetto di tesi è stato progettato e realizzato completamente dal sottoscritto presso l’azienda *Onit Group s.r.l.*, in un lasso di tempo lavorativo di circa 4 mesi.



## Contesto aziendale di riferimento

*Onit Group s.r.l.* nasce nel 2001 a Cesena e svolge la propria attività nel settore dell’ICT e della consulenza, mirata al management e all’organizzazione aziendale. L’azienda, che nel 2019 conta circa 120 collaboratori, lavora in diversi segmenti di mercato sul territorio nazionale, realizzando soluzioni che trovano applicazione in ambito industriale, di Business Intelligence[[8]](#footnote-8), mobile e sanitario.

Negli ultimi anni l’azienda sta ponendo particolare attenzione nella fornitura di nuovi prodotti per i propri clienti, con l’intenzione di espandere il mercato tramite l’erogazione dei propri servizi con nuove tecnologie. Per questo motivo è nata la possibilità di lavorare alla progettazione di *OnAssistant,* un prototipo realizzato per comprendere benefici e criticità inerenti all’applicazione di un agente conversazionale in campo sanitario. Il nome del progetto deriva dall’unione del prefisso *“On”* che viene assegnato a tutti i software proprietari ed *“Assistant”*, che in inglese significa assistente.



Figura 3.1 Logo di Onit Group s.r.l.

## Descrizione del prototipo

Il progetto nasce dal proposito di offrire una valida alternativa ad un paziente che necessita l’accesso ad una o più prestazioni mediche erogate dal *Servizio Sanitario Regionale*. Il sistema è strutturato per riuscire ad emulare il processo di prenotazione, che generalmente viene eseguito ad uno sportello del *CUP,* mettendo a disposizione un punto d’accesso disponibile al paziente, eludendo vincoli temporali e fisici. Aprendo l’applicazione di messaggistica istantanea preferita, esso sarà in grado di avviare il processo in una manciata di secondi e fissare comodamente l’appuntamento.

Il sistema è anche in grado di rispondere alle domande più frequenti richieste dagli utenti inerenti ad argomenti differenti, che possono essere:

* assistenza sull’utilizzo della chat e sulle funzionalità predisposte.
* consigli ed informazioni sulle prestazioni mediche prenotabili.

### Processo di prenotazione di un appuntamento

Per poter avviare il flusso della prenotazione occorre conoscere il codice fiscale del paziente e il codice della ricetta elettronica.

**Ricetta Elettronica**

La ricetta elettronica, o ricetta dematerializzata, rappresenta l’evoluzione della classica ricetta cartacea ed è un documento elettronico che viene rilasciato dal medico. Questo documento è identificato univocamente dal codice *NRE*, e può essere riferito alla prescrizione di farmaci o alle richieste di visite specialistiche.

Al codice della ricetta elettronica possono essere associate una o più visite specialistiche prescritte dal medico, le quali verranno notificate all’utente al momento dell’inserimento nella chat e per ognuna di esse si aprirà un processo distinto di prenotazione. A questo punto l’utente può inserire i giorni settimanali e la fascia giornaliera preferita, mattina o pomeriggio, come filtri di ricerca. Per ogni visita verranno caricati gli appuntamenti disponibili, filtrati in base alle richieste dell’utente, visualizzati in ordine temporale, riportandone la data, l’orario, la struttura sanitaria e il costo. L’utente può navigare tra gli appuntamenti disponibili messaggiando in modo naturale con l’assistente, scorrere le disponibilità successive e precedenti ed eventualmente fissare l’appuntamento. Per la prima utilizzazione il paziente dovrà registrarsi sull’applicativo *On.Health*.

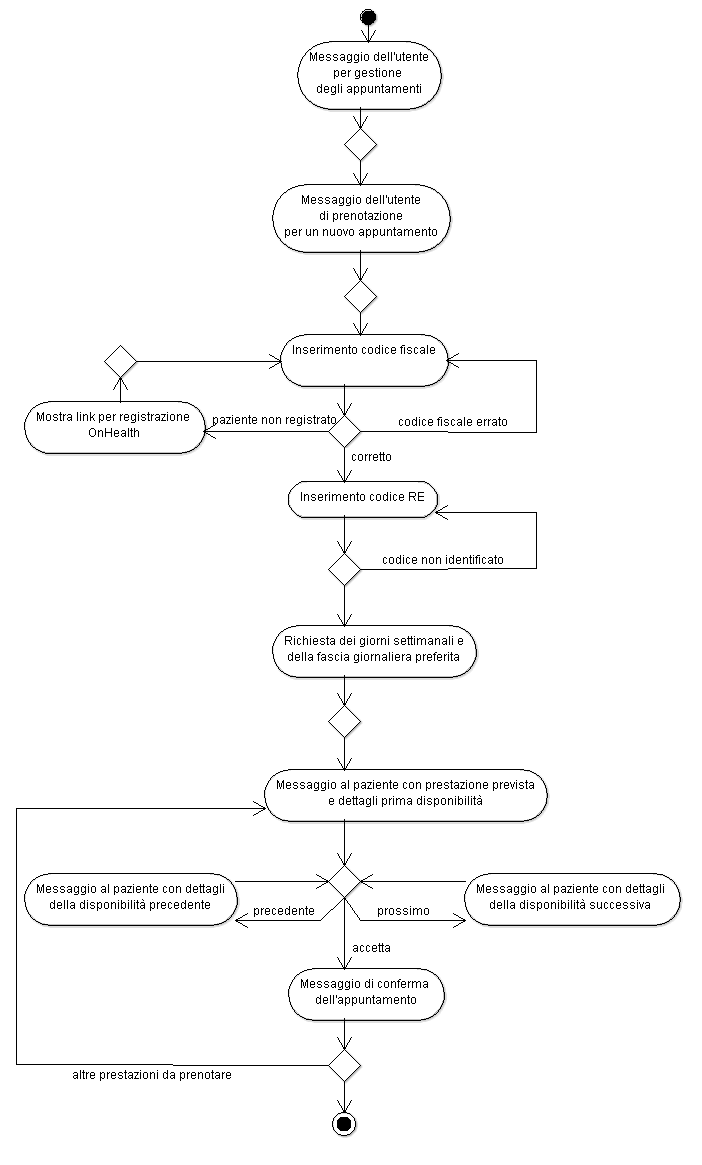


Figura 3.2 Diagramma delle attività per il processo di prenotazione di una visita specialistica di SSN

### Integrazione con l’applicativo On.Health

Per l’effettivo funzionamento, il software si integra con un altro applicativo sviluppato dall’azienda, On.Health, una piattaforma web-based che permette la completa gestione dei processi clinici delle organizzazioni sanitarie. Nello specifico il bot richiama un set di REST API per tutti i processi attinenti alle informazioni del paziente, ricerca degli appuntamenti e recupero delle informazioni sulle visite specialistiche. Nel capitolo 4, che concerne la progettazione e lo sviluppo, si analizzano le funzionalità preesistenti utilizzate e quelle che sono state implementate.

## Microsoft Bot Framework SDK

È la piattaforma di sviluppo fornita da *Microsoft* che è stata adottata per la progettazione, la quale fornisce allo sviluppatore un ambiente per la creazione, testing e distribuzione di bot intelligenti, fornendo il supporto sia per *Node,js* che per *.NET[[9]](#footnote-9)* [9]. Per la realizzazione è stato utilizzato C# e l’IDE Visual Studio 2017.

I componenti principali di *Bot Framework* sono il *Bot Framework Service*,che si occupa di fornire l’interfacciamento al bot con canali di comunicazione, e quello inerente alle meccaniche di elaborazione del linguaggio naturale,chiamato *Language Understanding Intelligent Service* (*Luis.ai*).

### Bot Framework Service

Questo componente invia delle *Activity* al bot ad ogni messaggio inviato dall’utente, le quali contengono informazioni che dipendono dal tipo di messaggio, dall’utente e dal canale utilizzato. Le attività più importanti messe a disposizione dal framework sono di due tipologie:

* *Conversation Update*: attività di aggiornamento della conversazione. Generalmente viene inviata al bot quando un utente si aggiunge alla conversazione.

* *Message*: attività inerente ad un messaggio effettivamente spedito dall’utente. Per esempio, potrebbe essere un semplice testo, un inserimento vocale dell’utente, contenere immagini, etc.

### Luis.ai

È il servizio di NLU sviluppato da *Microsoft,* facente parte della suite di prodotti *Azure*, utilizzato per permettere all’agente di elaborare il linguaggio naturale dell’user sotto forma testuale, comprendendo il significato delle frasi ed abilitandolo all’estrazione delle entità all’interno delle stesse [10]. Per utilizzare le API Luis.ai viene fornito un portale apposito, dove è possibile creare applicazioni composte da una serie di Intents ed Entities; per ogni applicazione vengono generati un endpoint ed una chiave (*Authoring Key*) che verranno inserite nei files di configurazione del progetto per permetterne il raggiungimento. In questo modo sarà possibile inoltrare l’input testuale ricevuto dal Bot Framework Service*,* per mano dell’user, verso i servizi collegati, i quali forniranno i dati elaborati strutturati in formato JSON[[10]](#footnote-10) e quindi interpretabili dall’applicativo sviluppato.

Per quanto riguarda il costo sono disponibili diverse soluzioni economiche, come quella gratuita che consente la risoluzione di massimo 10,000 input testuali ogni mese oppure a pagamento se si necessita di più risorse [11].

### Contesto di turno ed elaborazione delle attività

Le attività scambiate dal bot e il servizio Bot Framework arrivano tramite richieste http POST, le quali vengono confermate con relative risposte con codice di stato http 200.

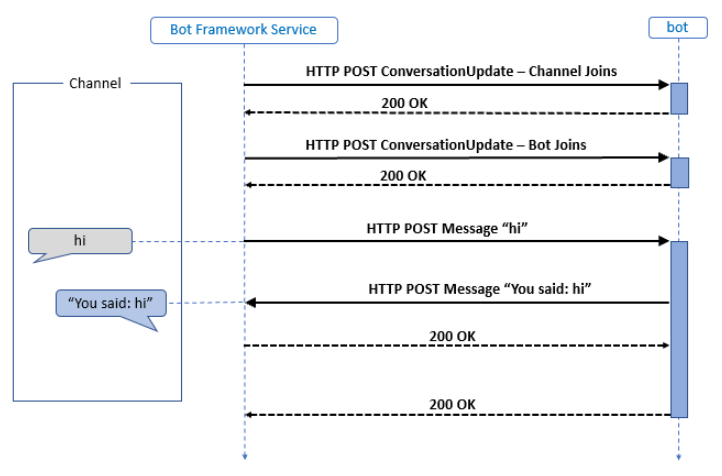


Figura 3.3 Modello di conversazione tra il canale, servizio Bot Framework e il Bot

L’interazione tra il bot e l’utente è strutturata come una conversazione tra due persone, dove spesso si parla uno alla volta, a turno. Il bot reagisce agli input dell’utente rispondendo in modo pertinente, ricreando così proprio il concetto di *turno*.

In Bot Framework SDK, un turno è composto da una attività che l’utente invia al bot e dalla relativa risposta. L’oggetto *contesto di turno* fornisce informazioni sulle attività che compongono la conversazione, indicando il mittente, destinatario, il canale utilizzato e altri elementi utili. In *Figura 3.4* viene rappresentato nel dettaglio il processo di elaborazione di una chiamata http POST e come gli elementi del framework processino l’attività.

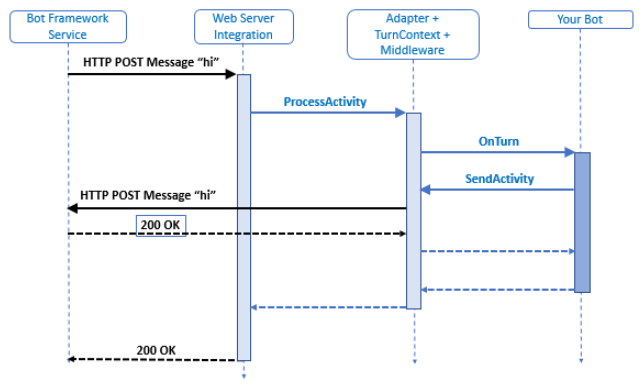


Figura 3.4 Elaborazione di una chiamata http POST dal canale di messaggistica in concetto di turno

Inizialmente il Web Server, che nel prototipo realizzato in C# è un progetto *ASP.NET*[[11]](#footnote-11), elabora il *body* della chiamata POST, che contiene le informazioni del messaggio, deserializza il contenuto informativo ricevuto in formato JSON e richiama l’*Adapter* tramite il processo dell’attività (*ProcessActivity*). Quest’ultimo riceve l’attività e crea un contesto di turno, il quale permette l’invio di messaggi sul canale di comunicazione attraverso metodi asincroni (es. *SendActivity*).

## Analisi dei requisiti

Di seguito viene descritto lo studio dei requisiti dell’applicativo, suddividendoli tra quelli “obbligatori” e quelli “desiderabili”, prendendo come dominio del problema tutti gli aspetti discussi nel Capitolo 1. Vengono analizzati gli elementi critici della soluzione scelta e riportati gli scenari di utilizzo.

**Tracciamento dei requisiti**

Il tracciamento dei requisiti è stato discusso congiuntamente con il responsabile dell’Area Sanità di *Onit* e il correlatore, interno all’azienda, che ha seguito lo sviluppo. In particolare vengono individuati tutti i casi d’uso possibili, gli attori, le modalità di sviluppo e l’integrazione con il software proprietario On.Health*.*

I requisiti, sono suddivisi in due tipologie principali (A e B); ognuno è caratterizzato da un codice identificativo composto dal tipo di gruppo di appartenenza, dalla funzionalità (*Obbligatoria* o *Desiderabile*) e da un indice incrementale.

**Tipologie Requisiti**

* *Tipologia A*: vengono classificati di tipo A tutti i casi d’uso inerenti alla gestione degli appuntamenti del paziente.
* *Tipologia B*: vengono classificati di tipo B tutti i casi d’uso inerenti al servizio delle domande frequenti (FAQ).

Per esempio con il codice *AO1*, si indica il requisito di Tipo A, Obbligatorio con indice 1.

### Requisiti obbligatori

|  |  |  |
| --- | --- | --- |
| **Tipologia** | **Descrizione** | **Funzionalità** |
| AO1 | Messaggio dall’utente per la gestione dei propri appuntamenti. | Obbligatorio |
| AO2 | Messaggio dall’utente per la richiesta di un nuovo appuntamento. | Obbligatorio |
| AO3 | Controllo, nella logica del bot, per la correttezza sintattica del codice fiscale. | Obbligatorio |
| AO4 | Messaggio dal bot contenente l’indirizzo di registrazione per l’applicativo On.Health per un nuovo utente. | Obbligatorio |
| AO5 | Controllo, nella logica del bot, per la correttezza sintattica del codice della ricetta elettronica. | Obbligatorio |
| AO6 | Messaggio dall’utente per inserimento specifico dei giorni preferiti per gli appuntamenti. (es. “*Lunedì e Sabato*”). | Obbligatorio |
| AO7 | Messaggio dall’utente per inserimento di un intervallo dei giorni preferiti per gli appuntamenti (es. “*dal Lunedì al Sabato*”). | Obbligatorio |
| AO8 | Messaggio dall’utente per inserimento delle fasce giornaliere preferite per gli appuntamenti (es. “mattina o pomeriggio” | Obbligatorio |
| AO9 | Messaggio dal bot con le informazioni principali di una disponibilità (data, ora, luogo, costo). | Obbligatorio |
| AO10 | Messaggio dall’utente per visualizzazione della disponibilità successiva. | Obbligatorio |
| AO11 | Messaggio dall’utente per visualizzazione della disponibilità precedente. | Obbligatorio |
| BO1 | Richiesta di informazioni dall’utente sul SSN. | Obbligatorio |
| BO2 | Richiesta di informazioni dall’utente sulle funzioni disponibili del bot. | Obbligatorio |
| BO3 | Richiesta di informazioni dall’utente sulle visite specialistiche specifiche. | Obbligatorio |

*Tabella 3.1 Requisiti obbligatori*

**Attori**

Gli attori che partecipano direttamente alla comunicazione sul canale sono l’utente, che utilizza l’applicazione di messaggistica dal proprio smartphone o PC, e il bot che si interfaccia tramite il Bot Framework Service. Il secondo funge anche da tramite verso “attori esterni” che interagiscono in modo transitivo alla comunicazione, come le applicazioni di Luis.aiper l’integrazione dell’AI e le API predisposte da On.Health*.*

Di seguito vengono riportati i casi d’uso delle funzionalità appartenenti alle tipologie A e B.

**Diagrammi dei casi d’uso tipologia A**

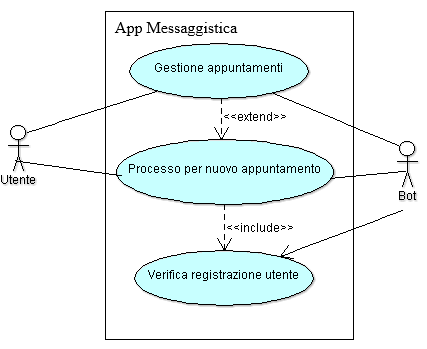


Figura 3.5 Caso d’uso relativo al processo di prenotazione del SSN

La funzione *“Processo per nuovo appuntamento”* comprende l’insieme delle singole operazioni che compongo l’iter per una nuova prenotazione del SSN.

**Diagrammi dei casi d’uso tipologia B**

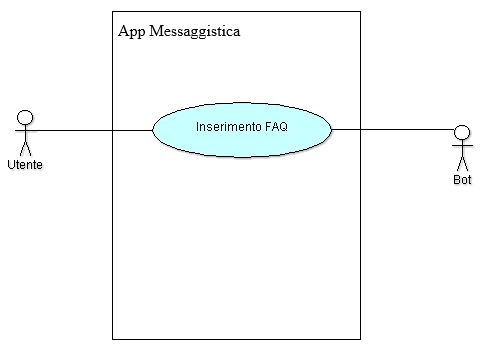


Figura 3.6 Caso d’uso relativo alle domande frequenti

Per le operazioni di tipologia B non è richiesta la registrazione, poiché limitate a semplici elementi informativi messi a disposizione dal bot per l’utente generico.

### Requisiti desiderabili

Anche i requisiti desiderabili sono stati redatti durante la fase di analisi e fanno parte di tutte le funzionalità che verranno sviluppate in futuro; seppur meno importanti dei requisiti obbligatori, è necessario prenderli in considerazione durante la fase di progettazione e sviluppo, per facilitare la loro integrazione durante le implementazioni future.

|  |  |  |
| --- | --- | --- |
| **Tipologia** | **Descrizione** | **Funzionalità** |
| AD1 | Messaggio dell’utente per la richiesta di visualizzazione degli appuntamenti prenotati. | Desiderabile |
| AD2 | Messaggio dell’utente per la richiesta di cancellazione di un appuntamento prenotato. | Desiderabile |
| AD3 | Messaggio dell’utente per la richiesta di modifica di un appuntamento prenotato. | Desiderabile |

*Tabella 3.2 Requisiti desiderabili*

## Scenari di utilizzo

In questo paragrafo vengono descritti gli scenari d’interazione tra gli utilizzatori e l’agente conversazionale, prestando particolare attenzione ai punti chiave delle conversazioni. Nel primo esempio viene riportato il flusso del dialogo finalizzato alla prenotazione di un appuntamento, mentre nel secondo è possibile esaminare come il sistema assolve alla casistica in cui l’utente richiede informazioni su una prestazione medica.

### Scenario di prenotazione

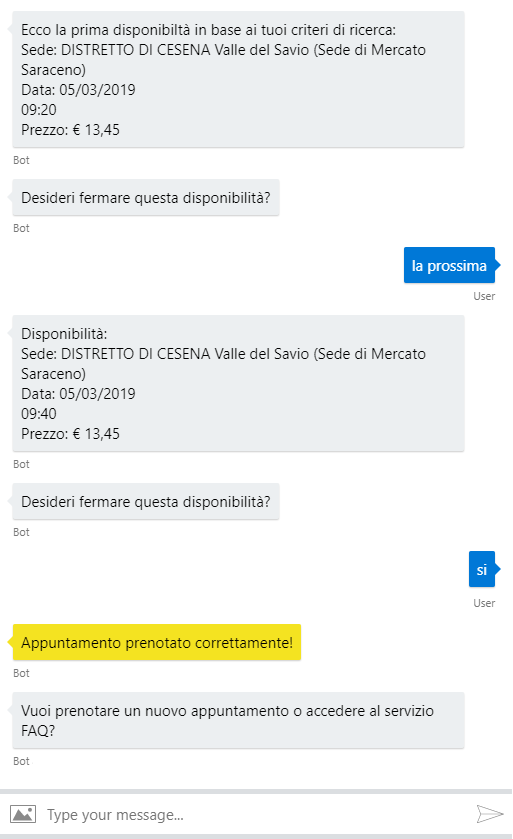
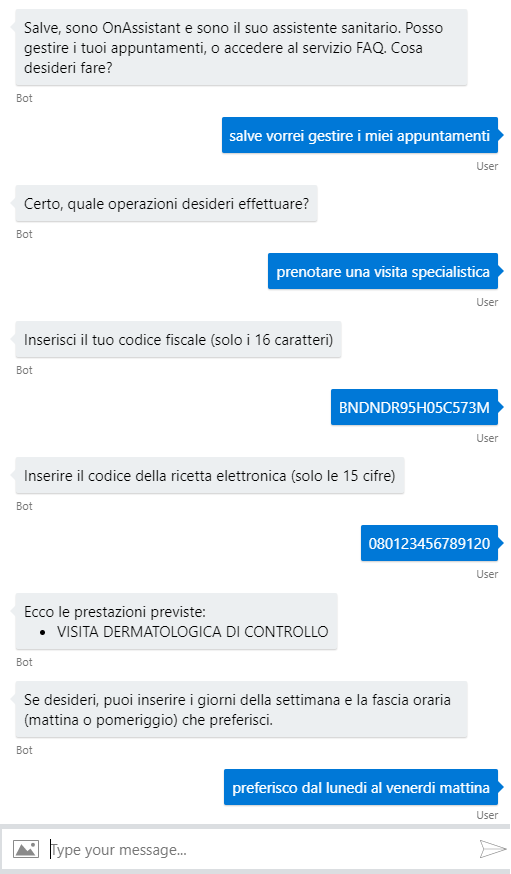


Figura 3.7 Prima parte di conversazione per la prenotazione di un appuntamento

Figura 3.8 Seconda parte di conversazione per la prenotazione di un appuntamento

1. Bot: appena l’utente si unisce alla conversazione il bot riceve un messaggio di tipo *Conversation Update*, saluta l’utente dichiarando le funzionalità per cui è predisposto.
2. User: richiede l’accesso al servizio per la gestione degli appuntamenti.
3. Bot: richiede il tipo di operazione da effettuare.
4. User: specifica di voler prenotare un nuovo appuntamento.
5. Bot: richiede il codice fiscale (solamente i 16 caratteri).
6. User: inserisce il codice fiscale.
7. Bot: controlla il codice fiscale, possibili risultati:
   1. errore sintattico (es. numero di cifre): richiede nuovamente l’inserimento del codice fiscale.
   2. formato corretto, ma non trova corrispondenza con un utente registrato su On.Health. Manda una scheda con un pulsante per reindirizzare l’utente sulla pagina Web di On.Health per la registrazione; annulla il contesto di prenotazione e ritorna alla root di conversazione.

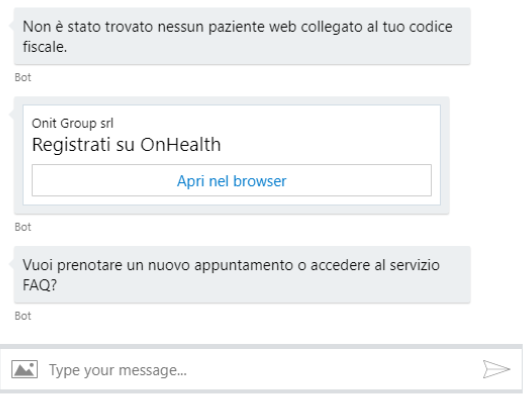


Figura 3.9 Il bot non trova corrispondenza tra il codice fiscale un utente registrato

* 1. formato corretto e collegato ad un utente registrato su On.Health, quindi è possibile procedere con la prenotazione. Chiede il codice della ricetta elettronica (solamente le 15 cifre).

1. User: inserisce la ricetta elettronica.
2. Bot: controlla il codice della ricetta elettronica, possibili risultati:
   1. errore sintattico: rimanda la richiesta per l’inserimento del codice della ricetta.
   2. formato corretto, ma nessuna prestazione medica collegata: annulla il contesto di prenotazione e ritorna alla root di conversazione.
   3. formato corretto e presenza di prestazioni mediche collegate: manda un messaggio con i nomi delle prestazioni e chiede all’utente se intenzionato a inserire filtri per la ricerca.
3. User: può comportarsi in 2 modi:
   1. non inserisce filtri di ricerca rispondendo con una negazione (es: *no grazie*). In questo caso saranno presi come validi tutti i giorni settimanali ed entrambe le fasce giornaliere, mattina e pomeriggio.
   2. inserisce giorni e fasce orarie preferite per filtrare la ricerca degli appuntamenti.
4. Bot: ricerca degli appuntamenti, possibili risultati:
   1. non trova un appuntamento, quindi annulla il contesto di prenotazione e ritorna alla root di conversazione.
   2. trova almeno un appuntamento, quindi invia le informazioni base della disponibilità più vicina in ordine temporale.
5. User: quando riceve un appuntamento prenotabile può:
   1. richiedere l’appuntamento successivo.
   2. richiedere l’appuntamento precedente.
   3. accettare l’appuntamento e ricevere la notifica dal bot di prenotazione eseguita. Se è presente un’altra prestazione da prenotare associata alla ricetta si ritorna al punto 11, altrimenti il processo viene concluso.

### Scenario di richiesta informativa

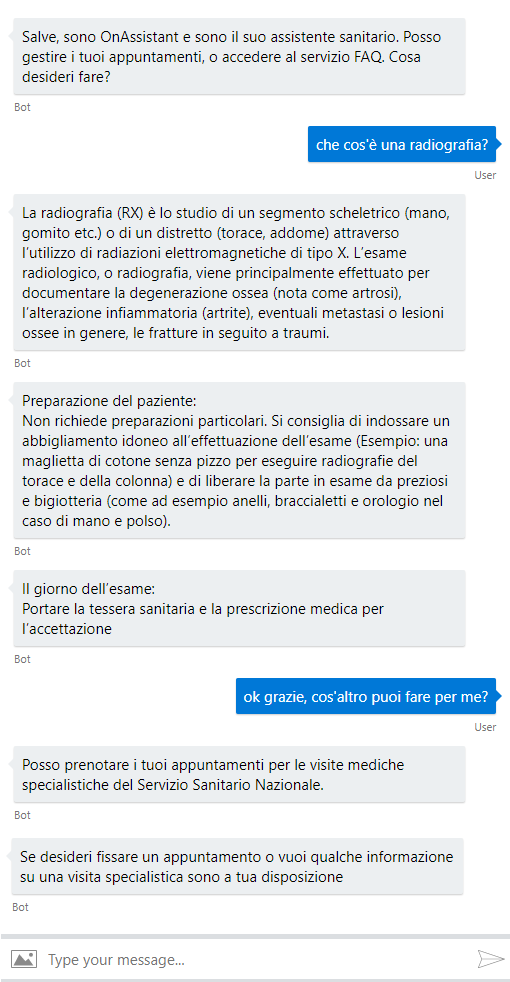


Figura 3.10 Conversazione dove l’utente richiede informazioni per una radiografia e successivamente sulle funzionalità del bot

1. Bot: appena l’utente si unisce alla conversazione il bot riceve un messaggio di tipo *Conversation Update*, saluta l’utente dichiarando le funzionalità per cui è predisposto.
2. User: effettua una domanda su una visita specialistica. Nell’esempio vengono richieste le informazioni su una radiografia.
3. Bot: risponde fornendo informazioni relative alla visita in oggetto.
4. User: richiede le funzioni che il bot può eseguire.
5. Bot: risponde dando la possibilità all’utente di chiedere ulteriori informazioni sulle visite mediche, o accedere al servizio di gestione appuntamenti.

## Criticità progettuali

Durante il processo di analisi dei requisiti sono state tracciate anche tutte le criticità della soluzione scelta come modello risolutivo, per avere una previsione delle possibili difficoltà legate alla fase di sviluppo e quelle inerenti all’eventuale rilascio in produzione.

Per quanto riguarda lo sviluppo, le problematiche più consistenti sono rappresentate dall’applicazione di un framework poco conosciuto dal sottoscritto e dalla maggior parte dei membri del team. Per questo motivo è stato necessario studiare approfonditamente la documentazione per comprendere gli aspetti tecnici e concettuali richiesti per realizzare un assistente conversazionale valido ed efficiente.

Un altro problema è legato all’esperienza di utilizzo dell’utente con un agente intelligente, che lo porta, a seguito di molteplici risposte corrette, a rilassarsi fornendo input sempre meno specifici e poco dettagliati. Questo aspetto è un elemento che può ostacolare il bot durante il suo operato.

Infine è presente un limite strettamente correlato alla possibilità d’accesso di alcune prestazioni mediche non prenotabili telematicamente; alcune di esse presentano particolari difficoltà d’accesso che richiedono l’intervento di un operatore.

Capitolo 4

# Progettazione e Sviluppo

Questa sezione si colloca all’interno delle scelte progettuali attuate e dei procedimenti operativi che hanno portato alla realizzazione del prototipo oggetto d’osservazione. Vengono indicate le vie implementative prese in considerazione, quelle percorse e gli strumenti impiegati per la costruzione dei componenti preminenti.



## Comparazione delle soluzioni progettuali

In primo luogo si è posta l’attenzione sulle tecnologie ed i framework utilizzabili, soffermando l’attenzione sul problema introdotto nel capitolo 1. Gli elementi più importanti che hanno richiesto una maggiore accuratezza in fase di progettazione sono stati, da una parte la scelta del servizio per la comprensione del linguaggio naturale, dall’altra il metodo per realizzare l’agente che assume il ruolo di *Dialog Manager (DM)*[[12]](#footnote-12).

La selezione della piattaforma da adoperare è stata guidata considerando i seguenti fattori:

* **Lingua italiana**: necessario il supporto della lingua italiana, sia per quanto riguarda il riconoscimento delle frasi, sia per il processo di *tokenizzazione* necessario per l’estrazione delle entità.
* **Costo**: ricercare una soluzione che offra, almeno per la fase di sviluppo e testing, piani tariffari gratuiti.
* **Documentazione**: una documentazione esaustiva e ben strutturata è fondamentale per le fasi di sviluppo e di manutenibilità del software.
* **Infrastruttura**: tecnologie e strumenti utilizzabili per implementare la logica del bot.

Posti i seguenti vincoli è stato necessario studiare la struttura e le potenzialità dei servizi fruibili dagli sviluppatori, che vengono velocemente descritti di seguito. Tra le varie soluzioni offerte sono state considerate:

**Google API.ai (Dialogflow)**: una piattaforma sviluppata da *Google* per la creazione di esperienze di conversazione articolate con gli utenti, che si appoggia sui concetti base di *Agents*, che possono essere visti come moduli di NLU, progettati per l’estrazione di Intentsed Entities*.* Per quanto riguarda la messa in produzione *Google* mette a disposizione i propri servizi web, con la possibilità di esecuzione sulla piattaforma *Google Cloud*. Il servizio si integra con diversi canali di messaggistica e supporta la lingua italiana. È possibile utilizzare una versione standard gratuita con il numero delle richieste limitato, oppure soluzioni a pagamento; documentazione approfondita e chiara [12].

**IBM Watson Assistant**: ambiente di proprietà di *IBM* che offre un set di strumenti per lo sviluppo di agenti intelligenti tramite l’applicazione dei più innovativi algoritmi di AI e NLP. Il numero di elaborazioni degli input testuali, per la soluzione gratuita, è limitato a 10,000. Lingua italiana supportata ma documentazione dispersiva e non immediata [13].

**Wit.ai**: sviluppata da *Facebook*, offre agli sviluppatori un SDK[[13]](#footnote-13) abbastanza efficace e pratico per strutturare chatbot, facilitando la realizzazione di sistemi che si interfacciano con dispositivi mobile e IoT[[14]](#footnote-14). È presente la compatibilità con la lingua italiana, tuttavia è possibile utilizzarlo solamente con il canale *Facebook Messenger*, precludendo l’utilizzo degli altri canali [14].

**Apache OpenNLP**: una libreria open source scritta in Java che si limita all’elaborazione del testo in formato naturale senza fornire alcun strumento per la gestione del flusso del dialogo. Italiano non supportato [9].

La scelta finale è ricaduta su **Microsoft Bot Framework** per svariati vantaggi, da quelli tecnici descritti nella sezione 3.3, a quelli pratici legati al contesto aziendale. Il framework di riferimento degli applicativi aziendali è ASP.NET, quindi questa scelta faciliterebbe l’integrazione degli altri membri del team, che si ritroverebbero a lavorare su un progetto del quale conoscono il linguaggio (C#), l’IDE (Visual Studio 2017) e i concetti generali di *project* e *solution*.

## Implementazione applicazioni Luis.ai

A questo punto è stato fondamentale passare in rassegna l’analisi dei requisiti del sistema, per mappare quelli obbligatori con un Intent specifico; le finalità individuate sono state a loro volta correlate a delle azioni gestite dalla logica del bot. Per ogni finalità individuata sono state riportate le entità utili da estrapolare.

Ogni applicazione di Luis.ai è un modulo, contenente un certo numero di Intents ed Entities, che viene utilizzato dal bot quando è necessario l’elaborazione di un input testuale. Nello specifico sono state create due applicazioni:

**OnAssistant-prenotazione**

Il modulo più articolato dei due, contenente tutte le finalità ed entità utilizzate per risolvere i messaggi inviati dall’user durante la prenotazione.

|  |  |
| --- | --- |
| **Nome Intent** | **Descrizione** |
| None | Intent di base inserito di default in ogni modulo Luis; viene selezionato a fronte di un input mancato. |
| Gestione\_appuntamenti | Intent per la gestione degli appuntamenti. |
| Inserimento\_filtri\_giorno\_ora | Intent per l’inserimento dei filtri temporali (giorni e fasce orarie). |
| Confermare | Intent di conferma. |
| Negare | Intent di negazione. |
| Indietro | Intent per tornare indietro. |
| Avanti | Intent per proseguire. |
| Stop | Intent per fermare il processo. |

*Tabella 4.1 Intents rilevabili durante il processo di prenotazione.*

|  |  |
| --- | --- |
| **Nome Entity** | **Descrizione** |
| Operazione | Entità per ricavare il tipo di operazione da eseguire per un appuntamento. |
| Giorno | Entità per ricavare un giorno settimanale. |
| DaGiorno | Entità per ricavare il giorno settimanale di partenza se viene inserito un intervallo. |
| AGiorno | Entità per ricavare il giorno settimanale di arrivo se viene inserito un intervallo. |
| FasciaOra | Entità per ricavare la fascia oraria giornaliera. |

*Tabella 4.2 Entities rilevabili durante il processo di prenotazione.*

**OnAssistant-FAQ**

Questo modulo è composto dalle finalità ed entità individuate per le possibili domande generali proposte dall’utente.

|  |  |
| --- | --- |
| **Nome Intent** | **Descrizione** |
| None | Intent di base inserito di default in ogni modulo Luis; viene selezionato a fronte di un input mancato. |
| Funzioni\_assistente | Intent per la richiesta delle funzionalità programmate dell’assistente. |
| Info\_SSN | Intent per la richiesta di informazioni del SSN. |
| Info\_Vis\_Specialistica | Intent per la richiesta di informazioni su una visita specialistica. |

*Tabella 4.3 Intents rilevabili durante il processo di FAQ*

|  |  |
| --- | --- |
| **Nome Entity** | **Descrizione** |
| Nome\_Vis\_Specialistica | Entità per ricavare il tipo di visita medica. (ecografia, risonanza magnetica, TAC, MOC etc.) |

*Tabella 4.4 Entities rilevabili durante il processo di FAQ*

Per la gestione del flusso lavorativo sono stati utilizzati degli strumenti messi a disposizione dal Bot Framework, ovvero *CLI tools* [15]. È una collezione di strumenti a riga di comando che supporta completamente il flusso lavorativo, partendo dalla possibilità di creazione dei vari componenti fino alle operazioni di deploy degli stessi.

Con lo strumento *npm-luis* si è reso possibile generare le applicazioni sul portale Luis.aipartendo da un semplice file di testo (con estensione *.lu*) formattato come in *Figura 4.1 e 4.2:*

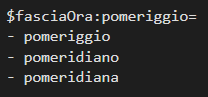
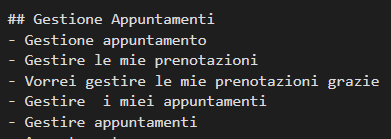
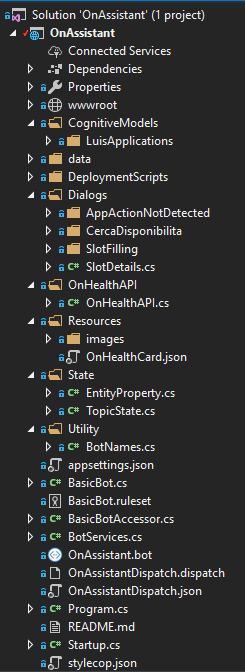


Figura 4.2 Sintassi per dichiarazione di una finalità

Figura 4.1 Sintassi per dichiarazione di una entità

## Implementazione del Dialog Manager

L’agente è il componente del sistema conversazionale responsabile per la gestione dello stato e del flusso del dialogo, il quale è stato realizzato in C#. Gli elementi concettualmente più interessanti dell’agente sono la gestione dello stack dei dialoghi (*Figura 4.5*) ed il controllo dello stato della conversazione. In *Figura 4.3* è possibile vedere l’organizzazione del progetto:



*Figura 4.3 Organizzazione del progetto C#*

**Dialoghi**

Sono i componenti forniti dall’SDK per modellare il flusso della conversazione e consistono in classi C# che estendono l’interfaccia *Dialog*. È possibile istanziare la classe *DialogContext* per gestire i dialoghi standard forniti dal framework (*Figura 4.4*), oppure implementare dialoghi personalizzati, detti “dialoghi componenti”, eseguendo l’override dei seguenti metodi:

* *BeginDialogAsync*: aggiunge un dialogo in cima allo stack.
* *ContinueDialogAsync*:per i dialoghi che necessitano di più turni per essere completati; per esempio quando l’utente scorre gli appuntamenti prenotabili, come visto nella sezione 3.5.1 inerente allo scenario di prenotazione.
* *EndDialogAsync*: termina il dialogo eliminandolo dallo stack, passando il risultato opzionale al dialogo padre.
* *ResumeDialogAsync*: chiamato quando il dialogo figlio viene terminato.

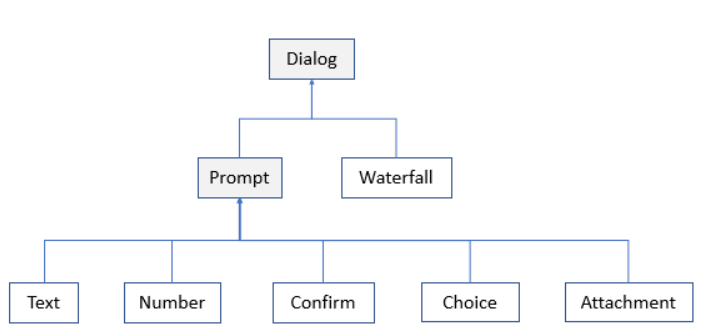
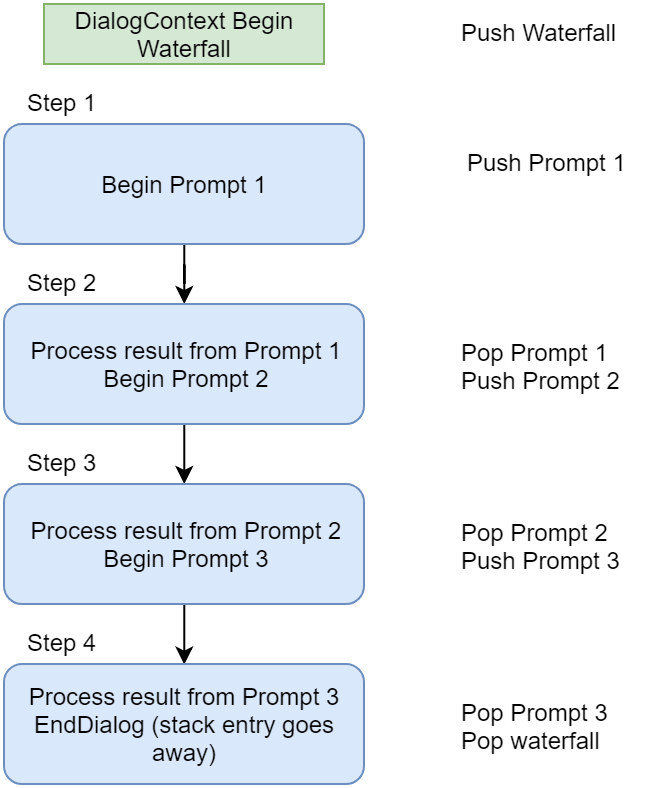


Figura 4.5 Esempio di un dialogo di tipo “Waterfall”

Figura 4.4 Dialoghi base del Bot Framework

**Stato**

La gestione dello stato è estremamente variabile a seconda del tipo di assistente che si desidera realizzare. Per esempio, ci sono sistemi con “memoria a lungo termine” che possono mantenere memorizzate le informazioni dell’utente per svariati giorni, oppure, come nel caso del progetto realizzato, sistemi che mantengono le informazioni importanti solamente durante la conversazione e le azzerano quando termina. Per salvare i dati e recuperarli tra i vari turni, viene utilizzata la classe *BotAccessor*, la cui istanza viene passata all’agente ad ogni turno.

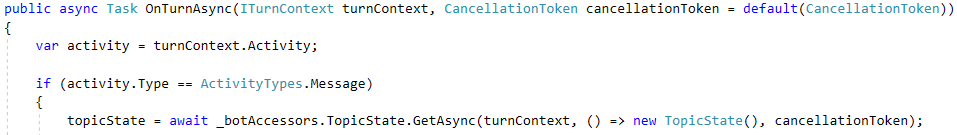


Figura 4.6 Per ogni turno, se l’attività è un messaggio dell’utente, viene recuperata una proprietà dell’Accessor che mantiene info. utili per lo stato del dialogo



Figura 4.7 Metodo per salvare il valore di una proprietà dell’Accessor.

Poiché il sistema fa largo uso di chiamate http esterne, sia alle API di Luis.ai che a quelle di On.Health, è stato applicato un approccio asincrono nell’implementazione dei metodi interessati.

**Attachment**

Come è possibile notare nella sezione 3.5.1, in particolare nella *Figura 3.9*, il framework permette di inviare degli “attachment” nei messaggi; ad esempio delle schede grafiche di diverso tipo, le quali possono contenere file audio, animazioni, pulsanti etc. Quella rappresentata è una scheda adattiva utilizzata per il reindirizzamento su una pagina Web esterna, che viene supportata dalla totalità delle applicazioni di messaggistica. Il design delle schede è definibile in formato JSON (*Figura 4.8*).



Figura 4.8 File JSON che definisce il design di una scheda di tipo adattiva.

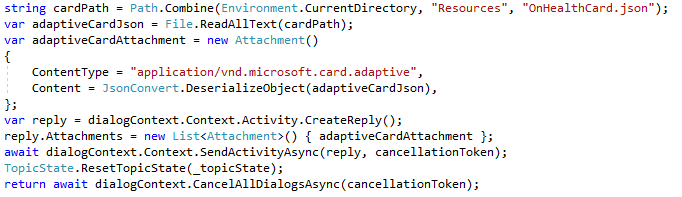


Figura 4.9 Creazione dell’attachment in C#

## Studio delle On.Health API

Terminate le parti costituenti l’agente conversazionale, è stato necessario capire le procedure da attuare per l’interfacciamento con On.Health. Per certificare l’applicativo OnAssistantè stato generato un GUID[[15]](#footnote-15) che viene passato ad ogni chiamata http. In particolare sono stati individuati sia i metodi preesistenti forniti dalle API, sia quelli da implementare per l’adempimento delle funzioni richieste.

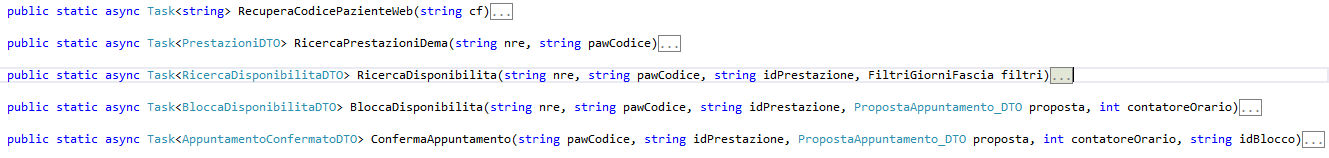


Figura 4.10 Metodi nei quali vengono eseguite le chiamate http ad On.Health

Capitolo 5

# Testing

La fase di test è stata effettuata in due modalità: in parallelo allo sviluppo per testare costantemente i singoli componenti e in una fase finale per appurare il funzionamento globale del sistema e verificare l’integrazione tra gli elementi progettuali sviluppati.

Ogniqualvolta veniva creata o modificata un’applicazione Luis.ai, si procedeva testandola sul portale, dove è predisposta una Web chat che permette di interrogare direttamente l’applicazione per ottenere un feedback preventivo (*Figura 5.1*).

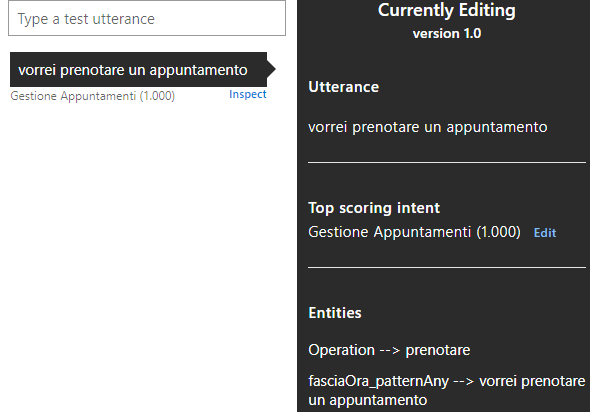


Figura 5.1 Web chat predisposta sul portale Luis.ai

Inoltre è presente una dashboard contenente alcuni grafici che aiutano lo sviluppatore a monitorare le risorse utilizzate, offrendo una panoramica sugli Intents individuati. Il grafico chiamato “endpoint hits” (*Figura 5.2*) riporta il numero di richieste giornaliere alle API, mentre il diagramma a torta “intent breakdown” raffigura le percentuali in base alle finalità più richieste dagli utenti. (*Figura 5.3*).

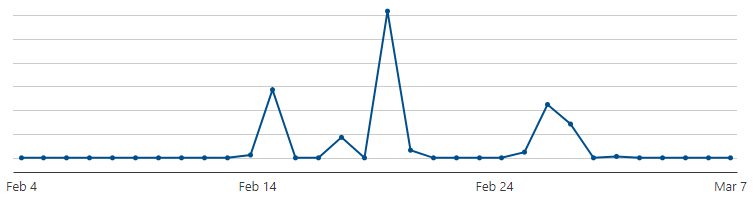


Figura 5.2 Diagramma endpoint hits

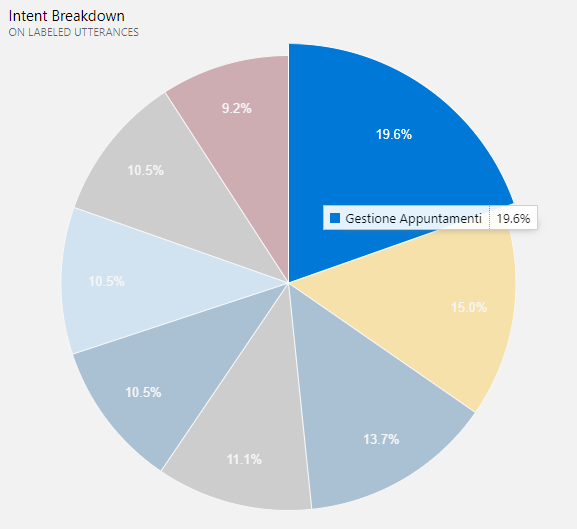


Figura 5.3 Diagramma intent breakdown

L’integrazione con On.Health è stata inizialmente testata utilizzando *Postman*, uno strumento utile per esaminare le rispettive API in fase di sviluppo, evitando di dover ricorrere continuamente al debug del codice sorgente. Questo tool offre un’interfaccia facile da usare che permette di inviare chiamate http ad un server, locale o remoto, e visualizzare tutte le risposte ottenute in modo dettagliato.

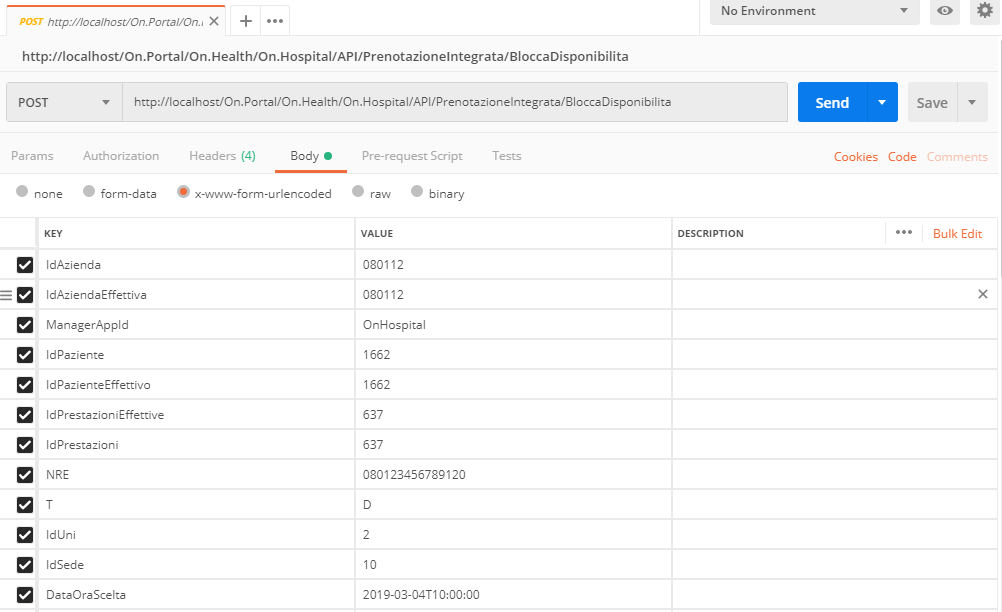


Figura 5.4 Chiamata http Post locale per la prenotazione di un appuntamento in Postman

I requisiti obbligatori del sistema sono stati collaudati su un ambiente locale utilizzando lo strumento *Bot Framework Emulator*, un’applicazione desktop che consente agli sviluppatori di eseguire il test e il debug dei bot. Usando l’emulatore, è stato possibile comunicare via chat con il bot e controllare i messaggi inviati e ricevuti. Nella Solution C# è presente un file di configurazione con estensione *.bot* che l’emulatore utilizza per contattare l’applicativo in esecuzione locale.

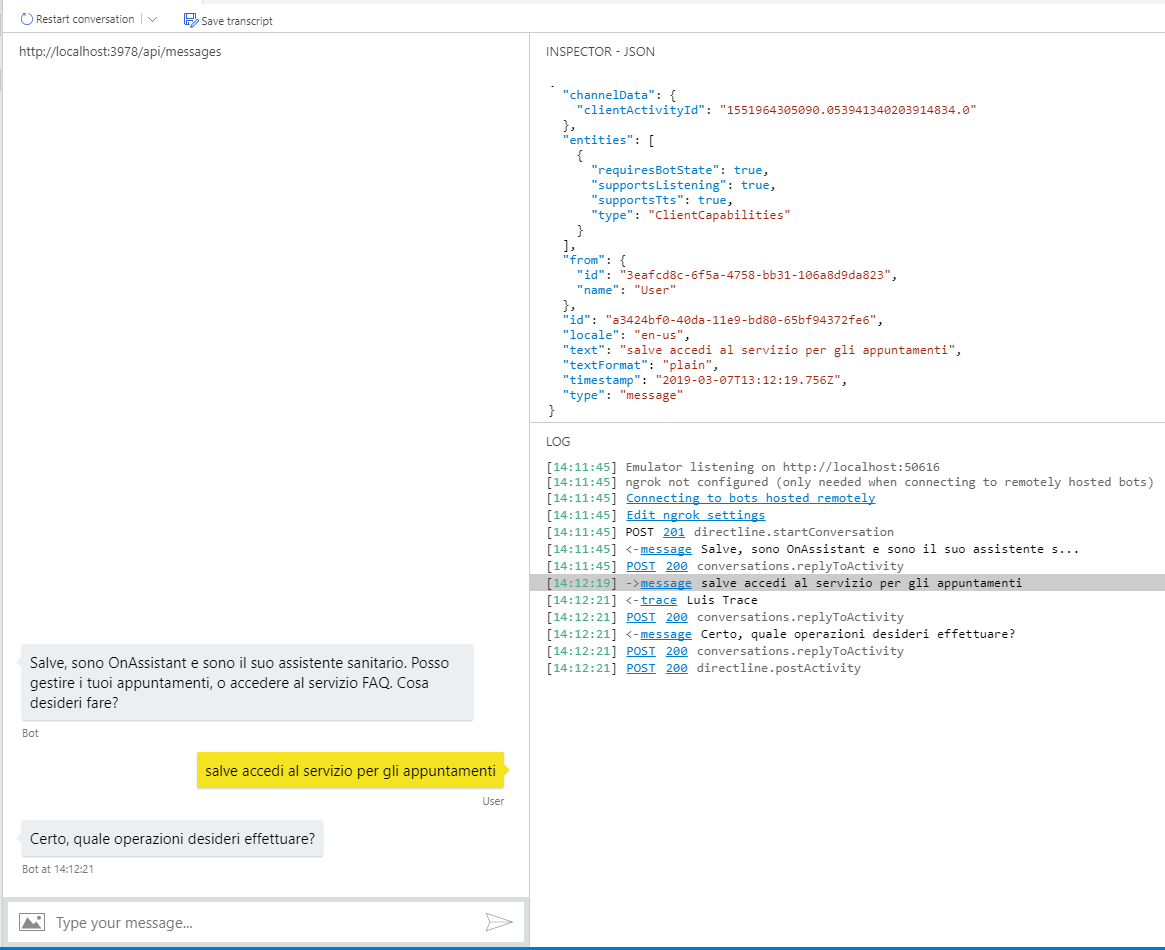


Figura 5.5 Test effettuato con lo strumento Bot Framework Emulator

A lavoro terminato sono stati effettuati test facendo comunicare il sistema con alcuni colleghi esterni al progetto, per capire se un utilizzatore fosse in grado sin da subito di prenotare un appuntamento e utilizzare il servizio delle FAQ in modo autonomo.

Capitolo 6

# Conclusioni

Raggiungimento degli obiettivi

Come anticipato nell’introduzione e nell’analisi dei requisiti, il prototipo è stato realizzato con l’intento di fornire uno strumento al paziente in grado di facilitare la prenotazione di prestazioni mediche del Servizio Sanitario Nazionale e rispondere ad eventuali domande sottoposte. L’obiettivo prioritario è consistito nello studiare l’effettivo funzionamento di un assistente conversazionale applicabile in ambito sanitario, per comprendere se fosse possibile ovviare alle problematiche legate ai metodi di prenotazione obsoleti fruibili attualmente. Il chatbot permette di instaurare un servizio sempre disponibile e pratico, tuttavia è stato necessario verificare se il sistema fosse in grado di sostenere una conversazione articolata con l’utente e raggiungere gli obiettivi preposti.

I test effettuati hanno confermato il soddisfacimento dei requisiti individuati, in particolare quelli di tipo obbligatorio descritti nella sezione 3.4.1. Le prove eseguite con l’emulatore, che nei test rappresenta l’applicazione di messaggistica, dimostrano come sia effettivamente possibile dialogare naturalmente con l’assistente e completare il processo di prenotazione velocemente. I “tester”, rappresentati da altri membri del team esterni al progetto che si sono interfacciati per la prima volta con la chat, hanno impiegato un tempo massimo di due minuti per fissare un appuntamento.

Il prototipo realizzato è ampiamente estendibile per quanto riguarda le possibili funzionalità. Con il team sono state tracciate le prossime attività che prevedono, in primis l’implementazione delle funzionalità marcate come desiderabili durante la progettazione, e successivamente ulteriori sviluppi più complessi. Per esempio la creazione di un sistema di notifiche per ricordare al cliente un appuntamento prenotato oppure la possibilità di gestire i propri referti medici.

Ringraziamenti

*Ringrazio innanzitutto il Professore Alessandro Ricci per avermi aiutato con grande disponibilità.*

*Dedico questo lavoro alla mia famiglia che mi ha dato la possibilità di intraprendere questo percorso e mi ha sempre sostenuto.*

Bibliografia

[1] *“Chatbot in Sanità, così l’intelligenza artificiale migliora il rapporto coi medici”* <https://www.agendadigitale.eu/sanita/chatbot-in-sanita-cosi-lintelligenza-artificiale-migliora-il-rapporto-coi-medici/>

[2] Wikipedia contributors. *“Customer-relationship management”* <https://en.wikipedia.org/wiki/Customer-relationship_management>

[3] *“Il patient relationship management: il crm in sanità a supporto del paziente”*

<https://www.reply.com/appheal-reply/it/content/il-patient-relationship-management-il-crm-in-sanita-a-supporto-del-paziente>

[4] Laurence Goasduff. *“Chatbots Will Appeal to Modern Workers”*

<https://www.gartner.com/smarterwithgartner/chatbots-will-appeal-to-modern-workers/>

[5] “*SAP and Russian Startup Robot Vera Boost HR Recruiting Efficiency”*

<https://news.sap.com/2018/10/sap-joins-forces-robot-vera/>

[6] Wikipedia contributors. *“Chat bot”*

<https://it.wikipedia.org/wiki/Chat_bot>

[7] Joël Plisson, Nada Lavrac e Dunja Mladenić. *“A Rule based Approach to Word Lemmatization”*

<https://www.researchgate.net/profile/Nada_Lavrac/publication/228525639_A_rule_based_approach_to_word_lemmatization/links/546e16dd0cf2bc99c2151cb0/A-rule-based-approach-to-word-lemmatization.pdf>

[8] Giuseppe Rizzo e Raphaël Troncy. *"NERD: Framework for Unifying Named Entity Recognition and Disambiguation Extraction Tools."* In: *EACL '12 Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics,* Apr. 2012, pp 73-76

<http://www.aclweb.org/anthology/E12-2015?CFID=54723043&CFTOKEN=f246c2e40373d127-1A2A827F-9015-4485-72C13B3469B0A922>

[9] *Apache OpenNLP Documentation*

[*https://opennlp.apache.org/*](https://opennlp.apache.org/)

[10] *Microsoft Bot Framework Documentation v4*

<https://docs.microsoft.com/it-it/azure/bot-service/?view=azure-bot-service-4.0>

[11] *Microsoft Luis.ai Documentation*

<https://docs.microsoft.com/it-it/azure/cognitive-services/luis/>

[12] *Dialogflow Documentation*

<https://dialogflow.com/docs>

[13] *IBM Documentation*

<https://www.ibm.com/watson/services/natural-language-understanding/>

[14] *Wit.ai Documentation*

<https://wit.ai/docs>

[15] *CLI tools Documentation*

<https://github.com/Microsoft/botbuilder-tools>

1. “L'intelligenza artificiale (AI) è una disciplina appartenente all'informatica che studia i fondamenti teorici, le metodologie e le tecniche che consentono la progettazione di sistemi hardware e sistemi di programmi software capaci di fornire all’elaboratore elettronico prestazioni che, a un osservatore comune, sembrerebbero essere di pertinenza esclusiva dell’intelligenza umana.” -Wikipedia [↑](#footnote-ref-1)
2. “Servizio Sanitario Nazionale, nell’ordinamento giuridico italiano, identifica il complesso delle funzioni, delle attività e dei servizi assistenziali gestiti ed erogati dallo Stato italiano.” -Wikipedia [↑](#footnote-ref-2)
3. “Le *Frequently Asked Questions*, sono letteralmente domande poste frequentemente, cioè una serie di risposte stilate direttamente dall'autore alle domande che gli vengono poste più frequentemente dagli utilizzatori di un certo servizio.” -Wikipedia [↑](#footnote-ref-3)
4. “Azienda sanitaria locale, è un ente pubblico della pubblica amministrazione italiana, deputato all’erogazione di servizi sanitari.” -Wikipedia [↑](#footnote-ref-4)
5. “In informatica il termine inglese open source viene utilizzato per riferirsi ad un software di cui i detentori dei diritti rendono pubblico il codice sorgente, favorendone il libero studio e permettendo a programmatori indipendenti di apportarvi modifiche ed estensioni.” -Wikipedia [↑](#footnote-ref-5)
6. “Con User Experience o UX s'intende ciò che una persona prova quando utilizza un prodotto, un sistema o un servizio. L'esperienza d'uso concerne gli aspetti esperienziali, affettivi, l'attribuzione di senso e di valore collegati al possesso di un prodotto e all'interazione con esso.“ [↑](#footnote-ref-6)
7. “Con Application Programming Interface o API, si indica un insieme di procedure atte all'espletamento di un dato compito; spesso tale termine designa le librerie software di un linguaggio di programmazione.” -Wikipedia [↑](#footnote-ref-7)
8. Per Business Intelligence (BI) si intendono tutti I processi e strumenti utilizzati da una azienda per raccogliere dati di diversa natura, analizzarli e trarne decisioni strategiche. [↑](#footnote-ref-8)
9. “Microsoft .NET è una piattaforma di sviluppo che mette a disposizione varie funzionalità come il supporto per più linguaggi di programmazione, modelli di programmazione asincroni e simultanei, interoperabilità, consentendo l'esecuzione su più piattaforme ed in presenza di scenari applicativi variegati.” -Wikipedia [↑](#footnote-ref-9)
10. “JSON, acronimo di JavaScript Object Notation, è un formato adatto all'interscambio di dati fra applicazioni client/server basato sul linguaggio JavaScript.” -Wikipedia [↑](#footnote-ref-10)
11. “ASP.NET è un insieme di tecnologie di sviluppo di software per il web, commercializzate da Microsoft. Utilizzando queste tecnologie gli sviluppatori possono realizzare applicazioni Web e servizi Web.” [↑](#footnote-ref-11)
12. “Il Dialog Manager (DM) è un componente di un sistema conversazionale responsabile della gestione dello stato e del flusso del dialogo.” [↑](#footnote-ref-12)
13. “Un software development kit (SDK) in informatica, indica genericamente un insieme di strumenti per lo sviluppo e la documentazione di software.” -Wikipedia [↑](#footnote-ref-13)
14. “In telecomunicazioni Internet delle cose (IoT, acronimo dell'inglese Internet of things) è un neologismo riferito all'estensione di Internet al mondo degli oggetti e dei luoghi concreti.” -Wikipedia [↑](#footnote-ref-14)
15. “Il GUID (Globally Unique Identifier, identificatore unico globale) è un numero pseudo-casuale usato nella programmazione software, per poter distinguere vari oggetti.” -Wikipedia [↑](#footnote-ref-15)