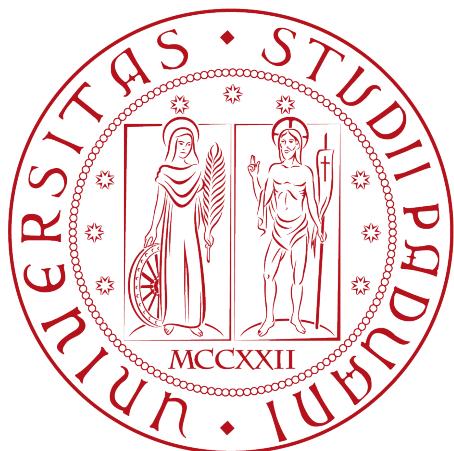


Università degli Studi di Padova
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"
CORSO DI LAUREA IN INFORMATICA



**Chatbot di Facebook Messenger per la
risposta a domande frequenti**

Tesi di laurea triennale

Relatore

Prof. Paolo Baldan

Laureando

Mauro Carlin

ANNO ACCADEMICO 2016-2017

Mauro Carlin: *Chatbot di Facebook Messenger per la risposta a domande frequenti*,
Tesi di laurea triennale, © Settembre 2017.

I'm personally convinced that computer science has a lot in common with physics. Both are about how the world works at a rather fundamental level. The difference, of course, is that while in physics you're supposed to figure out how the world is made up, in computer science you create the world. Within the confines of the computer, you're the creator. You get to ultimately control everything that happens. If you're good enough, you can be God. On a small scale.

— Linus Torvalds

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage dal laureando Mauro Carlin presso l'azienda i-contact s.r.l. di Belluno (BL). Lo stage è stato svolto alla conclusione del percorso di studi della Laurea Triennale ed è durato in totale 300 ore. Vengono riportate tutte le principali attività intraprese durante questo periodo, le difficoltà incontrate e una panoramica sul prodotto finale ottenuto.

Gli obiettivi da raggiungere erano molteplici. In primo luogo l'azienda ha richiesto un'analisi dei principali strumenti per il **NLP** presenti sul mercato, in modo da valutarne pregi e difetti. Questo strumento viene utilizzato per trasformare le domande di un utente in dati processabili.

Il passo successivo è stato studiare ed integrare questo sistema in due **chatbots** di Facebook Messenger creati e gestiti dall'azienda stessa, per dare la possibilità all'utente di interagire con essi anche tramite domande di senso compiuto, e non solo attraverso le possibilità offerte dalla piattaforma di Facebook.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Paolo Baldan, relatore della mia tesi, per l'aiuto e il sostegno fornитomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori Rita e Roberto per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ringrazio tutti gli amici conosciuti durante questo percorso, in particolare Luca, Pier Paolo, Nicola, Mattia, Marco, Simeone e Tommaso, per aver reso questi tre anni indimenticabili.

Infine ringrazio tutti coloro che mi hanno aiutato, anche con un piccolo gesto, a raggiungere questo meraviglioso traguardo.

Padova, Settembre 2017

Mauro Carlin

Indice

| | | |
|----------|--|-----------|
| 1 | Introduzione | 1 |
| 1.1 | Il progetto di stage | 1 |
| 1.2 | Profilo aziendale | 1 |
| 1.3 | Tecnologie utilizzate | 2 |
| 1.3.1 | Applicazioni iOS e Android native | 2 |
| 1.3.2 | Applicazioni web con Java | 2 |
| 1.3.3 | Frontend | 2 |
| 1.4 | Processi aziendali | 3 |
| 1.4.1 | Metodologia | 3 |
| 1.4.2 | Strumenti a supporto dei processi | 3 |
| 1.5 | Clienti | 5 |
| 1.6 | Struttura del documento | 6 |
| 1.6.1 | Convenzioni tipografiche | 6 |
| 2 | Progetto di stage | 7 |
| 2.1 | Chatbot | 7 |
| 2.2 | Descrizione del progetto | 8 |
| 2.3 | Modalità di svolgimento | 9 |
| 2.4 | Obiettivi dello stage | 9 |
| 2.4.1 | Obiettivi obbligatori | 10 |
| 2.4.2 | Obiettivi desiderabili | 10 |
| 2.5 | Pianificazione del lavoro | 10 |
| 2.6 | Principali problematiche | 11 |
| 2.6.1 | Difficoltà di ambientazione | 11 |
| 2.6.2 | Difficoltà lavorative e soluzioni adottate | 11 |
| 2.7 | Strumenti utilizzati | 12 |
| 2.7.1 | NetBeans | 12 |
| 2.7.2 | Api.ai | 12 |
| 2.7.3 | Hibernate | 13 |
| 2.7.4 | Database MySQL | 13 |
| 2.7.5 | Strumenti aziendali | 14 |
| 2.8 | Prodotto ottenuto | 14 |
| 3 | Analisi dei requisiti | 17 |
| 3.1 | Tracciamento dei requisiti | 17 |
| 3.2 | Analisi di mercato | 21 |
| 3.2.1 | Conclusioni dell'analisi | 25 |

| | |
|---|-----------|
| 4 Progettazione | 27 |
| 4.1 Studio di api.ai | 27 |
| 4.2 Progettazione agents api.ai | 28 |
| 4.2.1 Gestore di eventi | 28 |
| 4.2.2 Dati ARPA Veneto | 34 |
| 4.2.3 Meteo Veneto Bot | 35 |
| 4.3 Progettazione delle componenti | 41 |
| 4.4 Jaro Winkler distance | 41 |
| 5 Verifica e validazione | 43 |
| 5.1 Verifica e validazione dei chatbots | 43 |
| 6 Conclusioni | 47 |
| 6.1 Raggiungimento degli obiettivi | 47 |
| 6.2 Conoscenze acquisite | 47 |
| 6.3 Valutazione personale | 48 |
| Glossario | 49 |
| Bibliografia | 55 |

Elenco delle figure

| | |
|---|----|
| 1.1 Logo di i-contact s.r.l. | 1 |
| 1.2 Logo di Asana | 4 |
| 1.3 Logo di BeeBole | 4 |
| 1.4 Logo di Slack | 5 |
| 2.1 Modello generico di carosello per Facebook Messenger | 8 |
| 2.2 Diagramma di Gantt | 10 |
| 2.3 Logo di NetBeans | 12 |
| 2.4 Logo di api.ai | 13 |
| 2.5 Logo di Hibernate | 13 |
| 2.6 Esempio di menù di un chatbot di Facebook | 14 |
| 3.1 Funzionamento IBM Watson Conversation | 21 |
| 3.2 Logo di IBM Watson Conversation | 22 |
| 3.3 Esempio di creazione di un intent in wit.ai | 22 |
| 3.4 Logo di wit.ai | 23 |
| 3.5 Logo di Microsoft LUIS | 23 |
| 3.6 Utilizzo di Amazon Lex | 24 |
| 3.7 Logo di Amazon Lex | 25 |
| 3.8 Logo di api.ai | 25 |
| 4.1 dfsf | 28 |
| 4.2 Esempio di | 29 |
| 4.3 Esempio di | 29 |
| 4.4 Esempio di | 30 |
| 4.5 Esempio di | 30 |
| 4.6 Esempio di | 31 |
| 4.7 Esempio di | 31 |
| 4.8 Esempio di | 32 |
| 4.9 Esempio di | 32 |
| 4.10 Esempio di | 33 |
| 4.11 Esempio di | 33 |
| 4.12 Entity conference definita nell'agent | 34 |
| 4.13 Esempio di | 35 |
| 4.14 Esempio di | 36 |
| 4.15 Didascalia comune alle due figure | 36 |
| 4.16 Esempio di | 37 |
| 4.17 Esempio di | 37 |

| | |
|--|----|
| 4.18 Esempio di | 38 |
| 4.19 Esempio di | 38 |
| 4.20 Esempio di | 39 |
| 4.21 Esempio di | 39 |
| 4.22 Esempio di | 40 |
| 4.23 Entity time definita nell'agent | 41 |
| 4.24 Esempio di utilizzo della Jaro Winkler distance | 42 |
| 5.1 Flowchart del processo di verifica dei chatbots | 45 |

Elenco delle tabelle

| | |
|---|----|
| 3.1 Requisiti funzionali generali | 18 |
| 3.2 Requisiti funzionali del chatbot per la gestione degli eventi | 18 |
| 3.3 Requisiti funzionali del chatbot Meteo Veneto Bot | 20 |

Capitolo 1

Introduzione

1.1 Il progetto di stage

1.2 Profilo aziendale

i-contact s.r.l.¹ è un'azienda che nasce nel 2003 a Belluno, dove tuttora mantiene la propria sede. Nei primi anni si dedica allo sviluppo di applicativo web per conto di terzi, lavorando in stretta collaborazione con aziende di comunicazione per consulenze di tipo tecnico. Parallelamente inizia la creazione del prodotto di punta dell'azienda: **SMSHosting**². Si tratta di un *gateway* per l'invio e la ricezione di sms professionali da web, che consente agli utenti di comunicare con i propri clienti direttamente dalla propria area riservata, oppure dall'esterno tramite email, **FTP**, moduli web, software Windows, app per *smartphone* e molto altro.

Nel corso degli anni, e con l'introduzione di nuove tecnologie, l'azienda evolve i propri prodotti, introducendo nuovi strumenti di comunicazione per i propri clienti, focalizzando le proprie competenze sul mondo degli *smartphone*. Viene così creato un prodotto dedicato all'invio di **messaggi push** tramite le più diffuse *messaging apps*, come Facebook Messenger e Telegram. I clienti possono utilizzare questo servizio, integrato nella piattaforma SMSHosting, attraverso email o delle semplici **API REST**. Come ultima novità i-contact s.r.l. ha iniziato lo sviluppo di **chatbot** per Facebook Messenger e Telegram. L'azienda mette a disposizione sia una piattaforma semplice e intuitiva dove un utente può creare il proprio **chatbot** in pochi passi, seguendo i più comuni template di business, sia la possibilità di crearne di nuovi secondo le richieste del cliente.



Figura 1.1: Logo di i-contact s.r.l.

¹ *i-contact s.r.l.* URL: <http://www.i-contact.it/>.

² *SMSHosting.it*. URL: <https://www.smshosting.it/>.

1.3 Tecnologie utilizzate

Le tecnologie principali che vengono utilizzate da i-contact s.r.l. per lo sviluppo dei propri prodotti possono essere divise in tre diverse aree:

- * applicazioni iOS e Android native;
- * applicazioni web con tecnologie Java;
- * [frontend](#).

1.3.1 Applicazioni iOS e Android native

Le tecnologie utilizzate per lo sviluppo di applicazioni mobile dipendono naturalmente dal sistema operativo dove si vuole sviluppare. L'azienda utilizza anche dei [frameworks cross-platform](#) come *PhoneGap*³.

Android

Per quanto riguarda *Android* l'azienda si affida al linguaggio nativo di questo sistema operativo, cioè *Java*. Il team di sviluppo ha grande conoscenza di questo linguaggio, agevolando così lo sviluppo dei nuovi prodotti.

iOS

Per lo sviluppo di applicazioni *iOS* il linguaggio che viene principalmente utilizzato è *Objective-C*, un linguaggio ben noto a coloro che devono codificare questa versione dei prodotti.

1.3.2 Applicazioni web con Java

Le applicazioni web su piattaforma Java sono nel DNA di i-contact s.r.l. fin dalla nascita nel 2003.

Il team di sviluppo ha grande conoscenza dei principali [framework](#) di sviluppo ed ha lavorato a progetti di grande dimensione utilizzando sia [CMS open source](#) (Open CMS) che commerciali (Broadvision, Vignette OpenText). Il [framework Spring](#), *Hibernate ORM*, *Quartz Scheduler*, *Jersey for RESTful Web services* sono solo pochi esempi delle librerie comunemente utilizzate e che fanno parte del *core* dei loro prodotti.

1.3.3 Frontend

Nello sviluppo della parte [frontend](#) dei propri portali e applicativi web, i-contact s.r.l. mira ad utilizzare gli strumenti più innovativi per garantire la massima velocità di presentazione e la compatibilità con i *device* di nuova generazione.

I principali linguaggi e [frameworks](#) sono *HTML5*, *CSS3*, *JQuery*, *Bootstrap* e *Sencha*.

³*PhoneGap*. URL: <https://phonegap.com/>.

1.4 Processi aziendali

1.4.1 Metodologia

Dalla sua nascita ad oggi i-contact s.r.l. crea delle soluzioni software fortemente dipendenti dalle specifiche dei propri clienti. Per potere fare ciò è indispensabile mantenere una stretta collaborazione con il cliente, per capire le sue volontà e le sue richieste in modo preciso. La realizzazione di nuovi progetti deve quindi essere in grado di reagire al cambiamento dei requisiti anche in fase di sviluppo, in quanto il cliente può cambiare idee e giudizi sulle funzionalità del proprio prodotto.

La metodologia di sviluppo adottata dall'azienda è la [Adaptive Software Development](#). Il **metodo agile** utilizzato da i-contact s.r.l. prevede una forte e frequente collaborazione con il cliente, in modo da ricevere dei *feedback* puntuali sugli incrementi portati al prodotto. In ogni momento il cliente si trova ad avere un prodotto via via più completo, che rispecchia i requisiti da esso imposti e discussi con l'azienda.

La chiave per il successo di questa metodologia è racchiusa in questi punti:

- * sviluppare qualcosa di utile;
- * coltivare la fiducia degli **stakeholders**;
- * costituire gruppi di lavoro competenti e collaborativi;
- * far sì che il team abbia la possibilità e sia in grado di prendere decisioni;
- * consegnare spesso nuove versioni all'aggiunta di nuove funzionalità;
- * incoraggiare l'adattabilità;
- * cercare di ottenere l'eccellenza tecnica;
- * quando possibile, aumentare il volume di dati immessi.

1.4.2 Strumenti a supporto dei processi

Gestione di progetto

Per la **gestione di progetto** i-contact s.r.l. utilizza Asana⁴ per l'assegnazione di *task* relativi a nuove attività da svolgere o *bug* da correggere e BeeBole⁵ come *web timesheet*.

Asana Asana è un [SaaS web based](#) che mira a migliorare la collaborazione all'interno dei team di lavoro. Permette infatti la gestione di progetti e *tasks* online, senza dover utilizzare le email.

i-contact s.r.l. ha creato un proprio *workspace*, dove poter aggiungere nuovi progetti e nuovi *tasks* assegnati a questi progetti.

Per ogni *task* che si vuole creare, è possibile specificare una serie di dettagli:

- * **nome e descrizione** relativa;
- * il **membro del team** che ha il compito di svolgere quel task. La persona deve essere registrata all'interno del *workspace* di i-contact s.r.l.;

⁴ Asana. URL: <https://asana.com/>.

⁵ BeeBole. URL: <https://beebole.com/it/>.

- * la **data** entro la quale il compito deve essere svolto. Una settimana prima l'incaricato riceverà una mail di promemoria;
- * uno o più **tags** per differenziare i diversi *tasks* all'interno del progetto.

Una volta assegnato il *task*, incaricato e assegnatario, più tutte le persone che sono in grado di visualizzarlo, potranno aprire una conversazione dedicata nella sezione specifica di quel *task*, dove scambiarsi idee, file e molto altro. Terminato il lavoro, l'incaricato dovrà marcare il *task* come concluso attraverso l'apposita spunta.



Figura 1.2: Logo di Asana

BeeBole BeeBole è uno strumento di *web timesheet* che dà la possibilità ai propri utenti di monitorare in modo efficiente il tempo dedicato a progetti, clienti e incarichi. L'azienda i-contact s.r.l. sfrutta questo applicativo per controllare il budget dedicato ad ogni progetto e le ore effettivamente investite da ogni componente del team, per procedere con una fatturazione corretta e trasparente nei confronti del cliente.



Figura 1.3: Logo di BeeBole

Gestione di versione

L'azienda i-contact s.r.l. utilizza come **sistema di controllo di versione** del codice **Mercurial**. In particolare, per poter raccogliere tutto il codice derivante dai vari progetti delle varie aree sviluppo, viene utilizzata la versione premium di BitBucket⁶.

Alcuni vantaggi dell'adozione di un sistema di versionamento del codice sono:

- * **ridondanza**: ogni sviluppatore possiede un *backup* della *repository* localmente, limitando così la possibilità di perdita totale dei dati;

⁶BitBucket. URL: <https://bitbucket.org/>.

- * **disponibilità:** anche in assenza di connessione alla *repository* principale, è possibile continuare ad effettuare *commit* ed a lavorare. Una volta ripristinata la connessione, la *repository* locale può essere sincronizzata con quella remota rendendo le modifiche visibili a tutti;
- * **branch e merge:** permette con molta facilità la creazione dei cosiddetti *branch*, ossia delle ramificazioni dal prodotto stabile presente nella *repository*, per apportare modifiche o nuove funzionalità evitando di introdurre errori e bug. Una volta che anche questa nuova *feature* risulta corretta, può essere inserita all'interno del progetto principale tramite un *merge*.

Comunicazione aziendale

Come strumento di **comunicazione** tra i dipendenti dell'azienda, i-contact s.r.l. ha scelto Slack⁷. Slack è un software che rientra nella categoria degli strumenti di collaborazione aziendale utilizzato per inviare messaggi in modo istantaneo ai membri del team. Il suo punto di forza è la possibilità di creare dei **canali** dedicati a un progetto o ad un particolare argomento, dove vengono inseriti tutti o solo parte dei dipendenti aziendali. È possibile inoltre comunicare con il team anche attraverso chat individuali o chat con due o più membri.

La scelta è ricaduta su questo strumento anche per la sua facilità di utilizzo e la caratteristica di essere fruibile da tutti i dispositivi *iOs*, *Android*, *Windows* come applicazione e da *web browser*.



Figura 1.4: Logo di Slack

1.5 Clienti

Grazie ai prodotti offerti da i-contact s.r.l. la sua clientela spazia in molti ambiti diversi come automobilismo, banche, telecomunicazioni e molto altro. Queste aziende si rivolgono a i-contact s.r.l. sia per migliorare le comunicazione tra loro e la propria clientela, attraverso la piattaforma di SMSHosting e le sue numerose soluzioni, sia per proporre lo sviluppo di nuovi prodotti *web based*.

Il team di i-contact s.r.l. ha lavorato, direttamente o tramite i loro partner, con clienti in tutta Italia. Grazie ad una consolidata modalità di lavoro riescono infatti a garantire ai clienti progetti di qualità, nei tempi previsti e con costi ridotti evitando trasferte e lavorando principalmente dalla loro sede. Alcuni aziende poi si rivolgono a i-contact s.r.l. per consulenze e *partnership* nel ramo dello sviluppo di prodotti dedicati a *smartphone* e *tablet*, dove il team aziendale eccelle particolarmente.

⁷Slack. URL: <https://slack.com/>.

1.6 Struttura del documento

Il documento è stato strutturato per descrivere in maniera esaustiva il percorso di stage nell'azienda i-contact s.r.l.. I capitoli presenti sono i seguenti:

Il primo capitolo descrive l'azienda i-contact s.r.l., a partire dalla sua storia e dai prodotti che essa offre, fino alle tecnologie che vengono utilizzare giornalmente e le metodologie interne adottate nello sviluppo dei prodotti software.

Il secondo capitolo approfondisce l'argomento dello stage, descrivendolo in modo dettagliato. Vengono inoltre riportati gli strumenti e le tecnologie utilizzate, i problemi affrontati e un breve riassunto sul prodotto ottenuto.

Il terzo capitolo riguarda l'analisi dei requisiti del progetto di stage, riportando una tabella con il loro tracciamento.

Il quarto capitolo descrive l'attività di progettazione svolta per ottenere il prodotto finale. Viene descritto l'utilizzo di api.ai nel progetto e la sua integrazione nel software aziendale.

Il quinto capitolo contiene la descrizione delle attività svolte per verificare e validare il comportamento del prodotto ottenuto.

Il sesto capitolo contiene le mie valutazione finali sull'attività di stage svolta, sulle conoscenze acquisite e un riassunto degli obiettivi e requisiti raggiunti e soddisfatti.

1.6.1 Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- * gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- * le occorrenze dei termini riportati nel glossario vengono evidenziati in blu e contengono un riferimento al termine nel glossario, ad esempio: **API**;
- * i termini in lingua straniera o facenti parte del gergo tecnico sono evidenziati con il carattere *corsivo*;

Capitolo 2

Progetto di stage

L’azienda i-contact s.r.l. ha da sempre mostrato interesse nell’inserimento all’interno del contesto aziendale di stagisti, studenti o altre figure professionali, anche per limitati periodi di tempo. Questa particolare propensione permette all’azienda sia di promuovere nuovi progetti o attività, che con il personale a disposizione non sarebbero percorribili per questioni di tempo, sia introdurre idee e tecnologie innovative per futuri prodotti o per quelli già sviluppati ed utilizzati.

2.1 Chatbot

Un **chatbot** (o chat bot) è un software che dialoga con te. Un assistente virtuale capace di risponderti su una serie di argomenti. Il nome è composto da due parole: chat, dall’inglese *to chat*, che significa chiacchierare, e bot. Da questo si capisce che un **chatbot** è quindi una chat in cui il nostro interlocutore non è un essere umano come noi ma un robot virtuale con il quale possiamo chiacchierare, parlare, o al quale possiamo porre una serie di domande.

Il funzionamento per chi usa questi assistenti personali è molto semplice. La domanda può essere scritta ma può anche pronunciata, nel caso di **chatbot** che comprendono il linguaggio umano. Queste forme di intelligenza artificiale sono ancora in via di perfezionamento, anche se già adesso alcuni **chatbot** possono risultare utili, anche solo per capire quali potranno essere gli sviluppi nei prossimi anni.

Nel caso di Facebook Messenger, i bot assumono il ruolo di intermediari tra una azienda ed un cliente finale, fungendo da supporto automatizzato per le informazioni su prodotti o nella ricerca dei contenuti. I **chatbot** rappresentano dunque un’evoluzione delle relazioni online, perché in grado di fornire autonomamente link e notizie di interesse, supportare l’utente nell’ordine o nella prenotazione, indicare il prodotto più adatto alle singole esigenze o semplicemente rispondere a richieste di informazioni e chiarimenti. Un **chatbot** deve essere associato ad una pagina. Una volta installato, sarà in grado di intercettare ogni singolo messaggio che un utente invia in privato alla pagina Facebook, rispondendo nell’immediato in base alle impostazioni predefinite ed evolvendosi interazione dopo interazione.

2.2 Descrizione del progetto

Negli ultimi anni i-contact s.r.l. si è interessata allo creazione e sviluppo di **chatbot** per Facebook Messenger, aggiungendo così un ulteriore servizio alla piattaforma di SMSHosting: **botbuilder**¹. Questo strumento dà la possibilità ai clienti di creare il proprio **chatbot** autonomamente, senza alcuna conoscenza tecnica, attraverso dei template ottimizzati per i più comuni modelli di business. i-contact s.r.l. mette anche a disposizione la possibilità di sviluppare **chatbot** personalizzati in base alle specifiche dei clienti.

Questa nuova tecnologia è in rapida diffusione in svariati ambiti grazie alla sua facilità di sviluppo, esistono infatti molti *tool* che permettono di costruire il proprio **chatbot** in pochi semplici passi.

Il progetto di stage si inserisce in questo mondo, in quanto l'azienda desiderava introdurre la possibilità per gli utenti di scrivere delle vere e proprie domande al **chatbot**, senza dover utilizzare il menù o i pulsanti dei vari modelli che la piattaforma mette a disposizione.

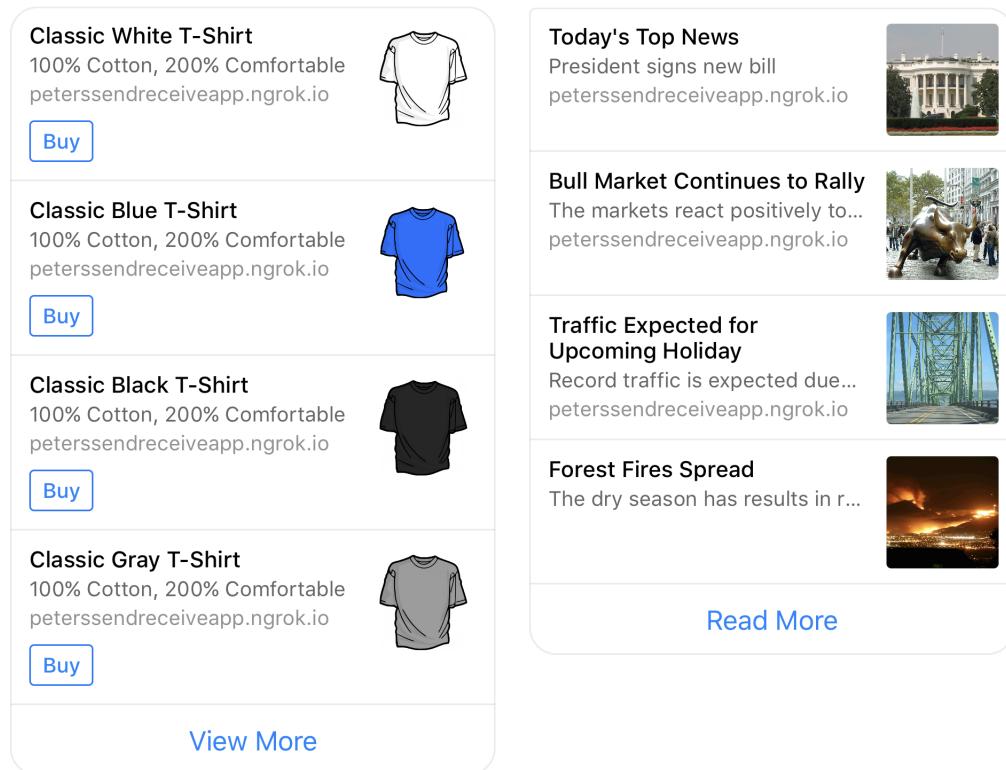


Figura 2.1: Modello generico di carosello per Facebook Messenger

¹ *Botbuilder*. URL: <https://www.smshosting.it/it/chatbot/bot-builder-italiano>.

Il compito assegnato allo stagista è stato quello di integrare questo meccanismo in due diversi **chatbot** creati dall'azienda:

- * **gestore eventi**: è un applicativo creato da i-contact s.r.l. per essere utilizzato durante lo svolgimento di eventi della durata di uno o più giornate. Attraverso questo **chatbot** l'utente può recuperare delle informazioni riguardanti le conferenze in programma, le aule disponibili e le giornate dell'evento;
- * **meteo Veneto bot²**: questo **chatbot** utilizza i dati di ARPA Veneto³ per mostrare le previsioni del tempo del Veneto;

Il progetto si divideva quindi in tre parti:

- * analisi preliminare degli **SDK** delle principali piattaforme per il **NLP** presenti sul mercato, in modo da valutarne pregi e difetti;
- * creazione della logica per la gestione delle domande utente all'interno della piattaforma scelta;
- * integrazione di questa nuova funzionalità nel software utilizzato da i-contact s.r.l. per la gestione dei **chatbot**.

Visto il bisogno di un periodo iniziale di studio, sia per capire le possibilità e i limiti degli strumenti da dover utilizzare nella trasformazione delle **FAQ** in dati processabili, sia per apprenderne a pieno il funzionamento tramite la documentazione presente, i-contact s.r.l. ha ritenuto questo progetto idoneo ad uno studente universitario, il quale ha a disposizione circa 300 ore per formarsi su tutto ciò di cui vi è bisogno, e successivamente portare a termine il prodotto richiesto.

2.3 Modalità di svolgimento

Per quanto riguarda lo svolgimento dello stage è stato deciso di comune accordo che dovesse essere svolto nella sede dell'azienda. Questa decisione mi ha permesso di confrontarmi con personale più esperto, un ambiente nuovo e stimolante e con la vera realtà del mondo del lavoro. Tutti i membri dell'azienda, compreso il tutor, si sono rivelati molto disponibili alle mie richieste di chiarimento su alcuni aspetti dello sviluppo e della gestione del progetto.

Durante la creazione del prodotto sono state mostrate le nuove funzionalità a tutto il team aziendale, in modo da discuterne pregi, difetti e possibili miglioramenti.

L'orario lavorativo è stato il seguente: da lunedì a venerdì, dalle 8:00 alle 12:00 e dalle 14:00 alle 18:00.

2.4 Obiettivi dello stage

Prima di iniziare il periodo di stage, sono stati delineati i seguenti obiettivi, divisi in base alla loro importanza: obbligatori o desiderabili.

² Meteo Veneto Bot. URL: <https://www.smshosting.it/it/blog/chatbot/il-primo-chatbot-sul-meteo-veneto-e-dolomiti>.

³ ARPAV Meteo. URL: <http://www.arpa.veneto.it/>.

2.4.1 Obiettivi obbligatori

- * analisi del problema e studio delle possibili soluzioni già esistenti nel mercato;
- * analisi e confronto delle piattaforme per il **NLP**;
- * progettazione delle categorie di domande che la piattaforma deve gestire;
- * sviluppo client di prova integrato tramite la piattaforma di **NLP**;
- * sviluppo dell'integrazione della piattaforma con Facebook Messenger.

2.4.2 Obiettivi desiderabili

- * test e monitoraggio dell'applicativo tramite l'integrazione con 2 pagine Facebook reali;

2.5 Pianificazione del lavoro

Prima del periodo di stage è stato pianificato, insieme al tutor aziendale, il lavoro da svolgere su base settimanale. Questa suddivisione prevede:

- * **prima settimana:** analisi preliminare del problema e verifica delle soluzioni attualmente disponibili sul mercato;
- * **seconda settimana:** analisi delle piattaforme di **NLP** presenti sul mercato;
- * **terza settimana:** analisi dei requisiti da soddisfare nello sviluppo del prodotto;
- * **quarta settimana:** progettazione della piattaforma scelta e dell'integrazione nel software aziendale;
- * **quinta e sesta settimana:** istruzione della piattaforma per il **NLP** per entrambi i **chatbot**, con la relativa integrazione nel software aziendale;
- * **settima settimana:** *testing* delle funzionalità attraverso 2 pagine di Facebook reali;

La figura 2.2 mostra la pianificazione del lavoro attraverso un [diagramma di Gantt](#).

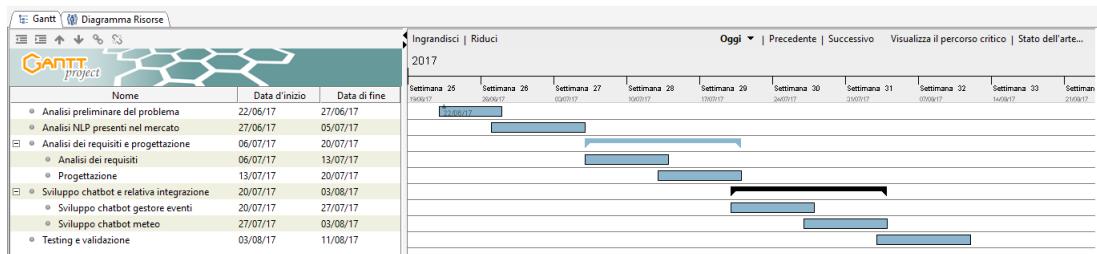


Figura 2.2: Diagramma di Gantt

2.6 Principali problematiche

Durante il periodo di stage, sono state incontrate diverse difficoltà, utili a migliorare la capacità di scelta e decisione dello stagista, ma anche la comunicazione e il lavoro all'interno di un team.

2.6.1 Difficoltà di ambientazione

Nei primi giorni in azienda l'ostacolo principale è stato l'inserimento in un team di lavoro affiatato e di esperienza, nonostante i suoi componenti siano stati molto disponibili e aperti a domande e chiarimenti. In particolare:

- * il **linguaggio tecnico/aziendale** non è stato da subito di facile apprendimento. In parte il problema sorgeva quando i **brainstormings** del team vertevano su tecnologie a me sconosciute, oppure su aspetti tecnici dei loro prodotti. Con il passare del tempo, attraverso l'utilizzo di un piccolo glossario individuale dove segnavo le spiegazioni dei termini a me non chiari, ho assimilato la maggior parte dei concetti, potendo così partecipare alle riunioni del team in modo attivo e costruttivo;
- * lavorare all'interno di un **team** non è un'attività semplice. Bisogna imparare a rispettare le scadenze nel rispetto dei colleghi e saper elaborare strategie nel caso in cui essi non siano riusciti a rispettare le proprie. Grazie all'esperienza che ho maturato sotto questo punto di vista, ho capito quanto sia importante pianificare con cura le attività e il tempo da destinare ad ognuna di esse.

2.6.2 Difficoltà lavorative e soluzioni adottate

Durante lo svolgimento del mio progetto ho riscontrato queste difficoltà principali:

- * scelta del **miglior strumento per il NLP**: il **NLP** è un problema che è stato affrontato relativamente recentemente nel mondo dell'informatica. Le tecnologie che il mercato mette a disposizione sono limitate e spesso non soddisfacenti per gli obiettivi che mi sono stati richiesti. La ricerca di uno strumento idoneo non è stata facile, anche perché l'azienda richiedeva non ci fossero costi per il suo utilizzo. Dopo una buona analisi del mercato, con la produzione di una relativa documentazione consegnata al tutor aziendale, è stato deciso di utilizzare api.ai, servizio acquistato da Google nel 2016, che offre svariate funzionalità senza alcun costo, ma con delle sole limitazioni in termini di richieste giornaliere e mensili (una breve analisi dei principali strumenti di NLP è presente nella sezione [3.2](#));
- * **possibili richieste degli utenti**: il mio compito era quello di fornire una risposta tramite i **chatbots** alle possibili domande di un utilizzatore. Se in primo momento questo non sembrava un *task* troppo complicato, si è però trasformato in un problema non di semplice gestione, in quanto il dominio di richieste di un utente può essere vasto e molto vario. Per cercare di superare questo ostacolo, in accordo con il tutor e il team aziendale, si è deciso di limitare le categorie di domande a cui il **chatbot** risponde, fornendo un messaggio adeguato in caso di una richiesta dell'utente non gestita. Sarà poi compito dell'azienda ampliare le funzionalità dei **chatbots**, visto il poco tempo a mia disposizione, valutando le richieste fatte dagli utenti nell'apposita sezione di Api.ai;

- * **gestione del sessionId:** nel [JSON](#) utilizzato per interrogare api.ai tramite una richiesta [POST](#), deve essere presente obbligatoriamente il campo **sessionId**, per identificare univocamente il mittente della richiesta. La gestione di questo campo è stata quindi di notevole importanza nel garantire ad ogni utente una corretta interazione con il [chatbot](#). Per assegnare ad ogni utente un sessionId univoco è stato deciso di creare un campo nella tabella degli utenti del database, dove memorizzare la stringa rappresentata dall'id dell'utente nel database e il *timestamp* del momento in cui esso si è collegato al bot, garantendo così l'unicità della stessa. In questo modo api.ai, e di conseguenza il [chatbot](#), gestiscono tutti i possibili utenti separatamente e non come un'unica persona;
- * **documentazione scarsa del software aziendale:** dopo aver istruito la piattaforma di api.ai, avevo il compito di integrare le nuove funzionalità nel software aziendale dedicato alla gestione dei [chatbots](#). Il problema è sorto quando non ho avuto a disposizione una chiara documentazione del codice, in modo da capirne *package*, classi e metodi. Per risolvere questa mancanza ho chiesto al mio tutor aziendale una breve spiegazione delle porzioni di codice che avrei dovuto riutilizzare nelle mie nuove classi, potendo così capire il funzionamento generale dei vari *package*. Grazie al mio studio individuale, e alcuni chiarimenti chiesti agli sviluppatori, ho potuto passare autonomamente all'attività di programmazione, introducendo così le mie nuove funzionalità del prodotto.

2.7 Strumenti utilizzati

2.7.1 NetBeans

NetBeans è un ambiente di sviluppo integrato multi-linguaggio, nato nel giugno 2000 e scritto interamente in Java, scelto dalla Oracle Corporation come [IDE](#) ufficiale da contrapporre al più diffuso Eclipse. L'azienda non ha posto nessun vincolo sull'ambiente di sviluppo da adottare, così come vale per i propri dipendenti. Ho deciso di utilizzare NetBeans in quanto è molto intuitivo e di semplice utilizzo.

Inoltre offre una integrazione con [Mercurial](#) per gestire *push*, *pull* e *commit* senza utilizzare le righe di comando.



Figura 2.3: Logo di NetBeans

2.7.2 Api.ai

Api.ai è una società nata nell'ottobre del 2010 e acquisita da Google Inc. nel 2016. Api.ai è una piattaforma di conversazione che permette interazioni sofisticate con il linguaggio naturale. All'interno del progetto è stata utilizzata per trasformare le domande degli utenti in dati processabili, dopo aver creato due *agents*, uno per ciascun

[chatbot](#), ed averli istruiti secondo le possibili [FAQ](#) dei rispettivi ambiti di utilizzo. Api.ai poteva essere integrato nel software aziendale in due diversi modi:

- * **connesso direttamente al chatbot** di Facebook Messenger, grazie alla semplice integrazione prevista da api.ai, estraendo le informazioni dal database aziendale tramite dei *webhook*, anch'essi facilmente gestibili tramite la piattaforma di api.ai;
- * come **strumento esterno** al software aziendale, con il bisogno di interrogarlo tramite delle richieste [POST](#) HTTP solo in alcuni casi specifici, con le domande poste dall'utente.

Dopo aver valutato entrambe le opzioni, ho scelto di intraprendere la seconda per due motivazioni:

- * utilizzando api.ai come strumento esterno non è stato necessario introdurre dipendenze verso di esso nel codice, dando così la possibilità, se ci fosse bisogno in futuro, all'azienda di cambiare questa tecnologia con un'altra senza grossi problemi, cosa che sarebbe stata molto più difficile nell'altro caso;
- * api.ai mette a disposizione un [SDK](#) per Java molto utile, anche se non ben documentato. In ogni caso dopo aver capito il suo funzionamento è stato semplice sfruttarlo per interrogare api.ai con poche righe di codice.



Figura 2.4: Logo di api.ai

2.7.3 Hibernate

Hibernate è una piattaforma [open source](#) ad alto rendimento per lo sviluppo di applicazioni Java, che fornisce il servizio di [ORM](#), ovvero si occupa della mappatura tra le classi Java e le relative tabelle di un database [SQL](#). Gestisce dunque il salvataggio degli oggetti di tali classi ed il reperimento dalle entità dal database, automatizzando le *query* necessarie e provvedendo alla reistanziazione dell'oggetto mappato sul database.



Figura 2.5: Logo di Hibernate

2.7.4 Database MySQL

L'azienda i-contact s.r.l. utilizza un database MySQL per la memorizzazione e gestione dei dati relativi ai progetti dei [chatbots](#). È il più diffuso database [open source](#) basato sul linguaggio [SQL](#) ed è di tipo relazionale, ovvero segue il principio che tutti i dati sono

rappresentati come relazioni e manipolati con gli operatori dell'algebra relazionale o del calcolo relazionale. Lo stagista è stato incaricato di capirne la struttura e come esso veniva gestito all'interno dell'applicativo, per poi utilizzarlo al meglio nello sviluppo del proprio prodotto.

2.7.5 Strumenti aziendali

Infine sono stati utilizzati alcuni degli strumenti consolidati nel contesto aziendale, come **BitBucket** e **Mercurial** per quanto riguarda il versionamento del codice, **Slack** per le comunicazioni con gli altri dipendenti e il tutor ed **Asana** per il tracciamento dei *task* da svolgere (strumenti esposti nella sezione 1.4.2).

2.8 Prodotto ottenuto

Il prodotto che mi è stato richiesto di sviluppare consisteva in un incremento delle funzionalità di due **chatbots**, creati e gestiti da i-contact s.r.l., per offrire agli utenti uno strumento che si avvicinasse il più possibile ad una chat normale con un essere umano.

Al mio arrivo in azienda questi due prodotti potevano essere utilizzati dagli utenti solamente attraverso l'interfaccia grafica che mettono a disposizione. In particolare, per i **chatbot** di Facebook Messenger è possibile creare un menù, usufruibile in qualsiasi momento attraverso il pulsante rappresentato dalle 3 righe orizzontali, dove inserire dei comandi per mostrare determinate categorie di informazioni (ad esempio nel bot per le previsioni del tempo esiste il comando "Previsioni" che mostra il meteo per tutta la settimana, nella zona di interesse).

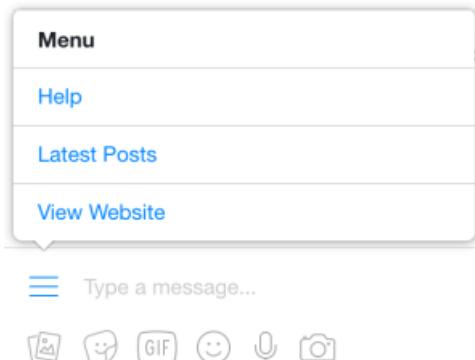


Figura 2.6: Esempio di menù di un **chatbot** di Facebook

Ci sono poi altri "caroselli" messi a disposizione dello sviluppatore per costruire le proprie risposte, spiegati ed illustrati nella documentazione di Messenger-platform⁴.

I risultati che sono riuscito ad ottenere nelle 300 ore di stage sono stati ritenuti molto soddisfacenti dall'azienda, che utilizza già le mie funzionalità nel bot del meteo. Gli incrementi che ho apportato ai due **chatbots** sono esposti di seguito.

⁴ *Messenger-platform documentation.* URL: <https://developers.facebook.com/docs/messenger-platform>.

Meteo Veneto Bot

Si tratta di un **chatbot** innovativo che sfrutta i dati messi a disposizione da ARPA Veneto attraverso un file **XML**. Il compito di questo **chatbot** è di dare delle informazioni all'utente sulle previsioni del tempo nella regione del Veneto, in particolare nella zona di interesse dell'utente che specifica all'inizio della conversazione inviando la propria posizione, o scrivendo il comune che desidera. Dopo aver premuto il pulsante "Previsioni", presente nel menù, si possono visualizzare i dettagli del meteo dei vari giorni della settimana, che mostrano temperature, precipitazione, venti ed ascoltare il bollettino.

Al termine del mio lavoro sono state introdotte nuove funzionalità che permettono all'utente di domandare:

- * il meteo per un determinato giorno in un comune preciso;
- * se è prevista pioggia, neve, sole, nebbia per un determinato giorno in un comune preciso;
- * le temperature previste;
- * di ascoltare il bollettino emesso da ARPA Veneto.

Gestore di eventi

Si tratta di un template di **chatbot** sviluppato da i-contact s.r.l. che viene personalizzato ogni qualvolta un cliente ne richieda l'utilizzo, mantenendo in ogni caso delle funzionalità principali. Il suo compito è quello di mostrare delle informazioni riguardanti l'evento per il quale il **chatbot** è stato personalizzato. In particolare si possono ottenere delle indicazioni sulle conferenze, come orario, durata, argomento, e sulle sale presenti.

Il mio compito in questo caso era di rendere queste funzionalità di immediato utilizzo attraverso delle domande da chiedere al bot. Adesso è possibile chiedere:

- * in **generale**:
 - il programma dell'evento, specificando giorno ora e sala (non obbligatoriamente tutti e tre);
- * per le **conferenze**:
 - la durata;
 - l'orario di inizio e fine;
 - in che sala si svolge;
- * per le **sale**:
 - le indicazioni per trovarla;
 - quale conferenza si svolge in un determinato giorno e orario;
 - la conferenza successiva a quella in corso;

Una volta posta una domanda al **chatbot**, e ricevuta la relativa risposta, è stato anche implementato un sistema che permette all'utente di fare lo stesso tipo di domanda, come se stesse colloquiando con un essere umano. Ad esempio:

- Utente: "Che conferenza si svolge alle 12 in Lum250?"

- Bot: "Analisi matematica"
- Utente: "E in P140?"
- Bot: "Programmazione 1".

Nella seconda risposta il sistema ha tenuto conto della prima domanda fatta dall'utente, cercando così la conferenza che si svolge alle 12 in P140. Questo meccanismo è stato implementato grazie ai *context* di api.ai.

Capitolo 3

Analisi dei requisiti

Il primo giorno di stage il tutor aziendale ha indetto una riunione con tutti i dipendenti per definire le funzionalità del prodotto che dovevo sviluppare. Attraverso questo **brainstorming** ho potuto farmi un'idea più precisa del compito a me assegnato. Una volta terminata l'analisi degli strumenti per il **NLP**, ho potuto svolgere l'analisi dei requisiti del mio prodotto. In accordo con il tutor non ho dovuto stilare un documento formale che comprendesse i casi d'uso, ma solo una lista di requisiti funzionali che il prodotto avrebbe dovuto soddisfare.

3.1 Tracciamento dei requisiti

Nella tabella che seguono verranno presentati i principali requisiti individuati durante l'analisi del problema. Ogni requisito individuato avrà un codice identificativo univoco così formato:

$$R\{Importanza\}\{Codice\}$$

dove:

* **importanza** può assumere uno dei seguenti valori:

- **O**: indica un requisito obbligatorio;
- **D**: indica un requisito desiderabile;

* **codice** indica il codice identificativo del requisito, è univoco e deve essere identificato in forma gerarchica.

La prima tabella contiene i requisiti comuni ai due **chatbot** oggetti dello stage, la seconda tabella riporta i requisiti specifici del **chatbot** per la gestione degli eventi, mentre la terza quelli di Meteo Veneto bot.

| Id Requisito | Descrizione | Importanza |
|---------------------|---|-------------------|
| RO1 | L'utente deve poter interagire con il sistema attraverso delle domande di senso compiuto | Obbligatorio |
| RO2 | Il sistema deve permettere all'utente di interagire con esso nel modo più naturale possibile | Obbligatorio |
| RD3 | Il sistema deve rispondere all'utente sfruttando i modelli messi a disposizione da Facebook Messenger | Desiderabile |
| RD4 | Il sistema deve fornire all'utente un messaggio idoneo, nel caso di una domanda non gestibile | Obbligatorio |

Tabella 3.1: Requisiti funzionali generali

| Id Requisito | Descrizione | Importanza |
|---------------------|--|-------------------|
| RO1.1 | L'utente deve poter chiedere informazioni riguardo una conferenza | Obbligatorio |
| RO1.1.1 | L'utente deve poter chiedere l'orario di inizio e fine di una conferenza | Obbligatorio |
| RO1.1.2 | L'utente deve poter chiedere la durata di una conferenza | Obbligatorio |
| RO1.1.3 | L'utente deve poter chiedere in che aula si svolge la conferenza | Obbligatorio |
| RO1.2 | L'utente deve poter chiedere informazioni riguardo le aule dell'evento | Obbligatorio |
| RO1.2.1 | L'utente deve poter chiedere delle indicazioni per raggiungere un'aula | Obbligatorio |
| RO1.3 | L'utente deve poter chiedere informazioni riguardo il programma dell'evento | Obbligatorio |
| RO1.3.1 | L'utente deve poter chiedere informazioni riguardo il programma dell'evento in una specifica giornata | Obbligatorio |
| RO1.3.2 | L'utente deve poter chiedere informazioni riguardo il programma dell'evento in una specifica giornata ed in una precisa stanza | Obbligatorio |
| RO1.3.3 | L'utente deve poter chiedere informazioni riguardo il programma dell'evento in una specifica ora ed in una precisa stanza | Obbligatorio |
| RO1.3.4 | L'utente deve poter chiedere informazioni riguardo il programma dell'evento specificando il giorno, l'ora e l'aula | Obbligatorio |
| RD5 | L'utente deve poter ricevere informazioni sul funzionamento del chatbot scrivendo "aiuto" | Desiderabile |
| RO6 | L'utente deve poter visualizzare la propria agenda scrivendo "agenda" | Obbligatorio |

Tabella 3.2: Requisiti funzionali del chatbot per la gestione degli eventi

| Id Requisito | Descrizione | Importanza |
|--------------|--|--------------|
| RO1.4 | L'utente deve poter chiedere le previsioni del meteo | Obbligatorio |
| RO1.4.1 | L'utente deve poter chiedere le previsioni del meteo in un determinato comune del Veneto | Obbligatorio |
| RO1.4.2 | L'utente deve poter chiedere le previsioni del meteo per un determinato giorno o periodo di tempo(es. weekend) | Obbligatorio |
| RO1.4.3 | L'utente deve poter chiedere le previsioni del meteo per un determinato giorno o periodo di tempo(es. wweekend), in un preciso comune del Veneto | Obbligatorio |
| RO1.5 | L'utente deve poter chiedere le temperature previste | Obbligatorio |
| RO1.5.1 | L'utente deve poter chiedere le temperature previste in un determinato giorno o periodo di tempo | Obbligatorio |
| RO1.5.2 | L'utente deve poter chiedere le temperature previste in un determinato comune del Veneto | Obbligatorio |
| RO1.5.3 | L'utente deve poter chiedere le temperature previste in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |
| RO1.6 | L'utente deve poter chiedere se è previsto il sole | Obbligatorio |
| RO1.6.1 | L'utente deve poter chiedere se è previsto il sole in un determinato giorno o periodo di tempo | Obbligatorio |
| RO1.6.2 | L'utente deve poter chiedere se è previsto il sole in un determinato comune del Veneto | Obbligatorio |
| RO1.6.3 | L'utente deve poter chiedere se è previsto il sole in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |
| RO1.7 | L'utente deve poter chiedere se è prevista pioggia | Obbligatorio |
| RO1.7.1 | L'utente deve poter chiedere se è prevista pioggia in un determinato giorno o periodo di tempo | Obbligatorio |
| RO1.7.2 | L'utente deve poter chiedere se è prevista pioggia in un determinato comune del Veneto | Obbligatorio |
| RO1.7.3 | L'utente deve poter chiedere se è prevista pioggia in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |
| RO1.8 | L'utente deve poter chiedere se è prevista neve | Obbligatorio |
| RO1.8.1 | L'utente deve poter chiedere se è prevista neve in un determinato giorno o periodo di tempo | Obbligatorio |
| RO1.8.2 | L'utente deve poter chiedere se è prevista neve in un determinato comune del Veneto | Obbligatorio |
| RO1.8.3 | L'utente deve poter chiedere se è prevista neve in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |

| Id Requisito | Descrizione | Importanza |
|--------------|--|--------------|
| RD1.9 | L'utente deve poter chiedere se è previsto bel tempo | Obbligatorio |
| RD1.9.1 | L'utente deve poter chiedere se è previsto bel tempo in un determinato giorno o periodo di tempo | Desiderabile |
| RD1.9.2 | L'utente deve poter chiedere le se è previsto bel tempo in un determinato comune del Veneto | Obbligatorio |
| RD1.9.3 | L'utente deve poter chiedere se è previsto brutto tempo in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |
| RD1.10 | L'utente deve poter chiedere se è previsto brutto tempo | Obbligatorio |
| RD1.10.1 | L'utente deve poter chiedere se è previsto brutto tempo in un determinato giorno o periodo di tempo | Desiderabile |
| RD1.10.2 | L'utente deve poter chiedere le se è previsto brutto tempo in un determinato comune del Veneto | Obbligatorio |
| RO1.10.3 | L'utente deve poter chiedere se è previsto brutto tempo in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |
| RO1.11 | L'utente deve poter chiedere se è prevista nebbia tempo | Obbligatorio |
| RO1.11.1 | L'utente deve poter chiedere se è prevista nebbia tempo in un determinato giorno o periodo di tempo | Desiderabile |
| RO1.11.2 | L'utente deve poter chiedere le se è prevista nebbia tempo in un determinato comune del Veneto | Obbligatorio |
| RD1.11.3 | L'utente deve poter chiedere se è prevista nebbia tempo in un determinato comune del Veneto, per uno specifico giorno o periodo di tempo | Obbligatorio |
| RO1.12 | L'utente deve poter chiedere se sono presenti fenomeni particolari o avvisi speciali | Obbligatorio |
| RO7 | Il sistema deve fornire le informazioni del comune in cui l'utente si è registrato, nel caso non ne specifichi un altro nella domanda | Obbligatorio |
| RO8 | L'utente deve poter ricevere l'audio del bollettino meteo scrivendo "ascolta bollettino" | Obbligatorio |

Tabella 3.3: Requisiti funzionali del chatbot Meteo Veneto Bot

3.2 Analisi di mercato

Il primo passo da compiere per iniziare lo sviluppo del prodotto è stato scegliere la piattaforma di **NLP** migliore in base ai requisiti imposti dall'azienda. Le richieste fatte da i-contact s.r.l. riguardanti questo strumento erano le seguenti:

- * **costo:** il prezzo per il suo utilizzo doveva essere uguale a 0;
- * **lingua:** deve supportare la lingua italiana, visto che al momento attuale i **chatbots** sono implementati solo con quella;
- * **documentazione:** il servizio deve essere ben documentato per permettere all'azienda, una volta finito il periodo di stage, di imparare ad utilizzarlo velocemente.

Questa attività di analisi di mercato si è rivelata quindi fondamentale per la buona riuscita del progetto, visto l'importanza che questo strumento avrebbe avuto nell'intero periodo di sviluppo. Le piattaforme da me studiate e analizzate sono riportate di seguito.

IBM Watson Conversation

IBM Watson Conversation¹ è un prodotto della piattaforma IBM Watson, che attraverso IBM Cloud dà la possibilità di integrare i più potenti mezzi di **AI** nelle tue applicazioni. Il servizio di Conversation, oltre alla possibilità di creare **chatbot** e agenti virtuali, può essere istruito ed interrogato per analizzare il testo posto in input, attraverso le **API** messe a disposizione.

È possibile infatti creare dei *workspace* dedicati dove, attraverso *intent* ed *entities* creati e gestiti dallo sviluppatore, analizzare le domande poste dagli utenti, estraendo i dati che più interessano. L'integrazione con l'applicativo aziendale risultava semplice, grazie al **SDK** di Java² messo a disposizione da IBM.

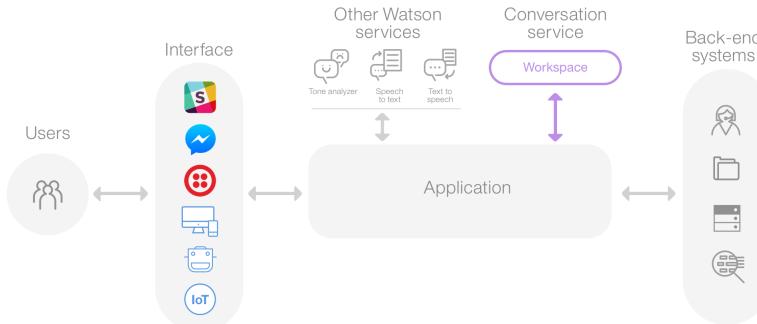


Figura 3.1: Funzionamento IBM Watson Conversation

Per quanto riguarda le richieste dell'azienda:

- * la **lingua italiana** è supportata, e non in versione beta;

¹IBM Watson Conversation. URL: <https://www.ibm.com/watson/services/conversation-2/>.

²IBM SDK for Java. URL: <https://github.com/watson-developer-cloud/java-sdk/tree/develop/conversation>.

- * la **documentazione** è chiara ed esaustiva, con dei video di esempio molto utili;
- * esiste un piano di **costi** gratis, chiamato *Lite*, che però dà la possibilità di creare un numero limitato di *workspace*, *intent* ed *entity*, risultando troppo vincolante per i futuri utilizzi aziendali. Le soluzioni a pagamento non sono state prese in considerazione in quanto non percorribili per l'azienda, almeno in un primo momento di utilizzo di questi servizi.



Figura 3.2: Logo di IBM Watson Conversation

wit.ai

wit.ai³ è una società nata nell'ottobre del 2013 e acquisita da Facebook Inc. nel 2015. L'obiettivo di wit.ai è quello di semplificare la creazione di applicazioni che prevedono interazioni testuali o vocali; per farlo viene messa a disposizione degli sviluppatori una piattaforma di linguaggio naturale aperta ed estensibile che ha la peculiarità di apprendere tramite ogni interazione eseguita.

wit.ai mette a disposizione un **SDK** gratuito ed **open source** per il riconoscimento del linguaggio naturale. Questa piattaforma è caratterizzata dall'utilizzo di *context*, *intent* ed *entity* che sono dei costrutti messi a disposizione per tradurre le richieste vocali dell'utente in dati processabili. In particolare il *context* si utilizza per monitorare lo stato della conversazione tra l'utente e wit.ai.

Figura 3.3: Esempio di creazione di un intent in wit.ai

Per quanto riguarda le richieste dell'azienda:

- * la **lingua italiana** è supportata;

³wit.ai. URL: <https://wit.ai/>.

- * la **documentazione** è chiara ed esaustiva;
- * l'utilizzo di wit.ai è completamente **gratuito** per progetti sia pubblici che privati.

Dal punto di vista tecnico l'unica mancanza di questo strumento, che ha influito nella decisione di non adottarlo, è la impossibilità di impostare delle *required entity* all'interno degli *intent*. Questo aspetto obbliga lo sviluppatore a introdurre dei controlli a livello di *business logic*, che altrimenti non sarebbero necessari, come nel caso di altre piattaforme che saranno esposte successivamente.



Figura 3.4: Logo di wit.ai

Microsoft LUIS

Microsoft LUIS (Language Understanding Intelligent Service)⁴ è un prodotto di *Microsoft Azure*, dedicato a comprendere le richieste di una persona tramite un *language model* (entity/intent).

Come nelle altre piattaforme lo sviluppatore può creare degli *intents*, cioè delle categorie di azioni che l'utente può intraprendere, dove nelle frasi ad esse correlate vengono evidenziate le *entities*, ossia i pezzi di informazione di interesse, per poi gestirle nella logica del **chatbot**. LUIS inoltre mette a disposizione la possibilità di "marcare" le *entity* come *required*, al contrario di wit.ai, e anche la creazione di cosiddette *composite entities*, che possono essere intese come il raggruppamento di più *entity* in una unica.

Per quanto riguarda le richieste dell'azienda:

- * la **lingua italiana** è supportata;
- * la **documentazione** è abbastanza chiara;
- * esiste un piano **gratuito** di utilizzo di LUIS, con una limitazione del numero di chiamate alle API.



Figura 3.5: Logo di Microsoft LUIS

⁴luis.

Amazon Lex

Amazon Lex⁵ è un servizio per la creazione di interfacce di comunicazione tramite voce e testo per qualsiasi tipo di applicazione. Amazon Lex offre funzionalità avanzate di apprendimento approfondito per il riconoscimento vocale e la dettatura, nonché per il riconoscimento del linguaggio naturale e la comprensione di testi, consentendo la creazione di applicazioni coinvolgenti e conversazioni realistiche. Con Amazon Lex, le stesse tecnologie di apprendimento approfondito su cui si basa *Amazon Alexa* sono disponibili a tutti gli sviluppatori, consentendo così la creazione di *chatbots* sofisticati e naturali in modo semplice e veloce.

L'interfaccia grafica consente di creare i propri *intents* in modo intuitivo, seguendo le linee generali delle altre piattaforme.

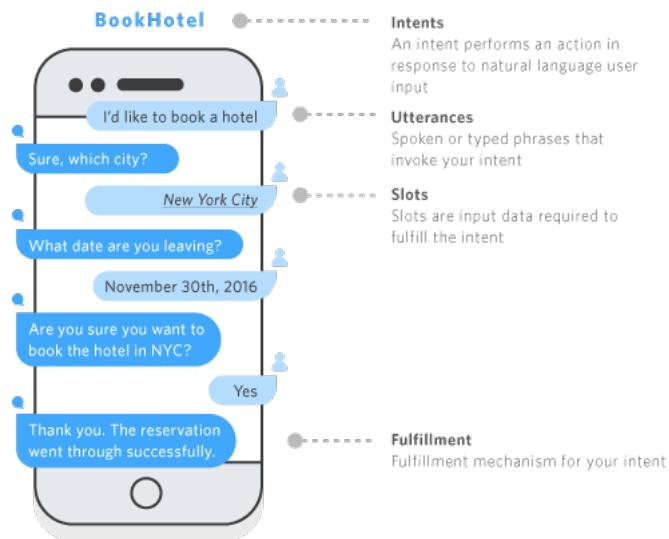


Figura 3.6: Utilizzo di Amazon Lex

Per quanto riguarda le richieste dell'azienda:

- * la **lingua italiana** è supportata;
- * la **documentazione** non è chiara, soprattutto per quanto riguarda la creazione di *intent*, *entity* e *utterance*;
- * esiste un piano **gratuito** per il primo anno di utilizzo di Amazon Lex. Finito, la piattaforma diventa a pagamento in proporzione al numero di chiamate effettuate.

⁵Amazon Lex. URL: <https://aws.amazon.com/it/lex/>.

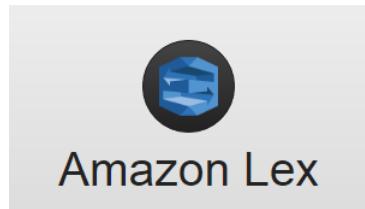


Figura 3.7: Logo di Amazon Lex

Api.ai

Api.ai⁶ è uno degli strumenti con il maggior numero di *features* per quanto riguarda il *machine learning* e il **NLP**. Una volta acquistato da Google nel 2016 il suo volume di utilizzatori è aumentato in maniera esponenziale.

Api.ai fornisce **SDK** per i principali linguaggi di programmazione tra i quali *C++*, *C#*, *Java*, *Node.js*, *JavaScript* e *Python*. Inoltre può essere integrato con *Amazon Echo* e *Microsoft Cortana*. Le applicazioni sviluppate su questa piattaforma sono costituite da *agent*, i quali si occupano di trasformare il linguaggio naturale in dati processabili. Tali *agent* sono a loro volta costituiti da *intent*, che hanno il compito di associare la richiesta dell'utente ad una determinata azione del software, ed *entity*, che sono strumenti per estrarre dal linguaggio naturale i parametri attesi.

Per quanto riguarda le richieste dell'azienda:

- * la **lingua italiana** è supportata;
- * la **documentazione** è molto chiara;
- * il suo utilizzo è **gratuito**, con delle limitazioni per il numero di richieste al minuto, ora, giorno e mese.



Figura 3.8: Logo di api.ai

3.2.1 Conclusioni dell'analisi

Dopo una mia attenta analisi sui pregi e difetti di tutti gli strumenti, in accordo con il tutor aziendale, è stata fatta una breve riunione con gli altri dipendenti, dove ho illustrato in modo sintetico i dati raccolti. Amazon Lex e IBM Watson Conversation sono stati considerati non adeguati per lo sviluppo del progetto a causa dei loro costi, mentre le altre piattaforme, non presentando queste limitazioni, potevano essere tutte adottate.

La mia proposta è stata quella di utilizzare api.ai, per la maggiore stabilità rispetto a wit.ai, per il maggior numero di lingue disponibili in vista di un supporto futuro ad interazioni con utenti di diversa nazionalità e per la maggiore maturità della piattaforma.

⁶api.ai. URL: <https://api.ai/>.

L'azienda dopo aver valutato anch'essa i dati raccolti ha deciso di approvare la mia proposta, in quanto api.ai rispettava tutte le sue richieste iniziali, mostrando delle potenzialità molto interessanti per lo sviluppo del prodotto.

Capitolo 4

Progettazione

La progettazione nel caso del prodotto che ho dovuto sviluppare si è concentrata particolarmente sulla piattaforma di api.ai, in quanto rappresentava la maggior parte del lavoro. Successivamente sono passato alla progettazione delle classi da introdurre nel codice già creato dall'azienda, per gestire le nuove funzionalità dei [chatbots](#).

4.1 Studio di api.ai

Prima di passare all'attività di progettazione è stato fondamentale analizzare e studiare a fondo le possibilità che api.ai mette a disposizione per la creazione del prodotto a me richiesto. I concetti base per capire il funzionamento di api.ai sono quattro:

- * **agent**: gli *agents* sono meglio descritti come moduli NLU (Natural Language Understanding). Questi possono essere inclusi nell'applicazione, nel prodotto o nel servizio e trasformano le richieste di utenti naturali in dati attivi. Questa trasformazione si verifica quando un input utente corrisponde a uno degli *intent* all'interno dell'*agent*;
- * **intent**: sono una mappatura tra quello che l'utente può scrivere in input e l'azione che il software deve intraprendere. Un *intent* è formato dalle seguenti sezioni:
 - **user says**: perché l'*agent* capisca la domanda, sono necessari esempi di come la stessa domanda può essere posta in modi diversi. Lo sviluppatore aggiunge queste permutazioni alla sezione *user says* dell'*intent*. Più variazioni vengono aggiunte all'*intent*, meglio l'*agent* comprenderà l'utente;
 - **action**: contiene il nome della *action*, che può essere utilizzato per attivare una particolare funzione del prodotto, e la tabella dei **parameters**. I *parameters* possono essere intesi come gli elementi che collegano le parole nelle *user says* alle *entities*;
 - **response**: in questa sezione è possibile definire la risposta di api.ai quando l'*intent* viene attivato. Non è stato quasi mai utilizzato, in quanto la risposta viene generata nella *business logic*.
- * **context**: i *context* rappresentano il contesto corrente della richiesta di un utente. Ciò è utile per differenziare frasi che possono essere vaghe o avere significati diversi a seconda delle preferenze dell'utente, della posizione geografica, della pagina

corrente di un'applicazione o dell'argomento della conversazione. È possibile impostare un *lifespan* ad ognuno di essi per definire dopo quante richieste il *context* deve scadere;

- * **entity**: le entities sono strumenti potenti utilizzati per estrarre i valori dei parametri dagli input degli utenti. Tutti i dati importanti che si desidera ottenere dalla richiesta di un utente, avranno un'entità corrispondente. Le *system entities* sono entità pre-costruite fornite da api.ai per facilitare la gestione dei più comuni concetti (luoghi, orari, colori, ecc..). È possibile poi definire le proprie *entities* in base alle necessità dello sviluppatore;

4.2 Progettazione agents api.ai

Durante la progettazione degli *agents* per api.ai è stato necessario definire tutti gli *intents* utili a soddisfare i requisiti definiti durante l'analisi dei requisiti. Il passo successivo è stato quello di progettare le *user says* per ogni *intent* e le relative *entity*.

4.2.1 Gestore di eventi

Intent

Per quanto riguarda la progettazione del *chatbot* dedicato alla gestione di eventi, gli *intents* che mi sono serviti per soddisfare tutti i requisiti sono stati i seguenti:

- * **durata_conferenza**: permette all'utente di domandare la durata di una conferenza e viene attivato con domande come: "Quanto dura la conferenza Y?". La risposta del *chatbot* contiene il nome, l'inizio, la fine e la durata (in minuti o in ore) della conferenza richiesta dall'ospite;

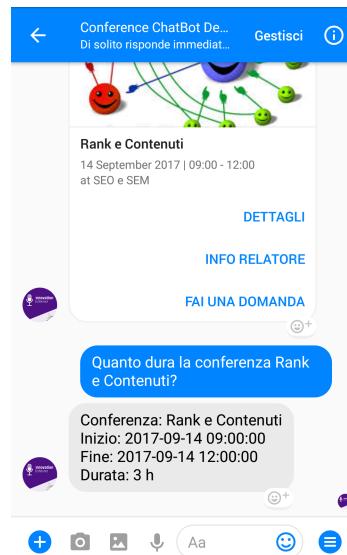


Figura 4.1: dffsf

- * **luogo_conferenza:** permette all'utente di domandare il luogo dove si svolgerà la conferenza. La risposta del **chatbot** contiene un carosello predefinito da Messenger, con tutte le informazioni sull'aula in questione;

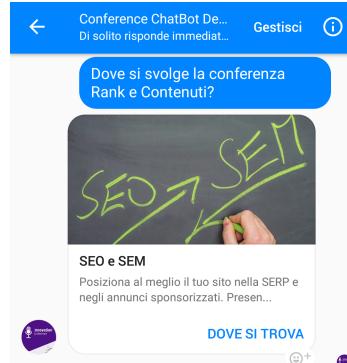


Figura 4.2: Esempio di

- * **ora_conferenza:** permette all'utente di chiedere l'orario di inizio e di fine di una conferenza. La risposta del **chatbot** contiene un carosello predefinito da Messenger, con tutte le informazioni sulla conferenza;



Figura 4.3: Esempio di

- * **indicazioni_stanza:** permette all'utente di domandare le indicazioni per trovare una determinata aula. La risposta del **chatbot** contiene le indicazioni presenti nel database, con una piccola mappa illustrativa;



Figura 4.4: Esempio di

- * **programma_giornata:** permette all'utente di domandare il programma dell'evento di un determinato giorno. La risposta del **chatbot** contiene un carosello per ogni conferenza in programma quel giorno;

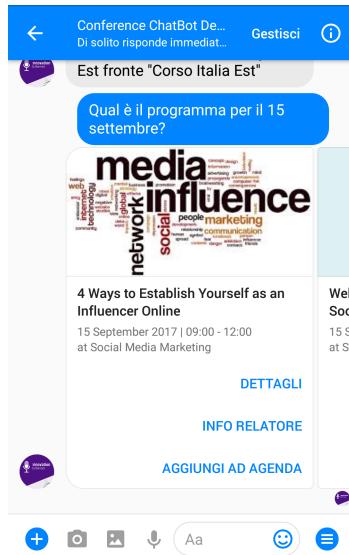


Figura 4.5: Esempio di

- * **programma_no_data:** permette all'utente di domandare quale conferenza si sta svolgendo in quel momento in una determinata stanza. La risposta del chatbot contiene un carosello con la conferenza in programma;

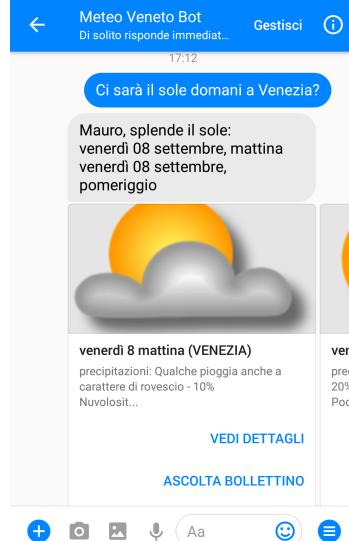


Figura 4.6: Esempio di

- * **data_ora_stanza_conferenza:** permette all'utente di domandare la conferenza in programma specificando data, ora e aula. La risposta del chatbot contiene un carosello con la conferenza in programma, se presente;

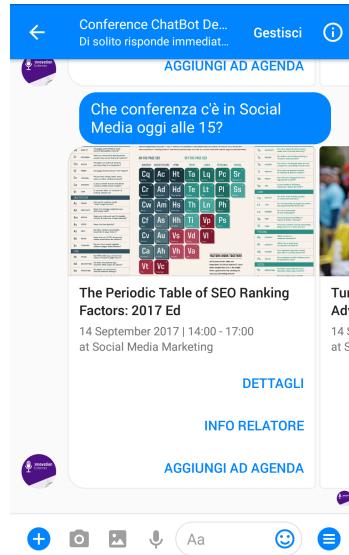


Figura 4.7: Esempio di

- * **data_stanza_conferenza:** permette all'utente di domandare la conferenza in programma specificando data e aula. La risposta del **chatbot** contiene un carosello con la conferenza in programma, se presente;



Figura 4.8: Esempio di

- * **ora_stanza_conferenza:** permette all'utente di domandare la conferenza in programma specificando ora e aula. La risposta del **chatbot** contiene un carosello con la conferenza in programma, se presente;

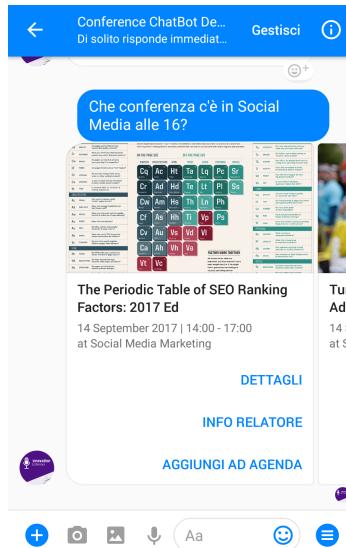


Figura 4.9: Esempio di

- * **visualizza_agenda:** permette all'utente di visualizzare la propria agenda (cioè le conferenze che ha aggiunto precedentemente). La risposta contiene tutte le conferenze aggiunte all'agenda;

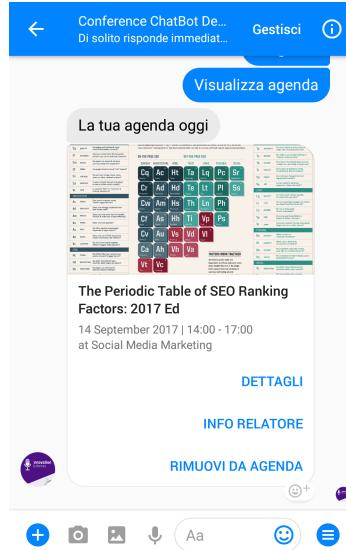


Figura 4.10: Esempio di

- * **richiesta_aiuto:** permette all'utente di ottenere delle informazioni per l'utilizzo del **chatbot**. La risposta contiene una breve spiegazione e delle domande che possono essere rivolte al bot.

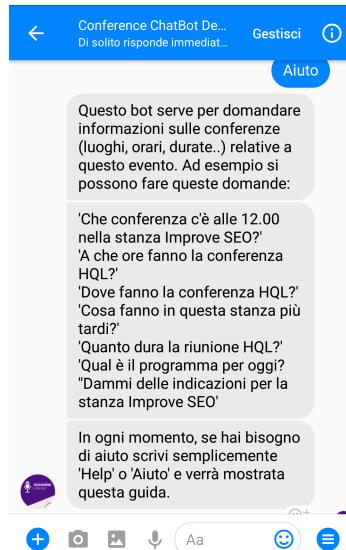


Figura 4.11: Esempio di

Entity

Nella progettazione di questo *agent* ho creato queste nuove *entities*:

- * **conference**: viene utilizzata per gestire i sinonimi della parola "conferenza". In questo modo scrivere assemblea, meeting ha lo stesso risultato di conferenza;
- * **stanza**: viene utilizzata per gestire i sinonimi della parola "stanza". In questo modo l'utente può scrivere aula, sala, padiglione con lo stesso risultato di stanza;
- * **my_time**: questa *entity* è formata da `@sys.time`, ossia una *system entity* fornita da api.ai per estrarre un orario dall'input dell'utente, e una serie di espressioni per indicare il momento attuale in cui l'utente fa la domanda (adesso, in questo momento, ora). In questo modo se l'utente scrive "*Cosa fanno in aula X adesso?*", nel **JSON** ritornato da api.ai ci sarà un parametro di tipo *my_time* con valore "adesso", che potrà essere gestito nella *business logic*.

The screenshot shows the api.ai interface with the sidebar navigation bar. The 'Entities' tab is selected. On the right, a table lists various words and their corresponding synonyms under the 'conference' entity. A 'SAVE' button is visible at the top right of the entity editor.

| synonym | definition |
|------------|-------------------------|
| conferenza | conferenze, conferenza |
| evento | evento, eventi |
| riunione | riunione, riunioni |
| meeting | meeting |
| convegno | convegno, convegni |
| convention | convention, conventions |
| congresso | congresso, congressi |
| assemblea | assemblea, assemblee |

Figura 4.12: Entity conference definita nell'agent

4.2.2 Dati ARPA Veneto

Il **chatbot** dedicato al meteo è stato realizzato grazie agli *open data* messi a disposizione dall'Agenzia Regionale per la Prevenzione e Protezione Ambientale del Veneto(ARPAV). Ogni giorno, nel sito ufficiale¹, vengono emessi tre bollettini:

- * **alle 9:00**: che rappresenta un aggiornamento del bollettino del giorno precedente;
- * **alle 13:00**: il nuovo bollettino;
- * **alle 16:00**: un aggiornamento del bollettino emesso alle 13.

Il file **XML** che è possibile scaricare, contiene queste informazioni:

- * le previsioni dei cinque giorni successivi per le 18 zone in cui è stata divisa la regione del Veneto;

¹ ARPAV Meteo.

- * una descrizione dell’evoluzione generale dei cinque giorni successivi, per tre macro zone: la regione intera, la zona delle Dolomiti e la pianura veneta.

Ad ogni nuova emissione del bollettino, i nuovi dati vengono inseriti nel database aziendale, in modo da comunicare agli utenti solamente le notizie più aggiornate.

4.2.3 Meteo Veneto Bot

Intent

Gli intents che ho deciso di creare per soddisfare tutti i requisiti sono i seguenti:

- * **richiesta_meteo**: permette all’utente di chiedere le previsioni del meteo specificando una giornata o un periodo di tempo (es. weekend) e il comune di interesse (se non viene specificato, si considera il comune da lui selezionato all’inizio dell’interazione con il **chatbot**). La risposta contiene un carosello con il meteo richiesto.



Figura 4.13: Esempio di

* **richiesta_sole**: permette all'utente di chiedere se è previsto il sole in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta è formata da due messaggi: il primo mostra le giornate dove è previsto il sole, tra quelle richieste dall'utente, il secondo contiene i caroselli delle previsioni.



Figura 4.14: Esempio di

* **richiesta_pioggia**: permette all'utente di chiedere se è prevista pioggia in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta è formata da due messaggi: il primo mostra le giornate dove è prevista pioggia, tra quelle richieste dall'utente, il secondo contiene i caroselli delle previsioni.

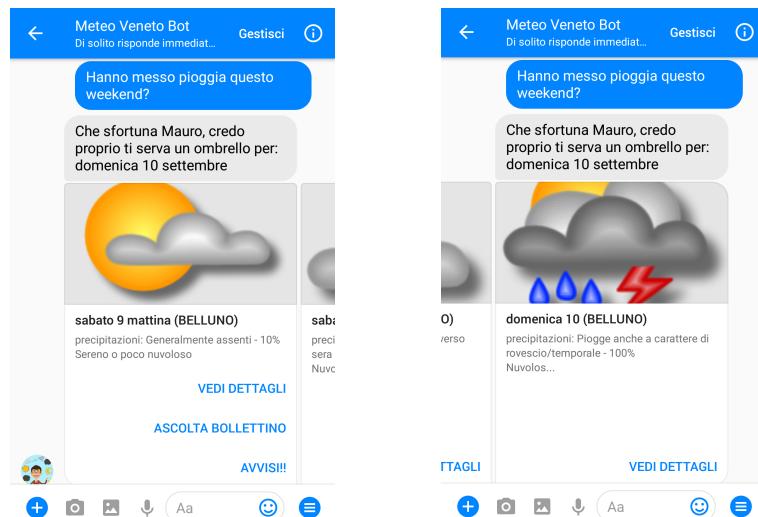


Figura 4.15: Didascalia comune alle due figure

- * **richiesta_nebbia:** permette all'utente di chiedere se è prevista nebbia in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta è formata da due messaggi: il primo mostra le giornate dove è prevista nebbia, tra quelle richieste dall'utente, il secondo contiene i caroselli delle previsioni.



Figura 4.16: Esempio di

- * **richiesta_neve:** permette all'utente di chiedere se è prevista neve in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta è formata da due messaggi: il primo mostra le giornate dove è prevista neve, tra quelle richieste dall'utente, il secondo contiene i caroselli delle previsioni.



Figura 4.17: Esempio di

- * **richiesta_bel_tempo:** permette all'utente di chiedere se è previsto bel tempo in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta è formata da due messaggi: il primo mostra le giornate dove è previsto bel tempo, tra quelle richieste dall'utente, il secondo contiene i caroselli delle previsioni.



Figura 4.18: Esempio di

- * **richiesta_brutto_tempo:** permette all'utente di chiedere se è previsto brutto tempo in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta è formata da due messaggi: il primo mostra le giornate dove è previsto brutto tempo, tra quelle richieste dall'utente, il secondo contiene i caroselli delle previsioni.

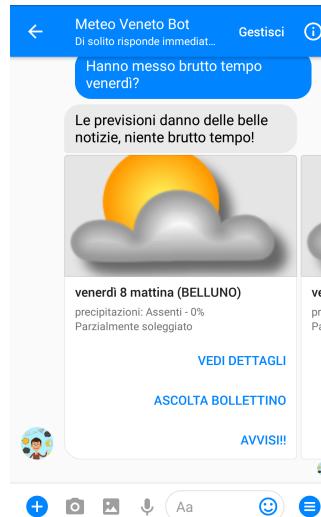


Figura 4.19: Esempio di

- * **richiesta_temperature:** permette all'utente di chiedere le temperature previste in una specifica giornata o un periodo di tempo (es. weekend), in un determinato comune. La risposta contiene le temperature massimi e minime previste fornite da ARPA Veneto.



Figura 4.20: Esempio di

- * **ascolta_bollettino:** permette all'utente di chiedere il bollettino audio emesso da ARPA Veneto ogni giorno. La risposta contiene il file audio richiesto.



Figura 4.21: Esempio di

- * **fenomeni_particolari**: permette all'utente di chiedere se sono presenti avvisi o fenomeni particolari emessi da ARPAV. La risposta contiene questi avvisi, se presenti.

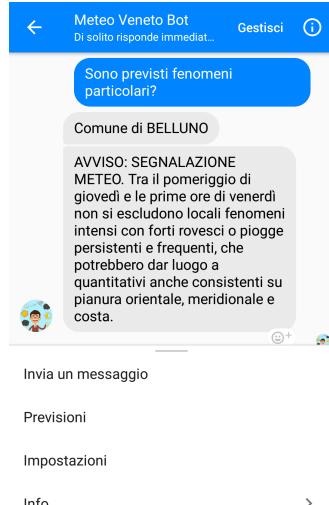


Figura 4.22: Esempio di

Entity

Nella progettazione di questo *agent* ho creato queste nuove *entities*:

- * **fenomeni atmosferici**: ho creato quattro *entities* per **sole**, **pioggia**, **neve**, **nebbia**, in modo da gestire i possibili sinonimi che un utente può scrivere;
- * **time**: si tratta di una *composite entity* per gestire gli orari e le date scritte dall'utente. In particolare questa *entity* combina tre *system entity* fornite da api.ai:
 - **@sys.date**: per le date in formato standard (es. 31 agosto);
 - **@sys.date-period**: per un periodo di tempo formato da più giorni (es. weekend);
 - **@sys.time-period**: per una parte del giorno (es. mattina, pomeriggio, notte);

Combinando queste *entities* l'utente può specificare precisamente le previsioni che desidera ricevere. Può infatti porre al **chatbot** domande del tipo: "Che tempo farà domani sera?", "Dimmi le previsioni per il prossimo weekend", "Ci sarà il sole venerdì pomeriggio?".

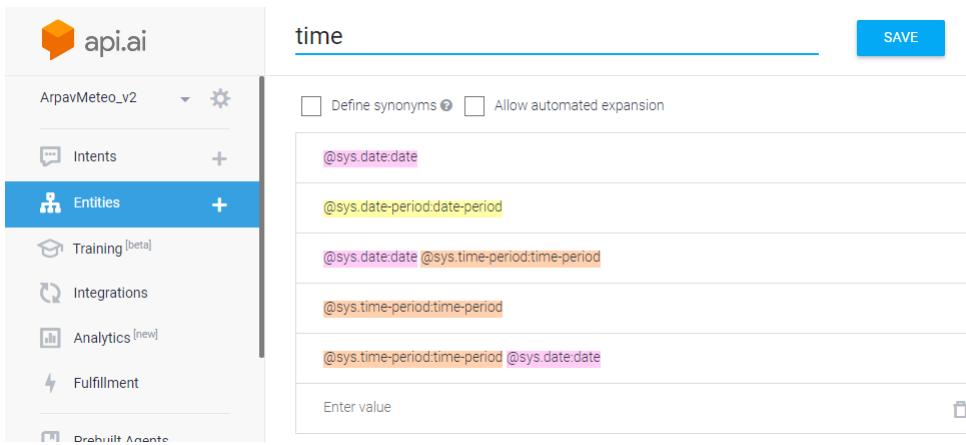


Figura 4.23: Entity time definita nell'agent

4.3 Progettazione delle componenti

Per quanto riguarda la progettazione delle componenti da aggiungere al software aziendale ho deciso di creare due classi per ciascun **chatbot**, con un comportamento simile. La prima ha il compito di:

- * **interrogare api.ai**, costruendo il messaggio da inviare con il token dell'agent, l'input e il sessionId dell'utente;
- * **analizzare la risposta**, invocando il metodo corretto a seconda del valore del campo *action* del **JSON**.

La seconda classe invece contiene tutti i metodi che rappresentano la logica del **chatbot**, analizzando i *parameters* del **JSON** ritornato da api.ai e formulando la risposta da visualizzare nella chat. Questi metodi hanno riutilizzato dei componenti già presenti nel software, per costruire modelli di risposta già previsti nel **chatbot** prima del mio arrivo.

4.4 Jaro Winkler distance

Durante l'attività di progettazione mi sono reso conto come fosse necessario gestire un possibile errore di scrittura dell'utente in una delle sue domande, soprattutto nelle parole fondamentali per formulare le risposte, come ad esempio il nome di un comune per il **chatbot** del meteo o il nome di una conferenza in quello degli eventi. In un primo momento infatti l'input dell'utente veniva utilizzato direttamente nelle *query SQL* per interrogare il database ed ottenere i dati di interesse, soprattutto attraverso l'operatore "*LIKE*". In questo modo però non è possibile gestire il caso in cui un utente scriva ad esempio il comune "Padvoa", intendendo Padova.

Per ovviare a questo problema quindi è stato deciso di introdurre, dopo uno studio delle possibili soluzioni, la Jaro Winkler distance², ossia una metrica che misura la "distanza" tra due stringhe per capire quanto esse siano simili tra loro. Grazie a questa accortezza, nel caso di errore di scrittura, il **chatbot** è in grado di:

²Jaro Winkler distance. URL: https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance.

- * fornire una serie di opzioni di cosa secondo lui l'utente voleva scrivere, dando la possibilità ad esso di selezionare quella giusta;
- * fornire i dati richiesti dall'utente nel caso ci sia un'unica corrispondenza simile a quanto scritto dall'utente nel database.



Figura 4.24: Esempio di utilizzo della Jaro Winkler distance

Capitolo 5

Verifica e validazione

La **verifica** e **validazione** di un prodotto software hanno lo scopo di accertare che esso rispecchi i requisiti e che li rispetti nella maniera dovuta. Con l'attività di **verifica** viene accertato che lo stato di avanzamento del prodotto soddisfi i requisiti precedentemente fissati. Grazie a questa attività è possibile accettare la corretta costruzione del software. L'attività di **validazione** ha invece lo scopo di accettare che il prodotto finale corrisponda alle attese in modo da soddisfare tutti i requisiti prefissati in fase di analisi.

Durante il mio stage lo sviluppo delle nuove funzionalità ha portato alla minimizzazione dei tempi di verifica e validazione. Tale decisione è stata presa in comune accordo con il tutor aziendale, in quanto lo scopo dello stage mirava all'estensione di più funzionalità possibili, che potranno essere testate successivamente dall'azienda. Questa decisione è inoltre appoggiata dalla metodologia *agile* utilizzata nello sviluppo del progetto, la quale definisce la qualità del software come la capacità di soddisfare i bisogni del cliente piuttosto che soddisfare metriche fissata a priori. In ogni caso, vista l'importanza di queste attività, sono state adottate tecniche di analisi statica e analisi dinamica per il codice sorgente, al fine di verificarne e valutarne il comportamento.

- * **analisi statica:** l'analisi statica è il processo di valutazione di un sistema o di un suo componente basato sulla sua forma, struttura, contenuto, documentazione senza che esso sia eseguito. Gli strumenti di analisi statica del codice consentono di individuare porzioni di codice del proprio programma ad alta probabilità di errore. Avendo a disposizione una lista di linee di codice sospette, un programmatore può poi verificare se siano presenti errori e, in caso positivo, rivedere il codice corrispondente e correggere le problematiche individuate;
- * **analisi dinamica:** l'analisi dinamica è il processo di valutazione di un sistema software o di un suo componente basato sulla osservazione del suo comportamento in esecuzione.

5.1 Verifica e validazione dei chatbots

Gran parte del progetto di stage verteva nell'istruire gli *agent* di api.ai per gestire le domande degli utenti fatte attraverso i [chatbot](#). Essendo quindi una parte molto importante del prodotto sono stati creati alcuni test per verificare e validare le sue funzionalità. Questi test non sono stati automatizzati in quanto si è deciso di svolgerli

attraverso due pagine Facebook create dall’azienda dove sono stati associati i due **chatbots** di prova.

Si è quindi pensato insieme al tutor di impostare una serie di domande da porre al bot, cercando di coprire tutti i requisiti che esso dovesse soddisfare.

Grazie a questi controlli è stato possibile accettare che entrambi i **chatbots** soddisfano tutti i requisiti obbligatori e desiderabili esposti nella sezione [3.1](#).

La procedura seguita per verificare e validare il comportamento dei due **chatbots** è stata la seguente:

- * decisione della domanda da sottoporre in base ai requisiti da soddisfare;
- * previsione della risposta che il **chatbot** doveva fornire;
- * scrittura della domanda tramite la chat di Facebook o Messenger;
- * controllo della risposta; nel caso fosse sbagliata:
 - scrittura della domanda nella sezione dedicata di api.ai per verificare che il **JSON** ritornato fosse corretto;
 - correzione della logica di api.ai nel caso fosse sbagliata la risposta;
 - controllo e correzione del codice nel caso il **JSON** fosse corretto.

Questa procedura viene rappresentata nel seguente diagramma di flusso.

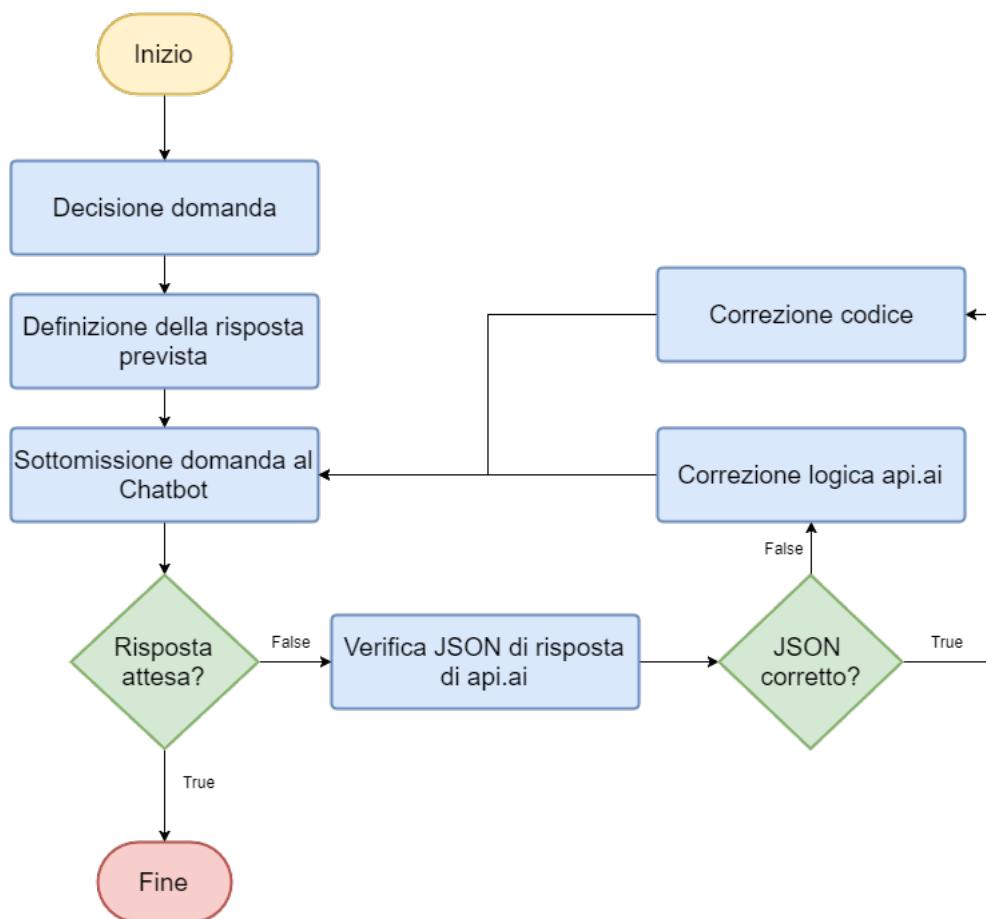


Figura 5.1: Flowchart del processo di verifica dei chatbots

Capitolo 6

Conclusioni

6.1 Raggiungimento degli obiettivi

Come descritto nella sezione 2.4, io e il mio tutor aziendale abbiamo stilato una serie di obiettivi da raggiungere nelle 300 ore di stage, divisi in obiettivi obbligatori e desiderabili.

Gli obiettivi obbligatori si concentravano sull'analisi di mercato dei principali strumenti per il NLP e sullo sviluppo delle funzionalità del prodotto, integrandole nell'architettura già creata dall'azienda. Gli obiettivi desiderabili si focalizzavano invece nel testare in modo approfondito il risultato ottenuto, attraverso l'utilizzo dei [chatbots](#) di Facebook Messenger in un ambiente locale. Al termine dello stage tutti gli obiettivi prefissati sono stati soddisfatti nei tempi previsti. Questo è stato possibile grazie alla buona pianificazione delle tempistiche necessarie allo svolgimento dei vari compiti, sia da parte dello stagista che da parte dell'azienda. La metodologia di sviluppo *Agile* si è inoltre dimostrata molto efficace, permettendomi di rispondere in modo rapido ai cambiamenti proposti dall'azienda durante il mio lavoro.

6.2 Conoscenze acquisite

Da un punto di vista formativo l'attività di stage è stata sicuramente molto positiva. Queste 300 ore mi hanno permesso di mettermi alla prova, dandomi un riscontro su quanto gli anni universitari mi hanno preparato, sia nell'ambito delle mie conoscenze, sia da quello umano, per affrontare il mondo del lavoro.

Lo stage ha sicuramente arricchito il mio bagaglio personale di competenze tecniche, dandomi una panoramica delle procedure e delle attività che giornalmente si svolgono all'interno di un'azienda. Questo mi ha inoltre permesso di rendermi conto quanto una buona collaborazione tra i membri di un team sia fondamentale nello sviluppo di un prodotto, di qualsiasi tipo si tratti. Le competenze acquisite non riguardano solamente il campo informatico, grazie alle nuove tecnologie utilizzate, ma anche le mie capacità organizzative, fondamentali per il rispetto dei vincoli e delle scadenze imposte dall'azienda. Ho dovuto infatti pianificare in modo preciso il lavoro delle mie settimane, per garantire ad i-contact s.r.l. la portata a termine del mio progetto, senza dover impiegare un dipendente per adempire alle mie mancanze.

6.3 Valutazione personale

Nel complesso ritengo la mia esperienza di stage molto positiva ed istruttiva. L'inserimento, se pur breve, in un contesto aziendale permette di far capire ad uno studente quanto siano differenti il mondo del lavoro e quello universitario, in modo da cogliere gli aspetti fondamentali che solamente esperienze di questo tipo ti possono dare.

Per quanto riguarda l'azienda che mi ha ospitato si è rivelata molto disponibile ed accogliente, mi ha guidato nei miei primi giorni di stage chiarendomi tutti i dubbi o le mie mancanze dal punto di vista tecnico. Il progetto che mi è stato proposto mi ha permesso di confrontarmi con nuove tecnologie, mettendo alla prova la mia capacità di apprendere nuovi concetti in breve tempo. Mi sono inoltre reso conto di quanto le mie esperienze universitarie, soprattutto per quanto riguarda i progetti individuali e di gruppo svolti, si sono dimostrate fondamentali per organizzare e gestire una quantità di lavoro che mai avevo dovuto affrontare.

Mi ritengo quindi molto soddisfatto dell'esperienza fatta, in un campo come l'informatica dove apprendere nuove nozioni e competenze resta un aspetto fondamentale per crescere e migliorare.

Glossario

Artificial Intelligence Definizioni specifiche possono essere date focalizzandosi o sui processi interni di ragionamento o sul comportamento esterno del sistema intelligente ed utilizzando come misura di efficacia o la somiglianza con il comportamento umano o con un comportamento ideale, detto razionale:

- * agire umanamente: il risultato dell'operazione compiuta dal sistema intelligente non è distinguibile da quella svolta da un umano.
- * pensare umanamente: il processo che porta il sistema intelligente a risolvere un problema ricalca quello umano. Questo approccio è associato alle scienze cognitive.
- * pensare razionalmente: il processo che porta il sistema intelligente a risolvere un problema è un procedimento formale che si rifà alla logica.
- * agire razionalmente: il processo che porta il sistema intelligente a risolvere il problema è quello che gli permette di ottenere il miglior risultato atteso date le informazioni a disposizione

. 21, 47

Application Program Interface in informatica con il termine *Application Programming Interface* API (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 1, 6, 21, 47

Adaptive Software Development L'*Adaptive Software Development* (ASD) è una metodologia composta da un insieme di regole di sviluppo software inserite in un sistema complessivo detto *Agile Project Management* i cui concetti base sono tre:

- * *Leadership-Collaboration Management* - Uno stile di gestione misto fra gerarchico e collaborativo;
- * *From Processes to Pattern* - Passaggio dall'idea di processo definito e misurabile a quella di processo non perfettamente definito, quasi un processo *fuzzy*;
- * *Peering into the Future* - Osservazione del futuro per capire come l'idea che produrrà un affare di successo debba essere legata al momento in cui diventerà una forma di business.

. 3, 47

Brainstorming L'espressione brainstorming, o brain storming (traducibile in lingua italiana come assalto mentale), è una tecnica creativa di gruppo per far emergere idee volte alla risoluzione di un problema. [11](#), [17](#), [48](#)

Content management system in informatica un *content management system*, in acronimo CMS (sistema di gestione dei contenuti in italiano), è uno strumento software, installato su un server web, il cui compito è facilitare la gestione dei contenuti di siti web, svincolando il *webmaster* da conoscenze tecniche specifiche di programmazione Web. [2](#), [48](#)

Chatbot in informatica con il termine *Chatbot* si indica un programma che simula una conversazione tra un robot e un essere umano. Questi programmi funzionano o come utenti stessi delle *chat* o come persone che rispondono alle FAQ delle persone che accedono al sito. [v](#), [xi](#), [1](#), [7–15](#), [17](#), [18](#), [21](#), [23](#), [24](#), [27–30](#), [34](#), [35](#), [40](#), [41](#), [43–45](#), [48](#)

Diagramma di Gantt il diagramma di Gantt è uno strumento di supporto alla gestione dei progetti, così chiamato in ricordo dell'ingegnere statunitense Henry Laurence Gantt (1861-1919), che si occupava di scienze sociali e che lo ideò nel 1917. Tale diagramma è usato principalmente nelle attività di *project management*, ed è costruito partendo da un asse orizzontale - a rappresentazione dell'arco temporale totale del progetto, suddiviso in fasi incrementali (ad esempio, giorni, settimane, mesi) - e da un asse verticale - a rappresentazione delle mansioni o attività che costituiscono il progetto. [10](#), [48](#)

Extensible Markup Language In informatica XML (sigla di *eXtensible Markup Language*) è un metalinguaggio per la definizione di linguaggi di *markup*, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo. [15](#), [34](#)

Frequently Asked Questions Le *Frequently Asked Questions*, meglio conosciute con la sigla FAQ, sono letteralmente le "domande poste frequentemente"; più esattamente sono una serie di risposte stilate direttamente dall'autore, in risposta alle domande che gli vengono poste, o che ritiene gli verrebbero poste, più frequentemente dagli utilizzatori di un certo servizio: soprattutto su Internet e in particolare nel web e nelle comunità virtuali vi sono domande ricorrenti alle quali si preferisce rispondere pubblicamente con un documento affinché non vengano poste troppo spesso, in modo da sciogliere i dubbi dei nuovi utenti. [9](#), [13](#), [48](#)

File Transfer Protocol in informatica il *File Transfer Protocol* è un protocollo per la trasmissione di dati tra *host* basato su *TCP* e con architettura di tipo client-server. Il protocollo usa connessioni *TCP* distinte per trasferire i dati e per controllare i trasferimenti e richiede autenticazione del client tramite nome utente e password. [1](#), [48](#)

Framework un *framework*, termine della lingua inglese che può essere tradotto come intelaiatura o struttura, in informatica e specificatamente nello sviluppo software, è un'architettura logica di supporto (spesso un'implementazione logica di un particolare *design pattern*) su cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore. [2](#), [48](#)

Frontend termine che denota l'insieme delle applicazioni, relative ad una piattaforma web, con le quali l'utente interagisce direttamente essendo la parte visibile da chiunque e raggiungibile all'indirizzo web del sito.. [2](#), [48](#)

Integrated Development Environment in informatica un ambiente di sviluppo integrato (in lingua inglese *integrated development environment* ovvero IDE, anche integrated design environment o integrated debugging environment, rispettivamente ambiente integrato di progettazione e ambiente integrato di *debugging*) è un software che, in fase di programmazione, aiuta i programmatore nello sviluppo del codice sorgente di un programma. Spesso l'IDE aiuta lo sviluppatore segnalando errori di sintassi del codice direttamente in fase di scrittura, oltre a tutta una serie di strumenti e funzionalità di supporto alla fase di sviluppo e *debugging*. [12](#), [48](#)

JavaScript Object Notation in informatica, nell'ambito della programmazione web, JSON, acronimo di *JavaScript Object Notation*, è un formato adatto all'interscambio di dati fra applicazioni *client-server*. [12](#), [34](#), [41](#), [49](#)

Mercurial Mercurial è un software multipiattaforma di controllo di versione distribuito creato da Matt Mackall e disponibile sotto *GNU General Public License 2.0*. È quasi completamente scritto in *Python*, ma include anche una implementazione *diff* binaria scritta in C. Il programma ha un'interfaccia a riga di comando, ma incorpora anche un'elementare interfaccia web. Tutti i comandi di Mercurial sono invocati come opzioni del programma principale hg, un riferimento al simbolo chimico dell'elemento mercurio. [4](#), [12](#), [14](#), [49](#)

Natural Language Processing l'elaborazione del linguaggio naturale, detta anche NLP (dall'inglese Natural Language Processing, elaborazione lingua naturale), è il processo di trattamento automatico mediante un calcolatore elettronico delle informazioni scritte o parlate in una lingua naturale. Questo processo è reso particolarmente difficile e complesso a causa delle caratteristiche intrinseche di ambiguità del linguaggio umano. Per questo motivo il processo di elaborazione viene suddiviso in fasi diverse, tuttavia simili a quelle che si possono incontrare nel processo di elaborazione di un linguaggio di programmazione:

- * analisi lessicale: scomposizione di un'espressione linguistica in token (in questo caso le parole)
- * analisi grammaticale: associazione delle parti del discorso a ciascuna parola nel testo
- * analisi sintattica: arrangiamento dei token in una struttura sintattica (ad albero: parse tree)
- * analisi semantica: assegnazione di un significato (semantica) alla struttura sintattica e, di conseguenza, all'espressione linguistica

Nell'analisi semantica la procedura automatica che attribuisce all'espressione linguistica un significato tra i diversi possibili è detta disambiguazione. [v](#), [9–11](#), [17](#), [21](#), [25](#), [45](#), [49](#)

Object-relational mapping In informatica l'*Object-Relational Mapping* (ORM) è una tecnica di programmazione che favorisce l'integrazione di sistemi software aderenti al paradigma della programmazione orientata agli oggetti con sistemi RDBMS. Un prodotto ORM fornisce, mediante un'interfaccia orientata agli

oggetti, tutti i servizi inerenti alla persistenza dei dati, astraendo nel contempo le caratteristiche implementative dello specifico RDBMS utilizzato. [2](#), [13](#), [49](#)

Open source In informatica, il termine inglese *open source* (che significa sorgente aperta) indica un software di cui gli autori (più precisamente, i detentori dei diritti) rendono pubblico il codice sorgente, favorendone il libero studio e permettendo a programmatore indipendenti di apportarvi modifiche ed estensioni. Questa possibilità è regolata tramite l'applicazione di apposite licenze d'uso. Il fenomeno ha tratto grande beneficio da Internet, perché esso permette a programmatore distanti di coordinarsi e lavorare allo stesso progetto. [2](#), [13](#), [22](#), [49](#)

POST Il metodo POST è un metodo HTTP usato di norma per inviare informazioni al server (ad esempio i dati di un *form*). In questo caso l'URI indica che cosa si sta inviando e il *body* ne indica il contenuto. [12](#), [13](#), [50](#)

Notifica push La notifica push è una tipologia di messaggistica istantanea con la quale il messaggio perviene al destinatario senza che questo debba effettuare un'operazione di scaricamento (modalità *pull*). Tale modalità è quella tipicamente utilizzata da applicazioni come *WhatsApp*, oppure servizi di sistemi operativi come *Android*, o come numerose applicazioni derivate da siti web (ad esempio il classico servizio meteo o quello delle notizie). Per permettere alle notifiche push di giungere al destinatario è indispensabile che l'applicazione sia attiva, ovvero operi in *background* e sia on-line. Successivamente, occorre che l'utente abbia autorizzato l'applicazione a inviare le notifiche. [1](#), [50](#)

Software Development Kit Un *software development kit* (SDK, traducibile in italiano come "pacchetto di sviluppo per applicazioni"), in informatica, indica genericamente un insieme di strumenti per lo sviluppo e la documentazione di software. Molti SDK sono disponibili gratuitamente e possono essere prelevati direttamente dal sito del produttore: in questo modo si cerca di invogliare i programmatore ad utilizzare un determinato linguaggio o sistema. [9](#), [13](#), [21](#), [22](#), [25](#), [50](#)

Structured Query Language In informatica SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale progettato per:

- * creare e modificare schemi di database (DDL - Data Definition Language);
 - * inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);
 - * interrogare i dati memorizzati (DQL - Data Query Language);
 - * creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language)
- . [13](#), [41](#), [50](#)

Software as a Service *Software as a service* (SaaS) (Software come servizio in italiano) è un modello di distribuzione del *software* applicativo dove un produttore di *software* sviluppa, opera (direttamente o tramite terze parti) e gestisce un'applicazione web che mette a disposizione dei propri clienti via Internet previo abbonamento. Si tratta spesso, ma non sempre, di un servizio di *cloud computing*. [3](#), [50](#)

Stakeholder in economia con il termine *stakeholder* (o portatore di interesse) si indica genericamente un soggetto (o un gruppo di soggetti) influente nei confronti di un'iniziativa economica, che sia un'azienda o un progetto. [3](#), [50](#)

Bibliografia

Siti web consultati

Adaptive software development. URL: https://en.wikipedia.org/wiki/Adaptive_software_development.

Amazon Lex. URL: <https://aws.amazon.com/it/lex/> (cit. a p. 24).

api.ai. URL: <https://api.ai/> (cit. a p. 25).

api.ai documentation. URL: <https://api.ai/docs/getting-started/basics>.

ARPAV Meteo. URL: <http://www.arpa.veneto.it/> (cit. alle pp. 9, 34).

Asana. URL: <https://asana.com/> (cit. a p. 3).

BeeBole. URL: <https://beebole.com/it/> (cit. a p. 3).

BitBucket. URL: <https://bitbucket.org/> (cit. a p. 4).

Botbuilder. URL: <https://www.smshosting.it/it/chatbot/bot-builder-italiano> (cit. a p. 8).

i-contact s.r.l. URL: <http://www.i-contact.it/> (cit. a p. 1).

IBM SDK for Java. URL: <https://github.com/watson-developer-cloud/java-sdk/tree/develop/conversation> (cit. a p. 21).

IBM Watson Conversation. URL: <https://www.ibm.com/watson/services/conversation-2/> (cit. a p. 21).

Jaro Winkler distance. URL: https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance (cit. a p. 41).

Messenger-platform documentation. URL: <https://developers.facebook.com/docs/messenger-platform> (cit. a p. 14).

Meteo Veneto Bot. URL: <https://www.smshosting.it/it/blog/chatbot/il-primo-chatbot-sul-meteo-veneto-e-dolomiti> (cit. a p. 9).

PhoneGap. URL: <https://phonegap.com/> (cit. a p. 2).

Slack. URL: <https://slack.com/> (cit. a p. 5).

SMSHosting.it. URL: <https://www.smshosting.it/> (cit. a p. 1).

wit.ai. URL: <https://wit.ai/> (cit. a p. 22).