Università degli Studi Di Modena e Reggio Emilia

DIPARTIMENTO DI INGEGNERIA "ENZO FERRARI"

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

INSERIRE I RIFUGIATI IN CITTÀ:
Sviluppo di un prototipo per
L'APPRENDIMENTO AUTO-DIREZIONATO

Relatore:

Prof.ssa Sonia Bergamaschi

Candidato:

Marco Orlando

Correlatore:

Prof. Roberto V. Zicari

ANNO ACCADEMICO 2015-2016

Abstract

Inserire i rifugiati in città
Sviluppo di un prototipo per l'apprendimento auto-direzionato

Il problema dell'inserimento dei rifugiati in alcune città è ormai noto. Nella città di Francoforte sul Meno, Frankfurt Big Data Lab e Accenture Foundation, stanno promuovendo il progetto "Integreat" che mira a migliorare la preparazione dei rifugiati, così che essi possano ottenere la conoscenza della lingua e le certificazioni necessarie per potere trovare occupazione in Germania. Integreat ricade fra i progetti per il bene sociale basati sull'analisi dei dati.

In questo lavoro di tesi si è sviluppato un assistente virtuale, *chatbot*, con la funzione di aiutare i rifugiati a scegliere il loro percorso di formazione secondo i principi dell'apprendimento auto-direzionato, SDL (Self-Directed Learning). L'assistente è dotato di un'interfaccia in linguaggio naturale.

Il prototipo realizzato integra un chatbot *Messenger*, necessario a comunicare con i rifugiati in linguaggio naturale, ed un Graph DBMS, *Neo4j*, necessario a memorizzare e collegare le offerte di formazione. Le principali tecnologie utilizzate sono: Facebook Messenger, come piattaforma di messaggistica; Api.ai, come servizio per elaborare il linguaggio naturale; Neo4j come DBMS a grafo; Graphenedb come hosting per Neo4j ed Heroku per distribuire l'applicazione online. Un risultato ulteriore del lavoro di tesi è un approccio metodologico per creare interfacce in linguaggio naturale con il supporto di un DBMS a grafo, applicato al caso di studio dell'inclusione dei rifugiati.

La tesi è stata svolta durante un periodo di studi di 6 mesi in Germania, settembre 2016 – febbraio 2017, nell'ambito del programma europeo ERASMUS+, grazie alla collaborazione tra il gruppo di ricerca, *DBGroup* (UNIMORE), guidato dalla professoressa Sonia Bergamaschi ed il gruppo di ricerca, *Frankfurt Big Data Lab (*Università Goethe di Francoforte sul Meno), guidato dal professore Roberto V. Zicari.

Parole chiave: Apprendimento auto-direzionato, Rifugiati, Graph DBMS, Neo4j, Chatbot, Messenger.

English Abstract

Integrate refugees in cities:
Development of a prototype for Self-Directed Learning SDL

Integrating refugees is a relevant social problem. In Frankfurt am Main, Frankfurt Big Data Lab and Accenture Foundation are promoting "Integreat", a project that aims to help refugees to acquire the necessary language skills and certifications to find a job in Germany. Integreat ranks among projects for social good based on data analysis.

A virtual assistant has been developed in this thesis, chatbot. Its task is to help refugees to choose their learning path according to the principles of Self-Directed Learning, SDL. The assistant is fitted with a natural language interface.

The prototype integrates a chatbot Messenger, needed to communicate with refugees in natural language, and a Graph DBMS, Neo4j, needed to store and link resources for educational and professional training. The main technologies used are: Facebook Messenger, as a messaging platform; Api.ai, as a service to process natural language; Neo4j as a graph DBMS; Graphenedb as hosting for Neo4j and Heroku to deploy the online application. A further result of the thesis is a methodological approach to create natural language interfaces with the support of a DBMS graph, applied to the case history of inclusion of refugees.

The thesis is the result of a six-month period of study in Germany, from September 2016 to February 2017, in the framework of ERASMUS + program, thanks to the collaboration of the research group, DBGroup (UNIMORE), led by Professor Sonia Bergamaschi and the research team, Frankfurt Big Data Lab (Goethe University Frankfurt), led by professor Roberto V. Zicari.

Keywords: Self-directed learning SDL, Refugees, Graph DBMS, Neo4j, Chatbot, Messenger.

Ringraziamenti

Desidero innanzitutto ringraziare di cuore la Prof.ssa Sonia Bergamaschi, relatore di questa tesi, per avermi sostenuto nelle mie idee, sorprendendomi con i suoi insegnamenti didattici e di vita.
Ringrazio il Prof. Roberto V. Zicari, correlatore di questa tesi, per aver reso possibile la mia esperienza Erasmus che ha dato vita a questa tesi.
Ringrazio tutti i membri del Frankfurt Big Data Lab per avermi reso partecipe del loro gruppo, soprattutto Concha Sanchez-Ocaña, per avermi seguito intensamente nella prima fase di ricerca, Kim Hee, per i numerosi piacevoli momenti passati insieme, Todor Ivanov e Karsten Tolle per i loro preziosi consigli, Naveed Mushtaq, per l'aiuto ricevuto nei problemi tecnici riscontrati, Marion Terrell, per il tempo speso per le mie pratiche burocratiche.
Ringrazio tutte le persone che hanno contribuito al progetto Integreat, in particolare Paola Siemion e Harry Underwood di Accenture e Mikel Niño.
Ringrazio Giuseppina Montalto per avermi aiutato in tantissime avventure, dalla scelta del barbiere, ai vari traslochi, dalla compagnia durante la spesa alla rilettura di questa tesi.
Ringrazio Miriam Dönges, per avermi ospitato a casa sua ed insegnato molti giochi matematici.
Ringrazio i miei amici e colleghi informatici, Fabio Lanzi, Federico Bolelli, Francesco Marano, Giordano Cirone, Michele Cancilla, Michele De Giovanni, Michele Mignogna.
Ringrazio i 23 coinquilini con cui ho trascorso i momenti della mia vita universitaria.
Ringrazio gli amici degli Erasmus per tutte le esperienze vissute insieme.
Ringrazio i miei insegnanti di yoga e salsa per essersi presi cura del mio corpo e della mia mente.
Ringrazio tutte le persone che hanno partecipato alla gestione delle pratiche burocratiche, in particolare la Prof. Isabella Lancellotti per quelle Erasmus.
Ringrazio i Professori del Corso di Laurea Triennale e Magistrale in Ingegneria informatica dell'Università di Modena e Reggio Emilia, dell'Università di Lleida (Spagna), dell'Università Goethe (Germania) per aver contribuito egregiamente alla mia formazione accademica.
Ringrazio gli autori di tutte le fonti da cui ho preso spunto.
Ringrazio le persone che, seppur non citate, ci sono sempre state e mi hanno dimostrato il loro affetto.

Desidero infine ringraziare chi sarà contento di questo traguardo e naturalmente la mia famiglia a cui è dedicato.

Indice

Lista delle figure

Lista delle tabelle

Capitolo 1

Introduzione:
Contesto e Motivazione

Questa tesi è stata svolta durante un periodo di studi di 6 mesi in Germania, settembre 2016 – febbraio 2017, nell'ambito del programma europeo ERASMUS+, grazie alla collaborazione tra il gruppo di ricerca, *DBGroup* (UNIMORE), guidato dalla professoressa Sonia Bergamaschi ed il gruppo di ricerca, *Frankfurt Big Data Lab (*Università Goethe di Francoforte sul Meno), guidato dal professore Roberto V. Zicari.
In fondo all'elaborato è inserita una descrizione dell'esperienza vissuta.
La tesi è inserita all'interno del progetto "Bring Inclusion to the City" riguardante l'inserimento dei rifugiati nella città di Francoforte. Questo primo capitolo offre una descrizione generale sul contesto in cui si è svolta la tesi, dalla descrizione del problema alla soluzione proposta. Vengono esposti brevemente i concetti di "Apprendimento Auto-direzionato" e di "Progetti per il Bene Sociale basati sull'analisi dei dati".

1.1 Descrizione del problema

Nel 2015, oltre 1 milione di persone si sono trasferite nell'UE, sia a causa delle guerre nei loro Paesi che per cercare migliori condizioni economiche. Nel 2016 invece i numeri hanno

mostrato una tendenza in diminuzione, prima di giugno 156.000 persone hanno raggiunto l'Europa[1].
Concentrandosi solo sulla città di Francoforte, dal 2015 a marzo 2016, 4.300 rifugiati si sono trasferiti in essa. Si prevede che entro la fine del 2019 un totale di 16.000 rifugiati dovrà essere integrato nella città. I rifugiati hanno diversi background, credenze e valori. Un inserimento veloce nel mercato del lavoro è per loro fra gli obiettivi principali, così che possano sentirsi subito parte attiva della società. Per affrontare questa sfida, il primo passo è aiutare i rifugiati ad acquisire le competenze necessarie per trovare un lavoro.

Dei rifugiati della città di Francoforte, l'80% sono di sesso maschile e circa il 70% sono al di sotto dei 30 anni. Globalmente, il 30% dei rifugiati sono provenienti dall'Afghanistan, il 25% dalla Siria, il 20% dall'Eritrea, il 15% da Iran e Iraq [1]. La quantità di rifugiati che entra nel Paese è leggermente in diminuzione ma la necessità di progetti di integrazione rimane alta. Fra tali progetti, questa tesi si colloca all'interno del progetto "Bringing Inclusion to the City[2]", anche detto "Integreat", promosso dal Frankfurt Big Data Lab[3] dell'Università Goethe di Francoforte in collaborazione con Accenture Foundation[4].

1.2 Il progetto Integreat

Il progetto Integreat è principalmente dedicato alla preparazione dei rifugiati, necessaria per ricevere certificazioni per lavorare in Germania. Al momento, si concentra solo sui rifugiati, singole persone, famiglie, bambini, che vivono già o arriveranno nella città di Francoforte. La maggior parte dei rifugiati ha un basso livello di istruzione ed una capacità di comunicare in inglese limitata, ma sono molto motivati e dediti all'apprendimento. Gli stessi sono dotati di smartphone che utilizzano per organizzare riunioni e scambi attraverso i social.
Il problema principale per l'integrazione è la fiducia da parte dei rifugiati. Molti di essi hanno affrontato molte difficoltà durante il loro percorso e ora il livello di diffidenza è alto, spesso si rifiutano di fasi aiutare, oltre ad avere una cattiva percezione della società. Per

[1] http://ec.europa.eu/echo/refugee-crisis_en
[2] http://www.bigdata.uni-frankfurt.de/data-refugees-project/
[3] http://www.bigdata.uni-frankfurt.de/
[4] https://www.accenture.com/de-de/

ottenerla, sarà necessario mostrare ai rifugiati esempi chiari dei benefici e gli effetti positivi che il progetto potrà portare a tutti i soggetti coinvolti, analizzati in seguito[1].

Il progetto si propone di facilitare l'inserimento dei rifugiati nella città di Francoforte, offrendo le basi e le implementazioni per vari servizi e diverse opportunità, con:

- Supporto all'occupazione e all'apprendimento;
- Creazione di nuove opportunità di sostegno a quelle già esistenti;
- Creazione di fiducia reciproca fra i cittadini e i rifugiati;
- Creazione di un gruppo di "rifugiati giovani sociali" che, alla fine della loro esperienza di inclusione, continueranno ad interagire la comunità autonomamente;
- Creazione di risorse per i media, il monitoraggio della percezione del pubblico e la comunicazione con la città vicine;
- Educazione dei Policymaker;
- Educazione dei rifugiati verso i loro diritti ed obblighi.

Le principali sfide per la città sono:

1. Un inserimento veloce dei rifugiati nel mercato del lavoro per offrirgli una prospettiva di vita reale e farli sentire parte attiva della società. Questo aspetto risulta fondamentale per la società, in quanto mette in luce sia gli effetti positivi per la comunità, che la volontà di integrazione da parte dei rifugiati.
2. Un processo di inclusione efficace nel medio-lungo termine per i rifugiati nella città.
3. Un processo di inclusione misurabile e sostenibile.

I rifugiati necessitano di buone competenze linguistiche in tedesco, al fine di integrarsi nel mercato del lavoro (attualmente nella maggior parte dei casi è richiesto almeno il livello B2). Molti rifugiati hanno anche un'istruzione limitata in matematica e scienze. Di conseguenza, sono necessari nuovi metodi che permettono un inserimento anteriore nel mercato del lavoro. Ciò richiede degli adeguamenti da parte delle industrie ed una formazione al di fuori del lavoro. L'inclusione richiede che la popolazione di ricezione sia aperta ad accettare i rifugiati. Vari cittadini hanno paura. È quindi importante una comunicazione trasparente fra gli attori (rifugiati, cittadini, governo ...). Qui a Francoforte la maggior parte dei cittadini, ancora oggi, non sa cosa succede esattamente.

Le strutture convenzionali e le risorse per la formazione scolastica e professionale sono inadeguate e non sufficienti per far fronte alla sfida di medio e lungo termine, a causa di una popolazione di rifugiati molto disomogenea all'interno del tessuto cittadino.

La soluzione proposta dal progetto si suddivide in due aspetti principali:

1. La definizione della **Città come una comunità di apprendimento**, per fornire un nuovo, innovativo ed efficace modo per sostenere l'apprendimento auto-direzionato.[2] Verrà istituito un centro per l'apprendimento auto direzionato (SDL) che comprende l'intero spettro di offerte nell'ambito dell'istruzione, formazione, lavoro ed offerte culturali. Il centro si occuperà della creazione di unità SDL accreditate, creerà e amministrerà i percorsi di formazione di SDL nella città a seconda delle necessità (ad esempio per i giovani, per le giovani ragazze, per le famiglie con i bambini, tenendo conto delle variabili quali la regione di origine, background, stato, ecc.). I rifugiati saranno in grado di scegliere uno o più percorsi di formazione. Ogni percorso sarà composto da passi. Ogni passo corrisponderà ad una esperienza di apprendimento svolto in città, ad esempio un laboratorio pratico. Il superamento di ogni fase sarà gestito dal centro. Al completamento di un intero percorso, sarà fornito un certificato. L'idea è di ridefinire la città come un laboratorio dinamico per SDL affiancato sia dal tutoraggio che dall'accreditamento.

2. La creazione di un **Data Forum per i cittadini**, attraverso l'utilizzo dei dati, per fornire una nuova piattaforma che sviluppi le migliori pratiche, sia per l'apprendimento auto-direzionato che per la creazione di nuove, originali ed innovative modalità di inclusione, coinvolgendo nel processo tutta la città.
   Tenendo conto delle esigenze, utilizzando una combinazione di tecniche di analisi dei dati, come ad esempio il Machine Learning, e le attività di coordinamento convenzionali, il Data Forum per i cittadini (DF) gestirà la creazione di percorsi auto-direzionati, la lista degli enti accreditati che offrono le esperienze nei vari percorsi, il monitoraggio ed il sostegno al processo di certificazione.

Il progetto è molto innovativo nel contesto urbano, in quanto mira a:

- Migliorare il tessuto urbano con i dati
- Mantenere alta l'attenzione sulle persone
- Cambiare l'esperienza dei cittadini
- Creare delle comunità
- Affrontare una sfida specifica: Inclusione dei rifugiati
- Lavorare con e non contro le istituzioni
- Creare opportunità impreviste e nuove relazioni

Invece di avere un certo numero di iniziative non relazionate fra loro e sovrapposte, non coordinate, come avviene adesso, l'introduzione della città come una comunità di apprendimento e l'introduzione del DATA Forum per i cittadini, avrà lo scopo di cambiare completamente il modo in cui i servizi per l'inclusione sono gestiti e creati nella città. Ciò contribuirà non solo ad includere i flussi recenti di rifugiati, ma soprattutto a gettare le fondamenta per un'infrastruttura sostenibile e scalabile per affrontare le sfide future.

La soluzione presentata integra perfettamente due dimensioni:

- "apprendere competenze per avere successo"
- "la fiducia attraverso la trasparenza".

Tutto ciò attraverso una piattaforma digitale, aperta per permettere l'aumento della fiducia, e con meccanismi di feedback di una comunità on-line per fornire un'ottimizzazione efficace dei percorsi SDL.
La piattaforma non fornisce attivamente i contenuti di apprendimento, ma è "solo" un incentivo verso i fornitori di contenuti di apprendimento a contribuire. Ciò la rende particolarmente scalabile (Es. Estendere la portata del progetto per altre regioni della Germania, altri paesi in Europa, ecc.).

1.3 Progetti per il bene sociale basati sull'analisi dei dati

Il progetto Integreat è un caso di studio del "Data Projects for Social Good: Challenges and Opportunities"[1], un documento che presenta un framework per analizzare i progetti per il bene sociale basati sull'analisi dei dati. L'obiettivo di tali progetti è migliorare l'efficacia e l'efficienza delle iniziative sociali, aiutando l'umanità in generale, ed in particolare le

persone bisognose. Ricadono in questi progetti tematiche come le crisi umanitarie, l'assistenza sanitaria, i problemi ambientali, ecc.

Ci sono numerose sfide in questi progetti normalmente non presenti in progetti "classici". Queste sfide derivano da vari aspetti, come, la portata e le dimensioni della questione sociale da risolvere, le barriere culturali e politiche, le risorse tecnologiche disponibili, la motivazione ad essere impegnati in questo tipo di progetti, le questioni etiche e giuridiche connesse ai dati sensibili, le competenze dei principali soggetti interessati.

Fra gli strumenti presentati nel framework di notevole importanza è l'analisi degli stakeholder, ovvero i portatori di interessi. Essi vengono suddivisi in "problem holders", "data holders" e "skill holders".

- **Problem holders** o titolari del problema, sono i soggetti più vicini alla questione sociale da affrontare nel progetto. Spesso questi soggetti sono i rappresentanti della pubblica amministrazione. Essi forniscono conoscenze fondamentali in merito alla questione e importanti obiettivi da raggiungere nell'ambito del progetto. Il loro profilo professionale è molto qualificato sulle questioni sociologiche, ma non tanto sulla tecnologia da applicare e il suo potenziale in questi contesti.

- **Data holders** o titolari dei dati, sono i soggetti più vicini all'acquisizione, elaborazione e analisi dei dati del progetto. In alcuni casi possono coincidere con i titolari del problema.

- **Skill holders** o titolari delle abilità. sono necessari per risolvere determinati problemi avendo la capacità di utilizzare strumenti tecnologici fondamentali per portare a termine determinati compiti.

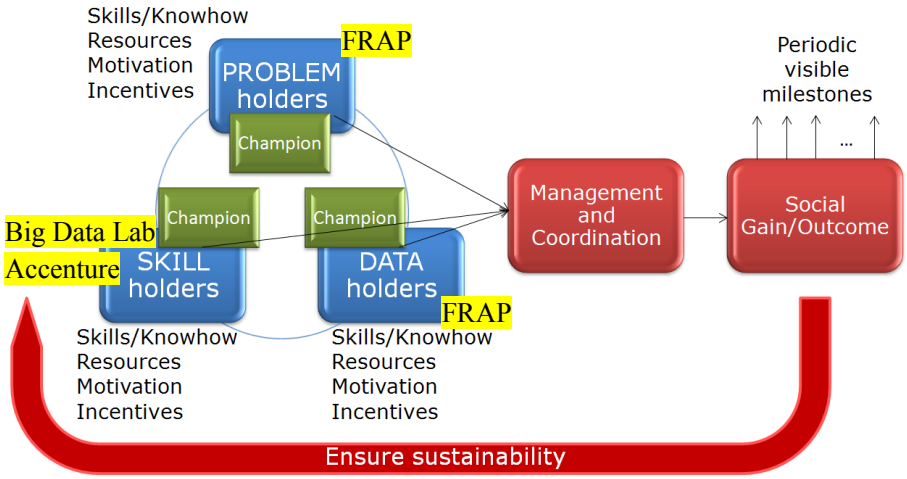


Figura 1 I soggetti interessati in progetti per il bene sociale basati sull'analisi dei dati [1].

1.4 I soggetti coinvolti nel progetto

Agenzia FRAP

L'Agenzia FRAP (FRAP Agentur[5]) è una società dipendente dal consiglio comunale della città di Francoforte. Lo scopo della FRAP è il coordinamento e lo sviluppo della promozione per l'occupazione locale. Essa offre consulenze individuali legate al lavoro e alla formazione continua all'interno di un centro di consulenza per il mercato del lavoro di Francoforte. In particolare i soggetti FRAP contribuiscono al progetto "Frankfurt Helps",[6] che mira a coordinare le attività per l'inclusione dei rifugiati nella città di Francoforte. Essi interagiscono, da un lato, con i fornitori dei servizi di formazione e con i potenziali datori di lavoro, dall'altro lato con i rifugiati, valutando le loro competenze.

Questa valutazione si basa sul tempo trascorso a scuola e sulla conoscenza del sistema scolastico del Paese di origine. Ciò implica la necessità di una precedente analisi del contesto socioeconomico e culturale di quei Paesi. L'interazione con i rifugiati avviene grazie all'aiuto di volontari che lavorano come traduttori.

[5] http://frap-agentur.de/
[6] http://frankfurt-hilft.de/

Accenture Foundation

La fondazione Accenture è esperta nelle collaborazioni con le organizzazioni non governative per realizzare progetti. I loro volontari, che provengono da diverse nazioni, sono esperti nella creazione di progetti ed attività di supporto. In particolar modo essi si occupano della formazione linguistica e culturale dei rifugiati.

Frankfurt Big Data Lab (from Goethe Frankfurt University)

Il Big Data Lab ha competenze altamente qualificate in attività interdisciplinari sulle tecnologie digitali, l'economia e le aree sociali.

1.5 Soluzione proposta

La soluzione proposta da questa tesi per migliorare l'integrazione dei rifugiati è un *chatbot*[7], un assistente virtuale con la funzione principale di aiutare i rifugiati nella scelta del loro percorso di formazione, mediante i principi dell'apprendimento auto-direzionato. La formazione è intesa nel senso più generale, quindi non solo veri e propri corsi, ma anche eventi, gite turistiche, tirocini o qualsiasi attività utile ai rifugiati. La soluzione, all'interno del progetto Integreat, si colloca sia nella definizione della Città come una comunità di apprendimento, per organizzare le varie offerte di formazione che nella creazione del Data Forum per i cittadini dove gestirà la creazione di percorsi auto-direzionati.
Un prototipo di chatbot, che chiameremo *Ibot* (Integreat bot), è stato sviluppato ed è disponibile sia per smartphone che per computer, essendo basato sul sistema di messaggistica di Facebook. Esso può dialogare quasi come una vera persona e suggerire i percorsi di formazione più adatti in base ai dati forniti dall'utente o da enti esterni.
Un DBMS a grafo, implementato con Neo4j[8], si occupa della generazione dinamica di tali percorsi di formazione. In questa tesi verranno analizzate tutte le fasi di sviluppo di Ibot, come mostrate in Figura 3.

---

[7] Il termine chatbot deriva dall'unione di due parole inglesi "chat" e "robot". La definizione fornita da Wikipedia Italia afferma: *"I chat bot, o chatbot, sono dei programmi che simulano una conversazione tra robot e essere umano.*[3]*"*

[8] https://neo4j.com/

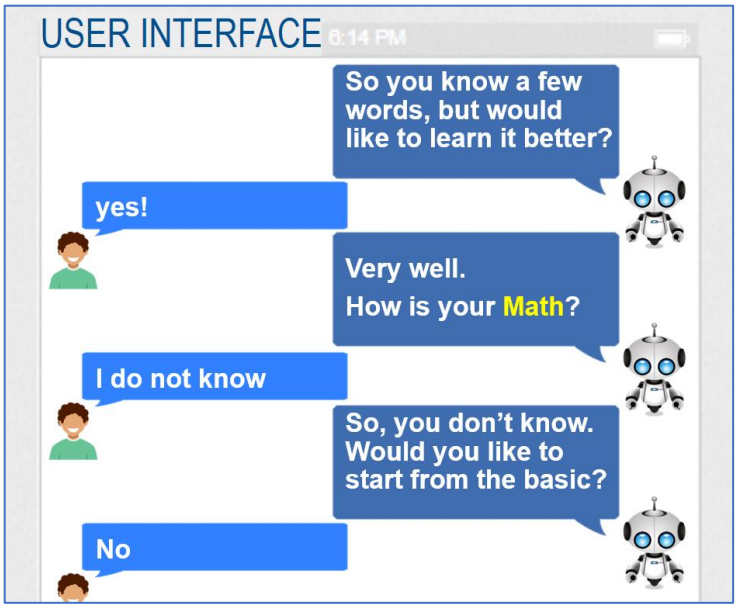


Figura 2 Esempio di conversazione del chatbot sviluppato.

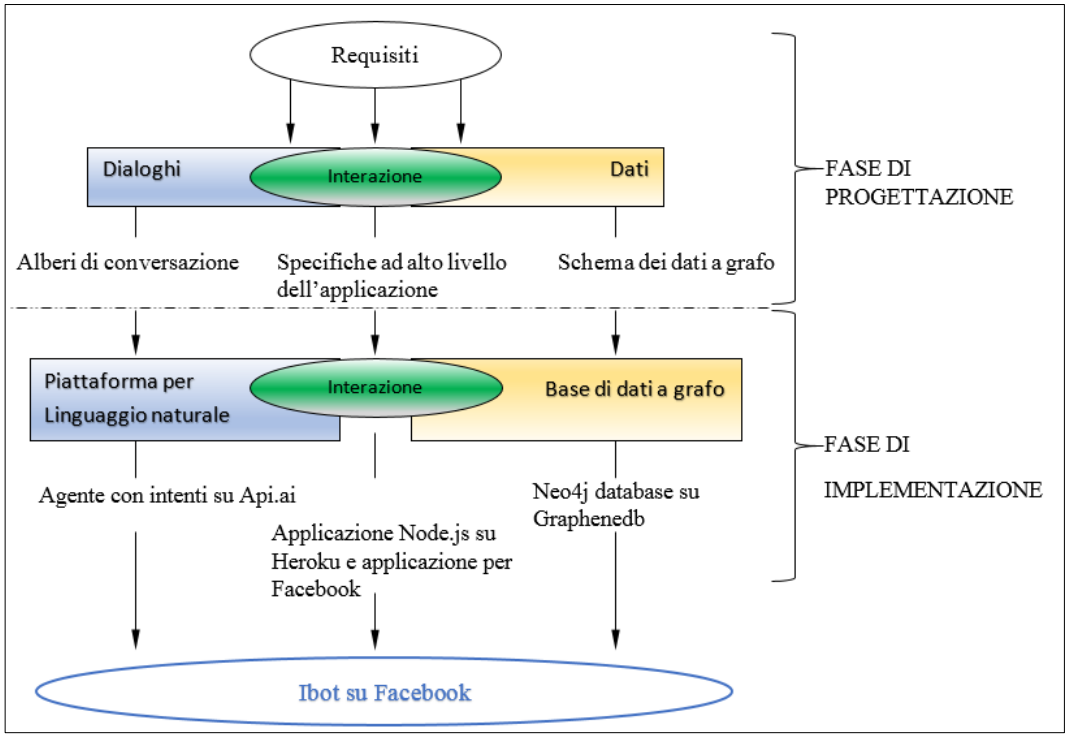


Figura 3 Fasi della realizzazione del chatbot

1.6 Apprendimento Auto-direzionato

L'apprendimento auto-direzionato, o in versione inglese Self-Directed Learning, è una metodologia per migliorare la formazione in termini professionali sia per chi insegna che per chi apprende[2]. Il self-directed learning, o SDL, è un approccio già molto diffuso negli Stati Uniti utile a formatori, docenti e studenti. Il suo obiettivo è rendere gli studenti e gli adulti, che partecipano ai corsi di formazione (detti learner), in grado di decidere quale sia il percorso migliore, personalizzato in base alle loro reali esigenze e bisogni. Ad esempio, in un corso universitario, dare la libertà agli studenti di scegliere quali temi di approfondimento sono più significativi per loro.
In Italia questo tema viene ampiamente descritto dal libro di Malcom S.Knowels [2]. Il metodo è rivolto principalmente allo sviluppo della formazione per l'apprendimento degli adulti. Disciplina davvero importante al giorno d'oggi, in un mondo pieno di cambiamenti. Una grande novità a tal proposito è in campo universitario: vi è la comparsa dell'indirizzo di laurea in psicologia del lavoro e dell'organizzazione, e il progetto dell'indirizzo di laurea in scienza dell'educazione degli adulti. Ciò mostra che questa esigenza vien riconosciuta anche dalla società e dal mondo del lavoro.

L'utilizzo di un percorso di formazione auto-direzionato, stabilito fra uno studente ed un formatore, anche detto contratto di apprendimento o Learning Contract (LC), in realtà, ha anche altri aspetti positivi: studenti motivati, coinvolti nel percorso di apprendimento, focalizzati sull'obbiettivo, responsabili delle loro scelte[2].
Il processo di apprendimento nel testo di riferimento viene diviso in 7 step, ciascuno con un bisogno da soddisfare associato:

1) **Creare il clima** - sapere perché si apprende
2) **Pianificare** - essere auto-direzionati
3) **Diagnosticare i propri bisogni di apprendimento** - considerare l'esperienza personale nel proprio processo di apprendimento
4) **Stabilire gli obbiettivi del proprio apprendimento** - acquisire la disponibilità del learner ad apprendere.

5) **Definire un progetto di apprendimento personale** - organizzare l'apprendimento sui compiti e problemi reali della vita.

6) **Intraprendere le attività di apprendimento** - essere spinti da una forte motivazione ad apprendere

7) **Valutare i risultati raggiunti**

Gli step 1, 3, 4, 5 sono la parte essenziale per il nostro progetto. Gli step 1 e 3 vengono realizzati mediante una conversazione con il nostro Ibot. Gli step 4 e 5, mediante scelte sulle opzioni offerte da Ibot, del tipo: "Esiste un corso di lingua tedesca certificato presso l'Università Goethe".

Capitolo 2

Tecnologie e Metodologie Utilizzate

Questo capitolo descrive alcune delle tecnologie e delle metodologie inerenti al progetto. Molte delle seguenti informazioni sono disponibili in rete, ma vengono riproposte in forma rielaborata per essere di aiuto alla comprensione del progetto, o per chi vuole sviluppare un progetto simile. Vengono descritti i tipi di architettura classici alla base dei bot e le principali scelte implementative:

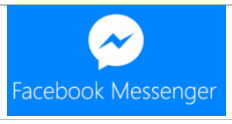
Facebook Messenger Piattaforma di messaggistica [4]	
Api.ai Servizio per elaborare il linguaggio naturale [5]	
Neo4j DBMS a grafo [6]	
Graphenedb Hosting per Neo4j [7]	
Heroku PaaS per distribuire l'applicazione online [8]	

Tabella 1 Principali tecnologie utilizzate.

2.1 Cosa sono i chatbot

Una definizione più dettagliata e moderna di chatbot, dal blog di semrush.com[9], è la seguente:

"Un chatbot è un servizio, caratterizzato da regole, schemi e (a volte) intelligenza artificiale, che permette agli utenti di interagire via chat, senza l'intervento umano. Può riguardare una serie infinita di argomenti, impieghi e applicazioni, da quelli più funzionali a quelli per mero divertimento ed è possibile trovarli quasi in ogni servizio di chat (Facebook Messenger, Slack, Telegram, Sms, etc.)[9]*"*

La letteratura riguardante i chatbot in generale è molto ampia e le sue radici storiche sono molto antiche (Nel 1950 Alan Turing pubblicò il suo famoso articolo Computing Machinery and Intelligence, in cui ha proposto un criterio in grado di determinare se una macchina è in grado di pensare o meno[10]). Tuttavia, questo elaborato tratterà solo ed esclusivamente i chatbot di Facebook, seppure molti concetti possono essere estesi ad altri tipi di chat.

I chatbot negli ultimi anni sono in rapidissima evoluzione per cui molte funzionalità usate durante lo sviluppo di questa tesi potrebbero essere non più valide già prima della pubblicazione della tesi stessa. I dettagli tecnici verranno principalmente descritti ad alto livello e si rimanda alle documentazioni ufficiali delle varie tecnologie per approfondimenti.

2.2 Architettura generale di un chatbot

Alla base di un chatbot spesso vi sono numerose discipline. Esso utilizza tecniche di: Data Mining, Machine Learning, Natural Language Processing (NLP), Web Database, ecc. Molte di queste tecniche oggi vengono inglobate in black box, "Bot engine", i quali espongono interfacce molto semplici. Una semplice architettura di un chatbot è formata da due elementi: un bot engine ed una chat platform[11].

[9] https://it.semrush.com/blog/chatbot-cosa-sono

Figura 4 Architettura semplice chatbot.

Questa architettura è la migliore per creare dei bot molto semplici in pochissimo tempo. Molto importante è la possibilità di essere implementata senza scrivere alcuna riga di codice. Questo, grazie alle nuovissime interfacce visuali dei bot engine. Può essere utilizzata per numerosi tipi di bot ed è particolarmente veloce nell'elaborare le risposte.

Se da un lato è molto semplice, dall'altro è limitata. Spesso vengono a mancare piccoli elementi specifici delle Chat Platform, come ad esempio dei tasti di risposta multipla dipendenti dalla piattaforma di messaggistica e sono necessari dei "workaround" complessi. Inoltre, non è possibile interagire in modo generale con altri servizi o creare funzioni ad hoc. Ad esempio, non sempre possono essere collegati dei database esterni. Proprio per tali limiti, il modello tipico di architettura è quello a tre livelli, un bot engine, una chat-platform, come nel caso precedente, ed una applicazione web che si interfaccia fra essi.

Come è possibile notare dall'immagine seguente (Figura 5), nell'architettura a 3 livelli l'utente invia un messaggio al bot tramite la chat platform, come ad esempio Messenger, Whatsapp, Telegram. Tale messaggio sarà inviato all'applicazione web, dove sarà validato. Se risulta valido sarà inviato al bot engine, che procederà utilizzando tecniche di machine learning e NLP per elaborare una risposta. La risposta sarà successivamente inviata alla webapp, dove verrà validata e sarà svolta la logica dell'app, come interazioni con il database o con altri servi

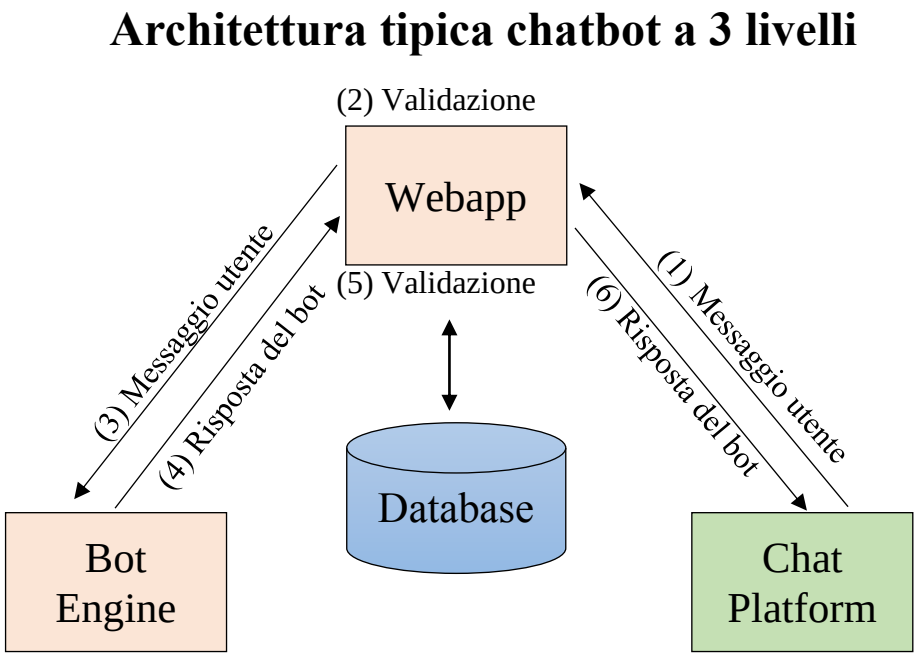


Figura 5 Architettura chatbot tipica a 3 livelli.

2.3 Chatbot su Facebook

Facebook Messenger è la Chat Platform utilizzata. Il 12 Aprile 2016, Facebook ha svolto la conferenza annuale dei suoi sviluppatori, l'evento F8, dove vengono presentati gli ultimi prodotti e la loro visione del futuro. Fra le varie novità lanciate, vi è stata "Messenger Platform", che estende le funzionalità su Messenger. [12]

Messenger, la seconda applicazione mobile più popolare su IOS a livello globale nel 2015 secondo nielsen.com, subito dopo quella di Facebook, ora disponibile anche in versione di applicazione web su *www.messenger.com* o semplicemente su www.m.me , ha introdotto i "chatbot" o semplicemente i "bot".

Messenger ha delle proprietà molto interessanti:

- È istantanea
- Mantiene le identità di tutti i partecipanti in ogni momento
- Non è mai fuori contesto
- Le conversazioni sono sempre canoniche

2.3.1 Esempio di bot Messenger

Il primo esempio di bot presentato durante il lancio della piattaforma è quello del CNN10. Il bot CNN invia su richiesta o regolarmente, nuovi articoli personalizzati in base ai gusti degli utenti. È possibile ricevere riassunti o cercare articoli semplicemente scrivendo un messaggio o registrando un audio-messaggio.

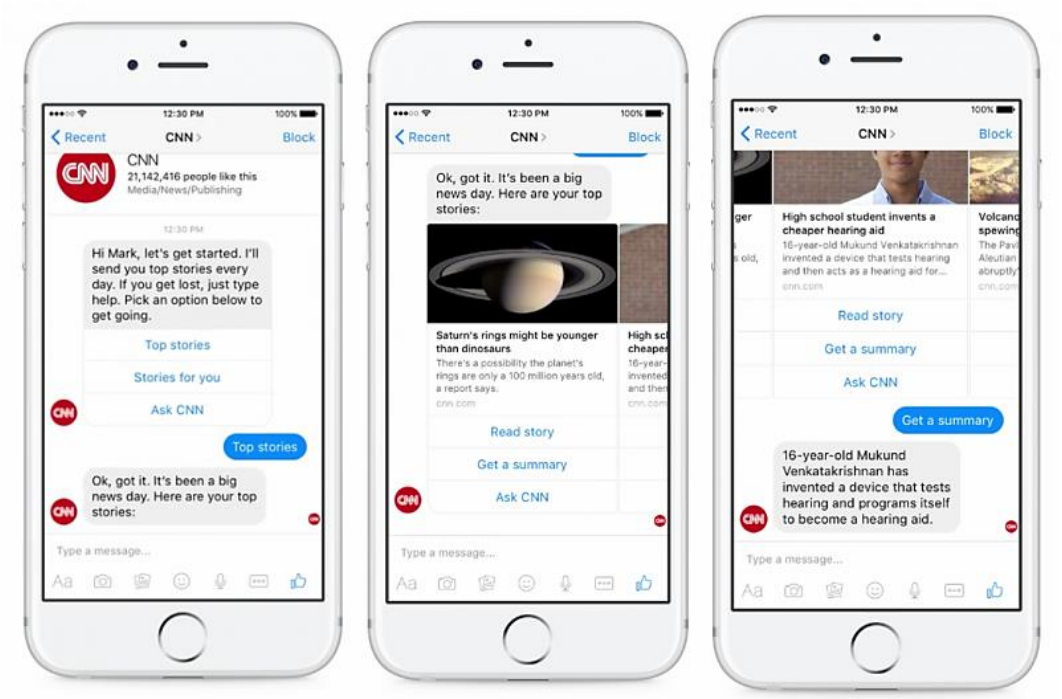


Figura 6 Screenshot chatbot CNN.

2.3.2 Estratti sul Messenger Platform dal F8 2016

Nel 2015 Messenger è stata l'app con il più alto tasso di crescita negli Stati Uniti, ed oggi fra Messenger e Whatsapp, le persone inviano circa 60 miliardi di messaggi al giorno. Gli sms all'apice della loro diffusione arrivavano a un massimo di 20 miliardi al giorno. Ora che l'app Messenger è così diffusa, il team di Facebook ha iniziato a sviluppare l'ecosistema intorno a essa. La prima cosa che hanno valutato è stata la comunicazione con

[10] Emittente televisiva statunitense

le aziende. Oggi tutti noi abbiamo a che fare regolarmente con delle aziende, ma nessuno vuole istallare una nuova app per ogni servizio da loro offerto o semplicemente interagire con esse. Messenger vuole dare la possibilità d'inviare messaggi alle aziende nello stesso modo in cui si manda un messaggio ad un amico. Si dovrebbe ottenere una risposta rapida e non impegnativa come una telefonata e senza il bisogno d'installare una nuova app.
Messenger Platform permette di creare bot per Messenger. Si tratta di una piattaforma semplice e con la possibilità di usare l'intelligenza artificiale, con cui creare servizi per comunicare direttamente con le persone in linguaggio naturale.

Caratteristiche principali dei bot:

- È possibile scoprire l'esistenza di un bot e accedervi in vari modi
- Tutti i bot hanno una propria identità, la stessa della pagina Facebook[11] associata al bot.
- I bot possono inviare e ricevere, testi, immagini, video, template, bottoni e inviti all'azione.

In futuro, le aziende potranno inviare messaggi sponsorizzati grazie ai bot.
Proprio per questo, sono stati creati dei controlli utente per bloccare messaggi in ingresso. Come mostrato in Figura 6, è presente un bottone "Block" che permette di bloccare determinati tipi di messaggi o tutti quelli provenienti dal bot. Si noti che il 13/01/2017 ho verificato che non era più presente il bottone "Block", ma, per interrompere il flusso di messaggi era necessario qualche passaggio in più. Ciò è solo un piccolo esempio che mostra la rapida evoluzione di questa piattaforma.

Numerose aziende utilizzano già gli sms per le comunicazioni con i loro clienti, per tale motivo è stato creato uno strumento in grado di associare i numeri di telefono dei clienti (che hanno accettato esplicitamente) agli account Messenger, in modo da poterli contattare su Messenger anziché inviargli sms. Ciò è possibile mediante lo sviluppo di codice proprio oppure collaborando con dei partner aziendali, primo fra tutti Twilio.[12]

[11] Nel seguito abbreviato con fb
[12] https://www.twilio.com/

L'ultima novità presentata durante l'evento, ma non ancora disponibile, è la possibilità di acquistare inserzioni sulle Notizie di Facebook che reindirizzeranno le persone direttamente al bot relativo su Messenger.

Molto importante è anche "M", l'assistente digitale di Messenger, che apprende grazie all'unione fra intelligenza artificiale e istruttori umani. Ciò che gli sviluppatori di Messenger hanno capito è che bisogna creare tanti piccoli diversi bot verticali per riuscire a comprendere realmente le interazioni delle persone. Pertanto, è stato sviluppato un Bot Engine chiamato *Wit.ai*, destinato inizialmente ad uso interno, ma adesso disponibile a tutti. Ciò permette di creare bot high-end, in grado di conversare e di apprendere da soli.
In Figura 7 è presentato un semplice esempio del funzionamento di Wit.ai in cui sono evidenziati i parametri chiave necessari a formulare la richiesta delle previsioni del tempo (when e where).

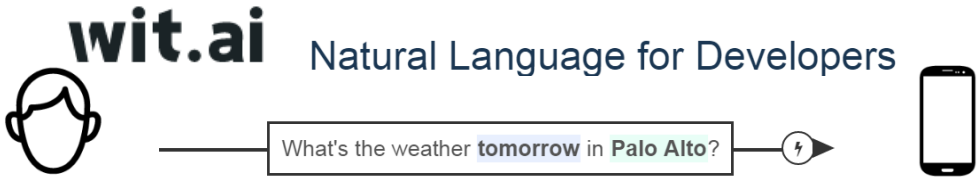


Figura 7 Bot engine wit.ai.

Per usare il Bot Engine è necessario caricare alcune semplici conversazioni su di esso e lui gestirà automaticamente le loro variazioni. Grazie alle tecniche di apprendimento automatico, esso migliorerà sempre di più nel tempo. Inoltre, la sua interfaccia è tale da permettere di non scrivere alcun codice software.

2.3.3 Costruire un Messenger bot: meccanismi e funzionalità

Durante l'F8 è stato presentato anche un *live coding* (scrittura di codice sorgente dal vivo) per costruire il proprio bot [13]. Le caratteristiche funzionali sono qui elencate:

- Il bot è multipiattaforma, basta scrivere una sola volta il codice per potere utilizzare il bot su tutte le principali piattaforme, come Android, IOS, Windows, Linux, ecc.

- L'interfaccia è basata sulle conversazioni, gli utenti Messenger sono già abituati a usarla
- Il logo del bot viene automaticamente prelevato dalla pagina facebook (fb) associata ed è possibile aggiungere una descrizione
- Per iniziare a conversare con un bot basta premere GET STARTED sulla sua pagina di presentazione
- Non è necessario imparare nessun linguaggio di programmazione per interagire con il bot. Molte interazioni avvengono principalmente tramite bottoni a cui sono associate parole, notevolmente più veloci rispetto a scrivere ogni volta del testo

La documentazione ufficiale per creare un Bot è molto chiara per cui di seguito verrà riportata solo una breve sintesi concettuale degli step necessari a costruire un bot:

1) **Generare il proprio "Page Access Token"**. Esso rappresenta una connessione fra la pagina fb personale e fb. La fb app è il connettore per Messenger Platform. È possibile utilizzare una fb app già esistente o crearne una nuova. La connessione fra fb app e fb page è di tipo molti a molti. Ci possono essere più fb app appartenenti alla stessa fb page oppure una fb app che gestisce più fb page.

Figura 8 Esempio di Page Access Token.

È possibile inviare messaggi all'utente attraverso la send/receive API[13]. Come mostrato nella Figura 9 ci sono due end point, Webhooks, per ricevere i messaggi inviati dall'utente al nostro bot, e Graph API per inviare messaggi. Entrambi utilizzano il token generato e l'user id. In Messenger platform ogni user id è page

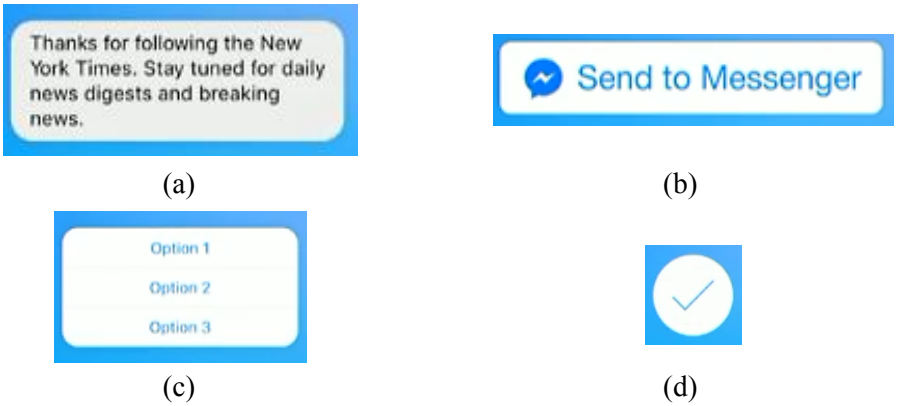
[13] https://developers.facebook.com/docs/messenger-platform/send-api-reference

scope, ciò significa, per pagine differenti, l'user id sarà differente per la stessa persona.[14]

Figura 9 Flusso messaggi fra un bot e l'API di Messenger.

2) **Impostare il "Webhook".** Esso viene utilizzato per ricevere dei dati, come il contenuto dei messaggi in base al verificarsi di determinati eventi selezionabili. Per iniziare a ricevere eventi è necessario registrare il Webhook con il token page generato precedentemente. Per ogni evento che raggiunge il Webhooks si ottiene:
   - Page scoped user-id (PSID)
   - Dati dell'evento
   - Tipo
   - Eventuale payload

Esistono quattro principali tipi di eventi come in figura



(a) (b)

(c) (d)

Figura 10 Principali tipi di eventi.

[14] https://developers.facebook.com/docs/messenger-platform/webhook-reference

- [message] per i semplici messaggi testuali (a)
- [messaging-options] quando gli utenti cliccano i plug-in sul web (b)
- [messaging-postbacks] per i bottoni (c)
- [message-deliveries] per gestire le conferme delle consegne (d)

È necessario generare il page token solo una volta per ogni pagina ed impostare il webhook solo una volta per ogni bot che viene creato.

3) **Inviare messaggi.** Per far ciò è stato creato un nuovo Graph End Point.

(a) (b) (c)

Figura 11 Contenuti multimediali supportati dai bot.

Esso supporta diversi contenuti multimediali:

- Text (a)
- Images (b)
- Templates strutturati o modelli (c)

Figura 12 Esempi di modello con pulsante, generico e di ricevuta.

Sui Template di Figura 13 ci sono caratteristiche interessanti come H-scroll e Postbacks. Il primo, detto anche carosello, permette di inviare più card all'interno di un singolo messaggio. Ciò aumenta di molto la quantità di informazioni che è possibile mostrare insieme. È possibile sfogliare le card semplicemente con il semplice swipe orizzontale. Il secondo permette ai bottoni, nominati precedentemente, di inviare messaggi da parte

dell'utente al bot con un semplice click. Questo è un ottimo modo per ricevere messaggi standard.

Figura 13 Esempio di carosello con 4 card uguali.

Seppur ancor limitata, vi è la possibilità di inviare messaggi direttamente al numero di telefono grazie alla "Customer Matching"

4) **Farsi scoprire**. Per far sì che i bot siano facili da trovare e accessibili senza problemi ci sono varie soluzioni:
   - Plug-in di Messenger da inserire nei propri siti web, app, email, ecc.
     Esso invierà eventi al nostro sistema ogni volta che verrà cliccato. Questo è un ottimo modo per collegare gli utenti dei vostri siti internet al vostro bot.

Figura 14 Esempi di punti di accesso al bot.

   - Codici, nomi utenti e link. È stato creato un nuovo codice simile al QR code per permettere all'arrivo sul bot anche dal mondo offline con una semplice fotografia.

Figura 15 Esempio di Fb code e di link diretto al bot.

- La barra di ricerca rimane sempre fissa, e con un semplice click permette la visione dei numerosi bot già funzionanti. Inoltre è possibile ricercare qualsiasi pagina, usando solo il suo nome.

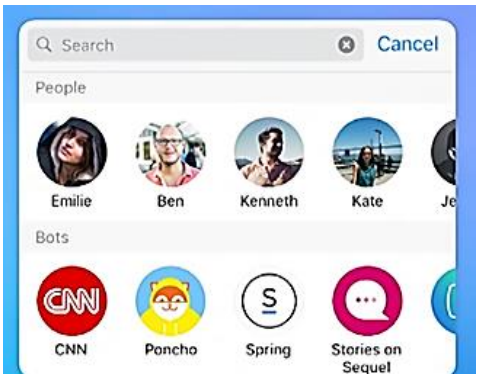


Figura 16 Nuova barra di ricerca Messenger con persone e bot.

Tutte queste soluzioni inviano l'utente ad una schermata con un testo di benvenuto al centro ed una chiamata all'azione "GET STARTED", che permette di iniziare la conversazione con un semplice click.

I bot al momento non sono creati per gli annunci pubblicitari, seppure in un futuro, probabilmente, saranno presenti. Inoltre è sempre l'utente che decide di iniziare a interagire con un bot. I bot non sono automaticamente pubblici ma devono superare una fase di revisione da parte del team di Messenger.

2.4 Api.ai: Elaborazione del linguaggio naturale

Api.ai è stato utilizzato come servizio per elaborare il linguaggio naturale.
Esso fornisce agli sviluppatori strumenti avanzati per costruire interfacce conversazionali. Permette ai bot di riconoscere ciò che l'utente, tramite messaggi testuali o la voce, gli dice e fornisce risposte accurate. [14]

2.4.1 Caratteristiche principali

Caratteristiche principali:

- **Soluzione completa.** Fornisce strumenti per capire il linguaggio naturale, al fine di progettare scenari di conversazioni a cui corrispondono delle azioni ed analizzare le interazioni con l'utente.
- **Pacchetti di conoscenza inclusi.** Possiede numerosi pacchetti di conoscenza, come meteo, notizie, orari voli, che permettono ai bot di svolgere nuove funzioni in modo pressoché immediato.
- **Machine Learning.** La piattaforma apprende sia dagli esempi forniti dagli sviluppatori che dalle conversazioni con gli utenti finali al fine di migliorare continuamente.
- **Supporto alla conversazione.** Permette di svolgere conversazioni con più argomenti, ricordando dove ogni argomento viene interrotto.
- **Integrazioni.** Sono presenti integrazioni con Alexa, Cortana e piattaforme di messaggistica (Facebook, Slack, Skype, Cisco Spark, Twilio Sms, ecc).
- **Facile da usare.** È davvero semplice creare e distribuire un agente di conversazione in pochi giorni.
- **Supporto multi-piattaforma.** SDK per iOS, Mac OS X, Apple Watch, Android, HTML, JavaScript, Node.js, C#, Python, Unity, Ruby, Xamarin, ed altre.
- **Supporto multi-lingua.** Al momento supporta 14 lingue, ma le altre arriveranno a breve.
- **Alte prestazioni.** Elaborano milioni di richieste utente al giorno con un uptime garantito del 99.99%.

Prezzo

Sono disponibili due versioni, una "standard" gratuita ed una "preferred" con prezzi variabili in base alle esigenze. La versione gratuita oggi offre query illimitate, una larghezza di banda limitata, utenti privati, l'uso dei propri dati da parte di Api.ai (proprietà molto interessante se si trattano dati privati, nella versione a pagamento ciò è facoltativo), il supporto da parte della community e non con Email, Chat, Telefono. L'uptime garantito del 99.99% e la consulenza sono presenti solo nella versione a pagamento.

2.4.2 Come funziona Api.ai

Concetti chiave di Api.ai sono:

1) **Agenti**

   Corrispondono a un bot o a un device che vorrebbe aggiungere un'esperienza utente conversazionale. Può essere descritto come un modulo per applicazioni NLU (Natural Language Understanding). Il suo scopo è quello di trasformare il linguaggio naturale dell'utente in dati fruibili.

2) **Entità**

   Rappresentano i concetti che sono spesso specifici per un dominio, come un modo di mappare le frasi del linguaggio naturale con delle frasi canoniche che catturano il loro significato. Ad esempio, se vogliamo modellare una playlist possiamo avere le entità "@artista" e "@canzone" con all'interno i vari artisti ed i loro sinonimi, è come un piccolo database interno.

   Sono molto importanti per estrarre dei parametri espressi in linguaggio naturale. Le entità dipendono dai valori che devono essere restituiti dall'agente, ciò implica che non devono essere create entità per ogni concetto, ma solo per quelli necessari.

   Esistono 3 tipi di entità:

   - System, definite da api.ai
   - Developer, definite dallo sviluppatore
   - User, costruite per ogni utente finale in ogni richiesta

Ciascuno di questi tipi può essere:

- Mapping, ha un valore di riferimento
  Es: @sys.date restituisce un valore nel formato ISO-8601 “January 1, 2015” oppure “The first of January of 2015” in "2015-01-01T12:00:00-03:00"

- Enum, non ha un valore di riferimento
  Es: @sys.color restituisce il colore senza un valore di riferimento come rgb. Sfumature di rosso come “scarlet” o “crimson”, non saranno mappate in “red” ma verranno restituiti i valori originali “scarlet” e “crimson”.

- Composite, contiene altre entità con alias
  Es: @sys.unit-currency per gli importi di denaro restituisce sia la quantità che la valuta, “50 euros” o “twenty dollars and five cents”. Restituirà un oggetto il cui tipo è una coppia di due attirbuti-valori: {"amount":50,"currency":"EUR"}

Per riferirsi alle entità normalmente si utilizza @entity oppure @entity:alias per quelle di tipo composite.

3) Intenti

Rappresentano una mappatura tra ciò che un utente dice e quale azione dovrebbe essere presa dal software. L’interfaccia per definire un intento è composta da 4 sezioni principali:

- Ciò che l’utente dice
- L’azione da eseguire in base a ciò che l’utente dice
- La risposta che dovrà essere inviata all’utente
- I contesti

4) Azioni

Corrispondono alle fasi che l'applicazione prenderà quando intenti specifici vengono innescati dall’input dell'utente. Un'azione può avere parametri per specificare le informazioni che devono essere estratte dall’input utente.

5) **Contesti**

Sono stringhe che rappresentano l'attuale contesto dell'espressione utente. Questo è utile per differenziare le frasi che potrebbero essere vaghe e hanno un significato diverso a seconda di ciò che viene detto in precedenza. Ad esempio, "riproduci la prossima canzone"

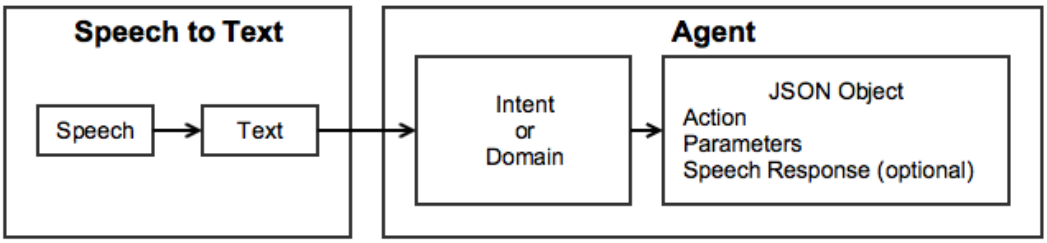


Figura 17 Schema del funzionamento di Api.ai.

Il funzionamento generale di Api.ai inizia con un testo o con un audio, dal quale viene riconosciuto il significato tramite un intento definito dallo sviluppatore oppure dai domini già presenti sulla piattaforma. Successivamente dal testo si estraggono gli eventuali parametri richiesti per soddisfare la richiesta, si segnalano eventuali azioni che un'applicazione esterna dovrà eseguire ed, eventualmente, si formula un testo di risposta.

Esempio

L'utente **scrive** al bot, "Voglio un piatto di pasta". Api.ai riconosce l'**intento** dell'utente e chiede i **parametri** necessari a svolgere l'ordinazione, come ingredienti, indirizzo di spedizione, ecc. Dopodiché esegue l'**azione** che segnala al cuoco di iniziare a cucinare e **risponde** all'utente con un messaggio di conferma.

2.5 Progettare i dialoghi

Prima di iniziare a costruire il proprio bot, un'ottima progettazione dei dialoghi può essere un fattore chiave. Essendo il bot inizialmente senza conoscenza o quasi, dovrà apprendere mediante delle conversazioni di esempio. La gestione dei dialoghi è una disciplina diffusa, dal cinema ai videogiochi, dai libri interattivi ai nostri chatbot. Ci sono vari metodi per la scrittura delle conversazioni, spesso sviluppati internamente per soddisfare le esigenze specifiche di particolari team, come la comunicazione dei requisiti per gli sviluppatori. Tuttavia ancora non si è giunti ad un metodo generale che possa essere usato da tutte le aziende. Fra essi uno molto interessante è il metodo Conversation Mapping (CM).[15]

2.5.1 Metodo mapping per descrivere conversazioni

L'ispirazione di tale metodo è il metodo sviluppato da Jeff Patton e descritto nel suo libro User Story Mapping (USM).

Figura 18 Esempio di User Story Mapping metodo di Patton [16].

Patton racconta:

"Costruire una mappa è molto semplice. Lavorando insieme agli altri, vi dirò la storia di un prodotto, scrivendo ogni grande passo che gli utenti prendono nella storia su dei post-it in un flusso da sinistra a destra. Poi, andremo indietro e parleremo dei dettagli di ogni passo, e scriveremo quei dettagli su dei post-it che metteremo in verticale sotto ogni passo. Il risultato è una semplice struttura a griglia che racconta una storia da sinistra a destra, e si va nei dettagli dall'alto verso il basso. È divertente e veloce." [16]

Il metodo Patton non è immediatamente applicabile al nostro problema di descrizione dei dialoghi. Due sono le principali modifiche necessarie affinché possa mappare delle conversazioni. La prima è scrivere delle note sui post-it, la seconda è catturare i percorsi ramificati che le conversazioni possono prendere.

Nel CM ogni post-it rappresenta un momento singolo della conversazione. In un dialogo generico ci sono individui che parlano, che ascoltano e che fanno domande. Per ognuna di queste azioni vi sarà un proprio post-it individuale.

Figura 19 Esempio post-it del metodo Conversation Mapping (CM).

Se nel metodo USM i team lavorano sulla storia di un singolo utente, nel CM tale progettazione avviene su un solo partecipante alla conversazione, ovvero il chatbot.

La logica è del tipo: ascolto un messaggio dall'utente e gli rispondo con una frase interrogativa o affermativa. Esattamente come funziona l'interfaccia grafica di API.ai

2.5.2 Problemi principali della conversazione con bot

Il modo di conversare di un chatbot è forse il concetto più importante per l'usabilità. All'utente finale non interessa il tipo di architettura o il database ma che il bot capisca ciò che lui dice e che gli risponda in modo adeguato.
Progettare un chatbot non è solo un'attività di tipo informatico, c'è di più.
Una frase molto carina di uno sviluppatore Facebook ne esprime il concetto

"the thing that makes me more excited to come to work every day is to spend half of my time understanding how computer works and half of the time understanding how PEOPLE works." – Emanuel Mazzilli ex studente di Ingegneria Informatica di Unimore [17]

Anche i migliori chatbot dei grandi player continuano a sbagliare.[18]

Figura 20 Esempio di errore chatbot.

La rete è piena di articoli che dettano regole per migliorare la progettazione dei dialoghi. Un articolo molto interessante di Carylyne Chan[19] descrive 5 aspetti da tenere in mente durante la progettazione:

1) **Costruire casi limite**

   Ci sono vari casi che i bot difficilmente sono in grado di gestire

   - **Domande a risposta multipla:**

     "Avete l'ultimo modello del mio telefono? Se si quanto costa?"

     Classico esempio di domanda costituita da più parti, molto difficile da gestire, in quanto il bot deve capire il rapporto tra le varie frasi. Se il bot risponde solo alla prima domanda, o non capisce che le domande sono correlate, gli utenti inizieranno ad avere un'interazione frustrante.

   - **Domande complesse**

     "Ho provato a riavviarlo, ho tolto anche la batteria ma niente, cosa fare?"

- **Domande temporali**
  "Il sito non funziona, non riesco ad accedere"
- **Richieste di funzionalità**
  "Dovreste aggiungere la voce per aiutarmi davvero"
- **Insulti**
  "Sei inutile"
- **Chiacchiere**
  "Dove sei nato?"

Tutti questi casi possono essere limitati aggiungendo la loro gestione ai bot. Proprio in questi casi i bottoni e le risposte multiple possono essere davvero di aiuto affinché la conversazione sia corretta e veloce.

2) **Comprendere il contesto**
   Da un lato spesso le frasi sono ambigue, dall'altro l'utente può aver avuto già interazioni con il supporto tecnico e quindi probabilmente dovrà ripetere tutti gli step di sicurezza o altro. Il bot non deve essere un sistema isolato.
3) **Integrazioni con il sistema aziendale**
   Il bot deve avere accesso ai dati utente in modo da rispondere a domande specifiche e non solo le faq. Lo stato di consegna di una spedizione ne è un esempio.
4) **Avere un protocollo di fuga**
   Se necessario valutare la possibilità di deviare la conversazione ad un esperto di quel settore.
5) **Imparare automaticamente**
   Deve essere in grado di analizzare le interazioni precedenti automaticamente in modo da apprendere e migliorare. Sebbene l'insegnamento manuale, come l'inserimento di sinonimi non possa essere eliminato.

2.5.3 Ispirarsi ai bot più diffusi

Oggi giorno sono già presenti ben 11000 bot su Messenger alcuni dei quali hanno ottenuto un successo rilevante. Un ottimo modo per cercare ispirazioni è analizzare i bot più famosi.[20]
Ecco una lista dei chatbot più diffusi sul mercato americano:

- HealthTap[15], per fare domande e inviare esami clinici a veri dottori
- Spring[16], shopping su Messenger
- EstherBot[17], una ragazza ha trasformato il suo curriculum cartaceo in un chatbot
- Surveybot[18], sondaggi su Messenger
- 1-800-Flowers.com[19], vendita di fiori

Di seguito vengono riportati piccoli spunti dell'articolo che li ha analizzati:

- Cercate di essere chiari nel messaggio di benvenuto affinché le aspettative dell'utente siano in linea su ciò che offrite. Se un utente cerca di comprare del pane in un negozio di fiori avrà sicuramente una pessima esperienza.
- Effettuate domande semplici per capire le esigenze dell'utente. A volte i bottoni per rispondere sono utilissimi per velocizzare l'interazione. Non utilizzare l'interazione testuale dove non è necessaria. Anche digitare "si", "no", "forse" può essere frustrante.
- L'interazione deve essere piacevole e non una fatica. I chatbot devono raccontare storie, non sono dei risponditori automatici come quelli già esistenti.
- Essi deve essere percepiti non come un robot ma come persone interessanti e simpatiche.
- Quando domandate cercate di evitare il modello "interrogatorio" preferendo quello "chiacchierata". Le domande migliori sono quelle a cui si può rispondere con un numero, un si / no o un massimo di tre parole.

Per valutare se un bot può aver successo è consigliabile porsi le seguenti domande:

- Offre un vero beneficio ad una parte dei 900 milioni di utenti?
- È di uso frequente? Come ad esempio le previsioni del tempo
- Quali sono i vantaggi fra un bot proattivo ed un altro reattivo? Ovvero fra bot che inviano autonomamente messaggi agli utenti o bot che rispondono in base alle richieste degli utenti?

In appendice A è presente una lista di vari strumenti inerenti ai bot suddivisi per categorie.

---

[15] https://www.messenger.com/t/healthtap/
[16] https://www.shopspring.com/ è il sito del produttore, non siamo riusciti a testare il chatbot.
[17] https://www.messenger.com/t/helloestherbot/
[18] https://surveybot.io/ permette di creare sondaggi attraverso bot, molto facile da usare.
[19] https://www.messenger.com/t/1800flowers/

2.6 DBMS a grafo

Una base di dati a grafo, o database a grafo, è una tipologia di database che utilizza strutture a grafo per rappresentare e archiviare l'informazione. I database a grafo sono molto recenti e si stanno diffondendo sempre di più, rimpiazzando delle tecnologie molto più studiate e ben stabilite nel corso degli anni. Questo perché permettono di ottenere notevoli miglioramenti in brevissimo tempo su certi domini di problemi caratterizzati da un modello concettuale a rete, quali reti semantiche, telematiche, energetiche.

2.6.1 Tipologie di grafi: il Labeled Property Graph

La definizione di Wikipedia di grafo afferma: “Un grafo è un insieme di elementi detti nodi o vertici che possono essere collegati fra loro da linee chiamate archi o lati o spigoli.” [21] Esistono grafi orientati e grafi non orientati. Un grafo orientato è composto da archi orientati cioè caratterizzati da una direzione.[22]

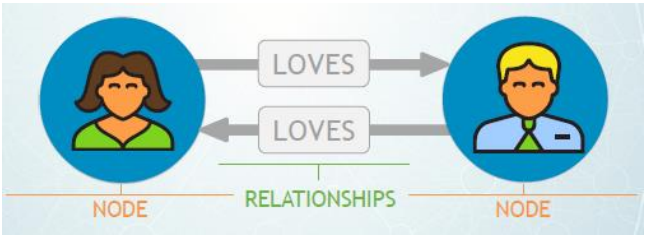


Figura 21 Esempio di grafo orientato (18).

Esistono vari modelli di grafi per rappresentare la realtà, spesso vengono mappate le entità sui nodi e le relazioni sugli archi. I più comuni sono i “Property graphs”, “Hypergraphs” e “Triples”. [23] Fra essi il modello più famoso e anche l’unico modello di cui parlerò è il “Labeled Property Graph”.

Le caratteristiche del Labeled Property Graph sono:

1. Contiene nodi e relazioni
2. I nodi possono avere delle proprietà descritte come coppie chiave-valore
3. Ai nodi possono essere assegnate delle etichette
4. Le relazioni hanno un nome e sono sempre direzionate con un nodo di partenza ed un nodo di arrivo
5. Le relazioni possono avere delle proprietà descritte come coppie chiave-valore

Questo modello, molto intuitivo e facile da capire, può essere utilizzato in tanti casi d'uso ed è quello più naturale per modellare chatbot. Nella Figura 22 è possibile avere un esempio di un Labeled Property Graph[20].

Ci sono due tipi di nodi, uno con la l'etichetta PERSON e uno con l'etichetta CAR. Ci sono 5 relazioni di 4 diversi tipi: LOVES, LIVES WITH, DRIVES, OWNS. Ogni relazione è direzionata. Tutti i nodi hanno delle proprietà con i rispettivi valori, ad esempio l'unico nodo con l'etichetta Car ha le proprietà brand e modello. Nell'esempio è presente anche una relazione, Drives, con una proprietà, Since.

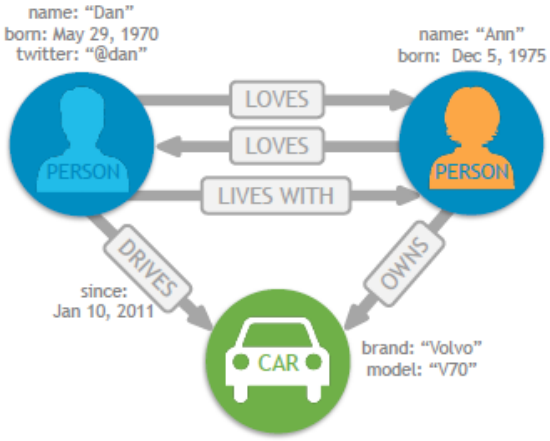


Figura 22 Esempio di Labeled Property Graph (18).

2.6.2 Vantaggi dei database a grafo

I punti di forza principali dei database a grafi sono: [23,24]

1) **Prestazioni**: In casi d'uso specifici, ad esempio con dati molto connessi fra loro, si possono velocizzare le operazioni di vari ordini di grandezza, si passa da minuti a millisecondi oltre ad avere query 10-100 volte più brevi. A differenza dei database relazionali, dove le prestazioni delle query join-intensive tendono a deteriorarsi all'aumentare di dati, con un database a grafo le prestazioni tendono a rimane relativamente costanti, anche se i dati aumentano. Questo perché le query sono

[20] Molti termini seppur con un'analoga traduzione vengono lasciati in lingua originale per non creare inutile confusione al lettore ed evitare di creare traduzioni senza senso. La traduzione Italiana di questo modello sarebbe *Modello dei dati del grafo di Proprietà etichettato*

localizzate a porzioni del grafo. Ciò implica che il tempo di esecuzione di ogni query è proporzionale solo alle dimensioni della parte del grafo interessata a soddisfare tale richiesta, non alla dimensione complessiva del grafo.

2) **Flessibilità:** I database a grafo, non avendo una struttura prefissata, possono adattarsi ai numerosi cambiamenti richiesti. I grafi sono naturalmente additivi, ciò significa che possiamo aggiungere nuovi tipi di relazioni, nuovi nodi, nuove etichette, nuovi sottografi ad una struttura esistente senza interferire con le query esistenti e le funzionalità delle applicazioni. Ciò aumenta la produttività degli sviluppatori che non devono creare un modello esaustivo fin da subito, abbassa i costi di manutenzione e riduce i rischi, grazie al minor numero di migrazioni che dovranno essere svolte.
3) **Agilità**: Il modello dei dati è in grado di evolvere di pari passo con il resto delle nostre applicazioni, utilizzando tecnologie che si interfacciano perfettamente con le moderne metodologie di gestione del ciclo di sviluppo del software di tipo incrementale ed iterativo. I moderni database a grafo permettono di sviluppare rapidamente applicazioni e ne agevolano la loro manutenzione, principalmente grazie al modello Schema-free, alle API ed al linguaggio di interrogazione progettato per la connettività.

2.7 Database Neo4j

Neo4j è stato utilizzato come DBMS a grafo. Neo4j è Graph database management system, open source, sviluppato completamente in Java. Secondo db-engines.com è il database a grafo più popolare. In appendice C sono presenti le sue caratteristiche principali.

2.7.1 Schema dei dati

Neo4j memorizza le informazioni in un grafo usando pochi e semplici concetti:

1) **Nodi**

   Sono i record di dati nel modello trattato in questa tesi (modellano oggetti). I nodi possono avere delle proprietà. Graficamente vengono indicati con un cerchio all'interno del quale sono descritte le sue proprietà (il tipo è quindi semi-strutturato).

Figura 23 Un nodo con due proprietà.

Nodi simili possono avere proprietà diverse. Neo4j può memorizzare miliardi di nodi.

2) **Etichette**

Vengono associate ad insiemi di nodi. Permettono di creare gruppi di nodi con delle caratteristiche comuni (classi di oggetti). Un nodo può non avere etichette o averne. Le etichette non hanno proprietà. Sono indicate associando un colore ad un nodo.

Figura 24 Un nodo con un'etichetta, "Persona" indicata dal colore rosso.

3) **Relazioni**

Si usano per descrivere che relazione c'è fra i nodi. Ogni relazione unisce due nodi. Una relazione ha sempre una direzione e un tipo. Individuano dei pattern di dati, ovvero determinate strutture di nodi e relazioni. Si indicano con delle frecce direzionate con il tipo descritto al centro di esse, partono da un nodo ed arrivano su un altro nodo o sul nodo stesso di partenza. La rappresentazione è simile alle reti semantiche.

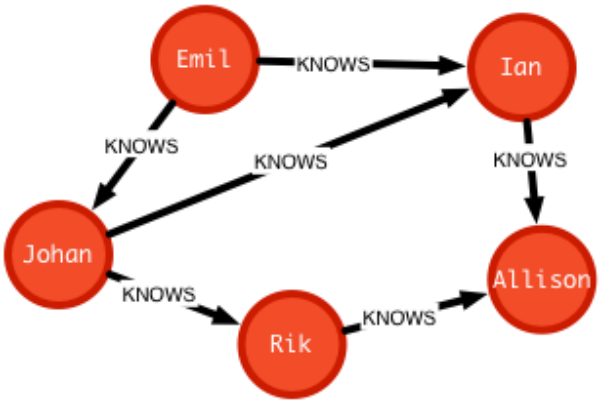


Figura 25 Grafo con 6 relazioni del tipo "Knows".

4) **Proprietà**

Sono semplici coppie nome/valore utilizzate per memorizzare i dati. Possono essere stringhe, numeri o booleani. Sia i nodi che le relazioni possono avere delle proprietà. Le proprietà delle relazioni indicano che le informazioni sono condivise da entrambi i nodi.

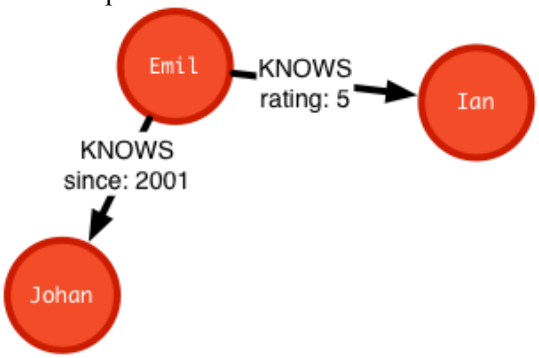


Figura 26 Grafo con 2 relazioni con proprietà.

2.7.2 Interrogazioni e modifiche

Il linguaggio di interrogazione nativo di Neo4j è il Cypher. Un linguaggio molto semplice e allo stesso tempo molto potente. Utilizza dei pattern per descrivere i dati del grafo, ha delle similarità con il linguaggio SQL ed è dichiarativo. Le operazioni più frequenti [25,26] sono:

1) **CREATE**

   La clausola CREATE serve a creare dati, sia nodi che relazioni.

   Es: query per creare un nuovo nodo con delle proprietà

```
CREATE (ee:Person { name: "Emil", from: "Sweden", klout: 99 })
```

()	Parentesi tonde per indicare un nodo
ee:Person	Al nuovo nodo viene assegnata una variabile 'ee' ed un'etichetta 'Person'. Le etichette vengono sempre precedute dal carattere ":" seguito dal nome dell'etichetta senza apici.
{}	Le parentesi graffe per aggiungere le proprietà al nodo.

2) **MATCH**

   La clausola MATCH permette di specificare dei pattern di nodi e relazioni.

   Es: query per cercare il nodo creato in precedenza

```
MATCH (ee:Person) WHERE ee.name = "Emil" RETURN ee;
```

(ee:Person)	Un pattern di un singolo nodo con l'etichetta 'Person' che assegnerà le eventuali corrispondenze alla variabile 'ee'
WHERE	Clausola per filtrare le corrispondenze trovate
ee.name = "Emil"	Confronta la proprietà "name" al valore "Emil"
RETURN	Clausola per definire cosa ottenere nei risultati, è possibile utilizzare degli alias ed altre funzioni al suo interno.

3) **CREATE multiplo**

   La clausola CREATE può creare anche più nodi e relazioni contemporaneamente.

   Es: query per creare due nuovi nodi, relazioni fra essi e il nodo dell'esempio precedente.

```
MATCH (ee:Person) WHERE ee.name = "Emil"
CREATE (js:Person { name: "Johan", from: "Sweden", learn: "surfing" }),
```

```
(ir:Person { name: "Ian", from: "England", title: "author" }),
(ee)-[:KNOWS {since: 2001}]->(js),(ee)-[:KNOWS {rating: 5}]->(ir),
(js)-[:KNOWS]->(ir),(ir)-[:KNOWS]->(js)
```

[]	Parentesi quadre per indicare la relazione
- >	Simbolo maggiore o minore per indicare la direzione della relazione.

4) Pattern matching

Consente di ricercare nel grafo mediante la descrizione di una determinata configurazione di nodi e relazioni detta pattern.

Es: query per cercare tutti gli amici di una persona chiamata Emil

```
MATCH (ee:Person)-[:KNOWS]-(friends)
WHERE ee.name = "Emil" RETURN ee, friends
```

(ee)	È il punto di inizio del pattern con un nodo etichettato "Person" specificato nella clausola WHERE
-[:KNOWS]-	Cerca le corrispondenze con le relazioni "KNOWS" in entrambi i sensi visto che non è specificata la direzione.
(friends)	È la variabile legata agli eventuali amici di Emil.

5) Raccomandazioni

Il pattern matching può essere utilizzato per effettuare delle raccomandazioni.

Es: query per cercare nuovi amici, ad una persona di nome Johan, che praticano surf

```
MATCH (js:Person)-[:KNOWS]-()-[:KNOWS]-(surfer)
WHERE js.name = "Johan" AND surfer.hobby = "surfing"
RETURN DISTINCT surfer
```

()	Le parentesi vuote servono ad ignorare questi nodi
DISTINCT	É necessaria per non avere risultati duplicate.
surfer	Conterrà gli amici di amici di Johan che praticano surf

6) DELETE

La clausola DELETE permette di eliminare nodi e relazioni identificati all'interno della clausola MATCH, eventualmente filtrati con la clausola WHERE. Non è

possibile eliminare un nodo senza prima eliminare tutte le relazioni che iniziano o terminano nel nodo stesso.

Es: query per un nodo mediante il suo identificatore e le sue relazioni.

```
MATCH (n)-[r]-()
WHERE id(n) = 5
DELETE r, n
```

id — Ogni nodo e ogni relazione hanno un identificatore interno numerico che viene attribuito automaticamente e può essere interrogato usando gli operatori <, <=, =, =>, <>, IN

7) REMOVE

La clausola REMOVE permette di rimuovere proprietà ed etichette dagli elementi del grafo.

Es: query per rimuovere una proprietà.

```
MATCH (soren {name: 'Soren'})
REMOVE soren.age
RETURN soren
```

Es: query per rimuovere un'etichetta.

```
MATCH (soren {name: 'Soren'})
REMOVE soren:Intern
RETURN soren
```

8) SET

La clausola SET permette di modificare le proprietà dei nodi e delle relazioni.

Es: query per aggiungere una proprietà ad una relazione, o modificarla se già esistente. Nel caso specifico una persona Mikey aggiunge un voto di 5 stelle a tutti i film che ha visto.

```
MATCH (fan:Person)-[w:WATCHED]->(movie)
WHERE fan.name = "Mikey"
SET w.rating = 5
```

9) MERGE

La clausola MERGE assicura che un determinato pattern esista nel grafo. È simile alla clausola CREATE con la differenza che il pattern viene creato solo se non esiste già.

Es: query per creare un nodo se non esiste e aggiornare (o creare) una proprietà.

```
MERGE (n:Person {name: "Rik"})
SET n.owns = "Audi"
```

2.7.3 Interagire con Neo4j

Gli sviluppatori possono interagire con Neo4j attraverso un linguaggio di interrogazione imperativo oppure dichiarativo. In questa tesi è stato usato il Cypher, il linguaggio dichiarativo nativo di Neo4j, perché facile da usare e da imparare. Tuttavia esistono altre API che, a seconda dei casi d'uso, possono essere utilizzate per interfacciarsi con Neo4j[23,27].

In un progetto è importante capire quale API utilizzare in base alle loro funzionalità. Tali API possono essere pensate in modo analogo ad una pila, come in figura seguente:

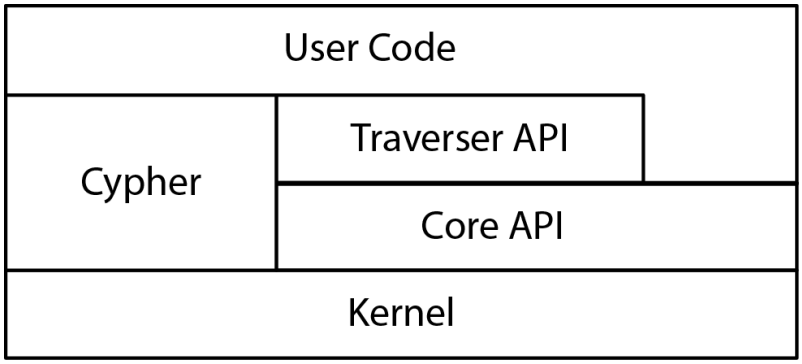


Figura 27 Vista logica delle API utente in Neo4j.[23]

In alto viene aumenta l'espressività e favorita la programmazione dichiarativa. In basso, vi è una maggior efficienza ed un linguaggio imperativo. Ogni API ha i suoi vantaggi e svantaggi, che si dovrebbero analizzare al fine di prendere la miglior decisione.

Kernel API

Al livello più basso della pila si trovano i "kernel's transaction event handlers". Quest'ultimi permettono al programmatore di monitorare le transazioni mentre fluiscono attraverso il kernel. In base al contenuto dei dati ed alla fase del ciclo di vita della transazione è possibile avere una reazione.

Core API

È un'API java di tipo imperativo che permette al programmatore di interagire con i vari elementi del grafo. Con essa si possono raggiungere maggiori prestazioni ma la struttura del grafo dovrà essere presente all'interno del codice Java e ciò va ad influire sull'agilità del sistema di evolvere in base alle variazioni del dominio dell'applicazione.

Traversal Framework

È un'API java di dichiarativa. Permette di interrogare il grafo specificando dei vincoli che limitano le parti del grafo a cui si può accedere con essa, ad esempio determinate relazioni da seguire. Poiché è un framework general-purpose tende ad avere prestazioni leggermente inferiori alle query ben formulate mediante la Core API.

2.8 Graphenedb.com

Graphenedb è stato utilizzato come hosting per il nostro database. Esso offre istanze di Neo4j nel cloud. Offre sia un piano gratuito per valutare la piattaforma ed effettuare dei test, che piani a pagamento.[28]

Le principali caratteristiche sono:

- Modello di pagamento basato sul consumo
- Risorse scalabili su AWS[21] e Azure[22] in diverse regioni, su richiesta, istantaneamente
- Backup online automatici e manuali
- Supporto per i plugin della community e le estensioni personalizzate
- Strumenti di gestione del database come log degli accessi al server e aggiornamenti di configurazioni
- Opzioni avanzate per Neo4j Enterprise Edition
- Disponibile come Heroku Add-on con un piano di prova gratuito.

[21] Amazon Web Services https://aws.amazon.com/it/
[22] Piattaforma e servizi di cloud computing Microsoft https://azure.microsoft.com/it-it/

2.9 Heroku.com

Heroku è stato utilizzato per distribuire la nostra applicazione online. É un Platform as a service (PaaS) sul cloud che supporta diversi linguaggi di programmazione, Ruby, Java, Node.js, Scala, Clojure, Python, PHP, e Go [29]. Secondo il sito ufficiale è la migliore piattaforma per sviluppare moderne architetture che scalano su richiesta. Permette di essere focalizzati sull'applicazione e non sul sistema operativo che viene gestito interamente dal loro team. Offre un ricco ecosistema di estensioni pre-integrate e servizi.[30]

Capitolo 3

Progettazione

Questo capitolo si occupa di descrivere gli aspetti chiave della progettazione del nostro prototipo di chatbot che chiameremo “Ibot”. I termini bot, chatbot, Ibot verranno usati per indicare l’intero sistema. Dopo un’analisi generale dei requisiti di tutto il progetto, la progettazione viene divisa in tre fasi sequenziali, la progettazione dei dialoghi, il progetto dei dati e l’interazione fra essi, come mostrato in figura.[31]

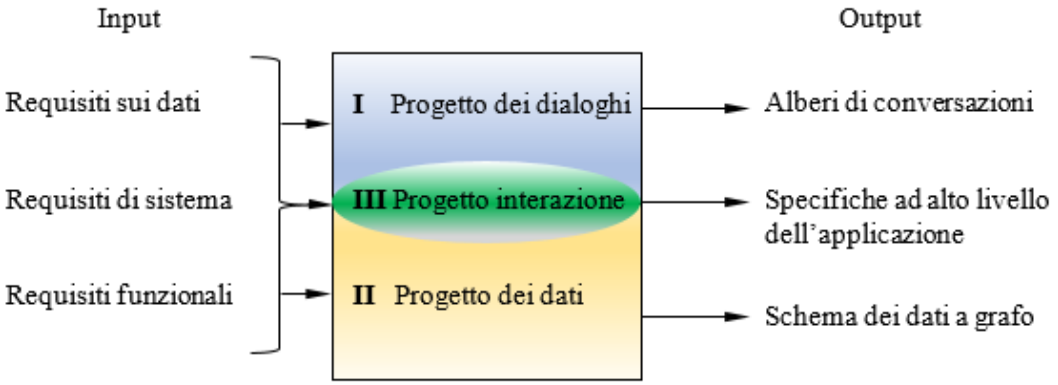


Figura 28 Schema metodologia di progetto.

3.1 Visione del sistema generale

Il nostro sistema Ibot in realtà è solo una parte di un sistema più ampio e complesso, che chiameremo sistema generale, SG. L’aspetto chiave di quest’ultimo è:

Creare dei collegamenti tra elementi non connessi fra loro.

Per spiegare il suo funzionamento, può essere utile usare delle analogie.
Pensiamo a SG come un servizio web che permette di organizzare dei viaggi, come Google Maps. Supponiamo che vogliamo effettuare un viaggio da Francoforte a Modena. Uno dei vari risultati è il seguente:

(treno)->(metropolitana)->(autobus)->(percorso a piedi)->(treno)->(treno)

Google Maps, quindi come il nostro sistema collega vari elementi, società di treni, autobus, aerei ed altri mezzi, che non sono collegati fra loro.
Una differenza essenziale fra Google Maps e SG è la modalità di interfacciarsi con gli utenti. Il primo utilizza un'interfaccia web classica, oppure un'applicazione mobile, il secondo, utilizza un chatbot. Ciò permette di dialogare con gli utenti. Il dialogo, oltre a rendere coinvolgente il servizio, permette al sistema di capire al meglio i bisogni dell'utente, che possono essere del tutto sconosciuti all'utente stesso.

Un'altra analogia utile a descrivere SG è pensarlo come fosse un medico specialista e non come un semplice motore di ricerca. Se una persona ha bisogno di una cura, può andare dal medico o cercare su internet. La differenza fra i due metodi è che nel primo, un esperto competente nella materia effettua una diagnosi e fornisce una cura subito, senza che il paziente conosca nulla di medicina, nel secondo è l'utente in prima persona che dovrà informarsi sulla cura, e ciò richiede tempo e può portare a cure sbagliate.

In Figura 29 viene mostrata l'applicazione del sistema al problema del miglioramento dell'inclusione dei rifugiati descritto nel capitolo 1.
Gli elementi che in questo caso il sistema va a collegare sono le varie offerte formative presenti nella città di Francoforte sul Meno. Tali connessioni creano dei percorsi di formazione necessari per ottenere dei certificati.

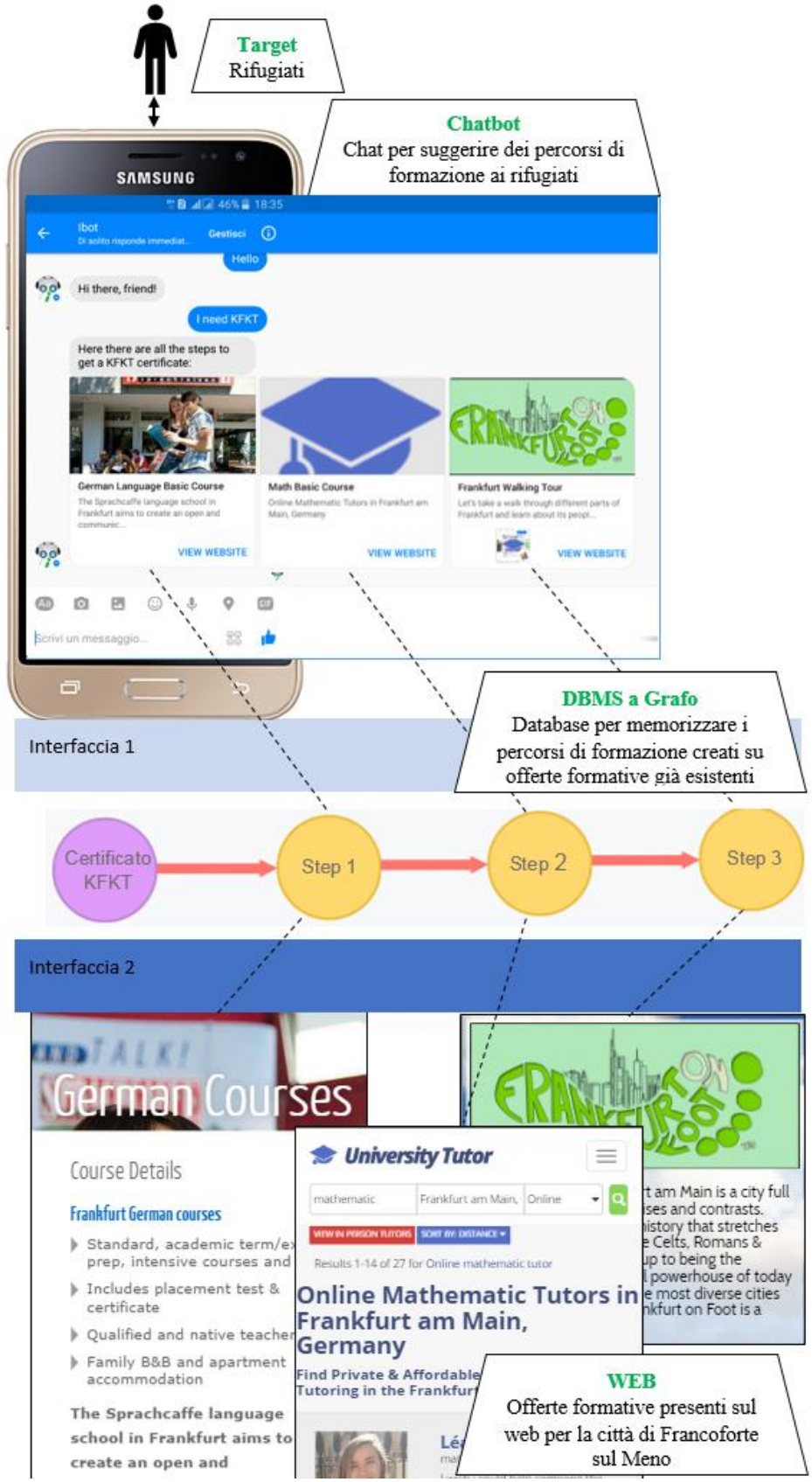

Target
Rifugiati
Chatbot
Chat per suggerire dei percorsi di formazione ai rifugiati
SAMSUNG
Ibot
Gestisci
Hello
Hi there, friend!
I need KFKT
Here there are all the steps to get a KFKT certificate:
German Language Basic Course
Math Basic Course
Frankfurt Walking Tour
VIEW WEBSITE
VIEW WEBSITE
VIEW WEBSITE
Interfaccia 1
DBMS a Grafo
Database per memorizzare i percorsi di formazione creati su offerte formative già esistenti
Certificato KFKT
Step 1
Step 2
Step 3
Interfaccia 2
German Courses
Course Details
Frankfurt German courses
Standard, academic term/ex
prep, intensive courses and
Includes placement test &
certificate
Qualified and native teacher
Family B&B and apartment
accommodation
The Sprachcaffe language
school in Frankfurt aims to
create an open and
University Tutor
mathematic
Frankfurt am Main,
Online
Results 1-14 of 27 for Online mathematic tutor
Online Mathematic Tutors in Frankfurt am Main, Germany
WEB
Offerte formative presenti sul web per la città di Francoforte sul Meno

Figura 29 Visione del sistema generale.

In figura, per indicare l'implementazione del sistema generale al caso pratico, sono presenti 4 trapezi:

1. **Target,** persone con caratteristiche comuni a cui il sistema offre soluzioni per problemi specifici.

Nell'esempio, il target consiste nei rifugiati della città di Francoforte che hanno bisogno di formazione e di certificazioni, ai fini di trovare un impiego.

2. **Chatbot**, un sistema basato su un insieme di regole del tipo, se l'utente dice una frase il sistema reagisce, ad esempio con un messaggio di risposta. Queste regole sono specifiche per il target selezionato e le funzionalità offerte.

Nell'esempio, i testi delle regole servono a capire le esigenze dei rifugiati e suggerirgli dei percorsi di formazione personalizzati. L'utente nella chat scrive "I need KFKT" (Key FranKfurt Test, acronimo inventato per simulare un certificato di base per lavorare a Francoforte), e il chatbot gli risponde mostrandogli il percorso di formazione necessario ad ottenere il certificato richiesto. Ci sono tre step: un corso di tedesco, un corso di matematica ed un giro turistico nella città.

3. **Web**, rappresenta un insieme di elementi non necessariamente collegati fra loro.

Nell'esempio, vi sono tre elementi che rappresentano le pagine web di tre servizi reali esistenti in un determinato istante di tempo:

- Sprachcaffe[23], per corsi di Tedesco dal vivo in città.
- Universitytutor[24], cerca tutor online per apprendere la Matematica.
- Frankfurtonfoot[25], offre un giro turistico nella città.

4. **DBMS a grafo**, utile ad archiviare i collegamenti fra gli elementi.

Nell'esempio c'è un solo percorso di formazione che collega le tre offerte formative.

[23] http://www.sprachcaffe.com/english/adults-german-courses-germany/frankfurt.htm
[24] http://online.universitytutor.com/online_mathematic-tutoring
[25] http://www.frankfurtonfoot.com/

Sempre in figura sono presenti due interfacce:

- **L'interfaccia 1** permette la comunicazione fra il Chatbot e il DBMS a grafo, costituendo insieme ad essi il sistema Ibot che verrà discusso in questo elaborato.

- **L'interfaccia 2**, permette di gestire il collegamento fra contenuti presenti nel web e la base di dati a grafo. In questo elaborato, ho assunto che essa è realizzabile e non verrà trattata. Alcune osservazioni su di essa:
  - Molte risorse sul web non offrono API per interfacciarsi con servizi esterni.
  - Le risorse sul web non hanno una struttura standard. Nel nostro esempio, alcune informazioni sui corsi sono perfino memorizzate in file pdf da scaricare, ciò mostra che non basta un semplice implementazione di web scraping[26].
  - Le informazioni e la loro organizzazione sono variabili nel tempo. Non è possibile quindi avere un sistema statico che memorizza una sola volta tutte le informazioni, ma è necessario un sistema dinamico che permetta di avere informazioni aggiornate.

[26] Tecnica informatica di estrazione di dati da un sito web per mezzo di programmi software, si veda https://it.wikipedia.org/wiki/Web_scraping

3.2 Analisi dei requisiti

Di seguito vergono analizzati tutti i principali requisiti necessari sia in fase di progettazione che in fase di implementazione. Altri requisiti specifici verranno discussi durante le singole fasi di progettazione, oppure omessi perché non particolarmente rilevanti.

La loro raccolta è avvenuta in linguaggio naturale secondo un processo di analisi iterativo costituito di tre fasi:

1) Raccolta dei requisiti e controllo dei requisiti precedenti, eventualmente implementati

2) Filtro ambiguità dei requisiti e progettazione inerente

3) Implementazione dove possibile.

3.2.1 Requisiti funzionali

La raccomandazione di percorsi di formazione é essenzialmente l'unica funzione nei confronti dell'utente finale di Ibot. Tale raccomandazione sarà basata sui dati acquisiti direttamente dall'utente durante le conversazioni con Ibot ed, eventualmente, con dati forniti da terze parti.

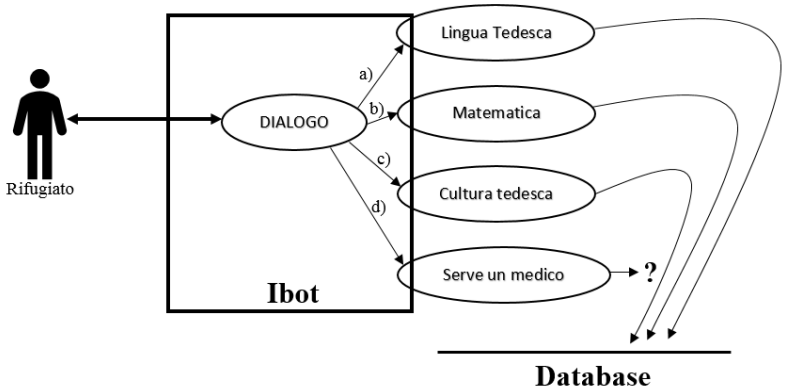


Figura 30 Schema funzionale che mostra l'interazione fra un rifugiato e il chatbot. Lo schema mostra quattro possibili tematiche del dialogo su cui il sistema può eventualmente consigliare dei percorsi di formazione.

Un requisito funzionale degno di nota, ma nel prototipo non realizzato, è la comunicazione nella lingua madre del rifugiato. L'unica lingua che verrà implementata nel prototipo è l'inglese ma in futuro si aggiungeranno altre lingue affinché il chatbot sia davvero utile a tutti i rifugiati. L'implementazione è comunque possibile, seppure la qualità delle conversazioni nelle lingue più comuni è maggiore. Tuttavia gli sforzi necessari a realizzare ciò in un prototipo non apportano un valore davvero reale ai fini della valutazione dell'idea. Per realizzare un bot multilingua ci sono dei servizi appositi, oppure è possibile crearli

manualmente con l'unione di vari agenti, uno per ogni lingua desiderata, e utilizzare servizi di traduzione per offrire i contenuti del database, come ad esempio la descrizione di un corso. Per semplicità di trascrizione alcuni dei seguenti esempi verranno presentati in inglese come implementati nel prototipo.
Un'ulteriore osservazione sulla lingua del chatbot è il livello di conoscenza. Il chatbot deve essere in grado di riconoscere le intenzioni degli utenti anche se commettono errori grammaticali o di digitazione.

3.2.2 Requisiti sui dati

Di seguito sono elencate le specifiche generali espresse in linguaggio naturale che il sistema dovrebbe essere in grado di gestire. In esse sono contenuti i requisiti sui dati necessari alla progettazione dello schema del database.

Gli elementi principali del nostro sistema sono:

- Rifugiati, coloro che converseranno con Ibot
- Percorsi di formazione, sequenze ordinate di passi
- Passo, entità che rappresenta una qualsiasi attività di aiuto per la formazione ed integrazione dei rifugiati nella città. Ad esempio, corsi di lingua, tour per la città, partite di calcio, cene di gruppo, ecc. Ci saranno caratteristiche comuni a tutti i passi, come la presenza di responsabili, ed altre completamente specifiche.

Il sistema è in grado di dialogare con i rifugiati al fine di ottenere maggiori informazioni necessarie a consigliare i migliori percorsi di formazione ai rifugiati. Tali informazioni NON devono essere memorizzate per permettere ai rifugiati di esprimersi liberamente ed evitare problemi dovuti al trattamento di dati sensibili.
Il dialogo è formato da varie domande e può terminare sia in base al volere del rifugiato, che del sistema. Il dialogo avviene in base a determinati argomenti. Le domande possono essere a scelta multipla o risposta aperta e saranno focalizzate principalmente a determinare i bisogni, le passioni, le abilità e i desideri personali dei rifugiati.
Il sistema può suggerire dei percorsi di formazione che possono prevedere dei certificati di partecipazione e/o con un esame. Possono esserci percorsi specifici, ad esempio Lingua, oppure generali che racchiudono in sé più discipline. I percorsi possono essere composti da qualsiasi tipo di attività.

Le attività, anche chiamate passi, sono identificate da un codice univoco e sono caratterizzate da vari campi generali obbligatori e campi specifici. Ad esempio, numero di partecipanti, descrizione orari, ecc. Ai fini della raccomandazione le attività hanno delle categorie, delle parole chiave, dei requisiti facoltativi ed obbligatori (l'età).

Osservazioni sui requisiti

Prendiamo in considerazione il requisito:
Il sistema non memorizza nessun dato sui rifugiati.

Vantaggi: Semplifica gli aspetti legali riguardanti il trattamento dei dati personali, permette ai rifugiati di esprimersi più liberamente, facilita la realizzazione e la gestione del prototipo ed ha un costo minore in termini di risorse.
Svantaggi: Non è possibile analizzare la storia e apprendere profili tipici che potrebbero essere riutilizzati.
Contrariamente, si potrebbero memorizzare dati sui rifugiati. Ci sono varie informazioni utili che possono essere estratte da ogni conversazione. Per iniziare possono essere estratti tutti i dati del profilo pubblico di Facebook, fb id, nome, cognome, immagine del profilo, genere, lingua, fuso orario, che sono immediatamente disponibili e ben strutturati. In seguito memorizzare i dati relativi alle conversazioni, eventualmente rendendoli anonimi.
Vantaggi: É possibile avere dei dati da analizzare per migliorare il sistema, sia dal punto di vista del singolo utente, migliorando le conversazioni e le raccomandazioni, che globale, monitorando il corretto funzionamento del sistema, la quantità di utenti attivi, ecc.
Svantaggi: la gestione di tali dati richiede particolari autorizzazioni, un sistema più complesso, maggiori risorse.
Un altro aspetto inerente alla memorizzazione è l'utilizzo di servizi esterni per gestire i dati. Come vedremo successivamente, ci sarà un servizio esterno per elaborare il linguaggio naturale ed è proprio quest'ultimo che, a seconda della tecnologia utilizzata, spesso memorizza tutte le conversazioni che elabora ai fini di migliorare le sue funzionalità.

Il prototipo da realizzare si occuperà di tutte le interazioni con il rifugiato. La gestione delle altre funzioni che non richiedono interazione con i rifugiati, come l'inserimento delle attività o degli esiti nel libretto, saranno possibili direttamente mediante operazioni sul database stesso in quanto non di rilievo ai fini della valutazione dell'idea.

3.2.3 Requisiti di sistema

a. **Accessibile via web** affinché il prototipo possa essere mostrato ovunque vi sia una connessione internet
b. **Applicazione su cloud** per gestire l'app direttamente online e rimanere sempre a disposizione per chiunque continui il prototipo
c. **Minimo di risorse economiche**
d. **Efficace** in grado di mostrare l'interazione simile all'applicazione finale in termini di velocità, grafica, stabilità.

3.3 Architettura

L'architettura scelta per Ibot è quella tipica a 3 livelli descritta nel capitolo precedente, scelta dovuta alla necessità di inserire una propria logica all'interno dell'applicazione ed un proprio database.

3.4 Fasi della progettazione

Come anticipato nell'introduzione, la progettazione viene divisa in tre fasi sequenziali, la progettazione dei dialoghi, il progetto dei dati e l'interazione fra essi. La prima fase si occupa delle conversazioni le quali saranno gestite interamente sul bot Engine. La seconda fase, dei dati da memorizzare nel nostro database. La terza fase si occupa delle interazioni fra il bot engine ed il database, gestendo le loro informazioni comuni. Ad esempio, se nel dialogo sono presenti dei quiz, le soluzioni devono essere memorizzate nel database per effettuare la verifica delle risposte corrette.

3.5 I - Progetto dei dialoghi

3.5.1 L'importanza dei dialoghi: Curva emozionale

Il dialogo con i rifugiati è un punto molto sensibile, perciò lo diventa anche la fase di progettazione dei dialoghi, la quale richiede principalmente competenze in psicologia e linguistica, piuttosto che in informatica. Sarebbe interessante effettuare uno studio sulle conversazioni fra i rifugiati e un chatbot.[32]

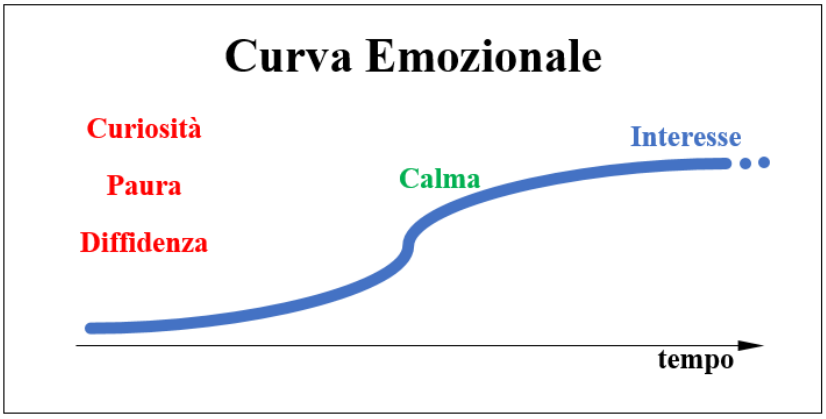


Figura 31 Curva emozionale teorica durante l'uso del chatbot da parte di un rifugiato ispirata dalla scala emozionale descritta nel libro "Analisi emozionale del testo"[32].

Sicuramente, ci sono varie emozioni che un rifugiato prova utilizzando il chatbot. Emozioni che cambiano in base alle conversazioni. Nella Figura 31 viene mostrata una curva emozionale teorica della prima conversazione di un rifugiato con il bot.

Nella curva, si possono notare: una prima fase in cui può essere curioso delle funzionalità ed allo stesso tempo spaventato di comunicare con un bot e non con un umano. Può succedere che non si fidi del bot e inserisca informazioni false. Una seconda fase, in cui si sente calmo e disposto a valutare effettivamente le funzioni del bot. Un'ultima fase, in cui capisce le opportunità offerte dal bot ed ha un notevole interesse nella conversazione. Questa curva appena vista è molto ottimista ma basterebbe ben poco per avere uno scenario del tutto opposto, per cui i dialoghi rappresentano il cuore di tutto il sistema.[33]

3.5.2 Modellare i dialoghi: modello a blocchi

Secondo i principi dell'apprendimento auto-direzionato, la conversazione con i rifugiati dovrebbe principalmente permettere loro di capire l'importanza di apprendere determinate tematiche, valutare le proprie competenze e mostrare i temi su cui dovrebbero migliorare, per poi definire un percorso di apprendimento personale con relativi obiettivi.[2]
Modellare una conversazione non è banale. Ci sono numerose conversazioni possibili nel nostro linguaggio. Come ogni problema complesso è utile suddividerlo in tante parti.
Il principio della nostra modellazione si basa su un concetto: ogni conversazione ha un obiettivo. Per cui abbiamo deciso di dividere la conversazione in blocchi.

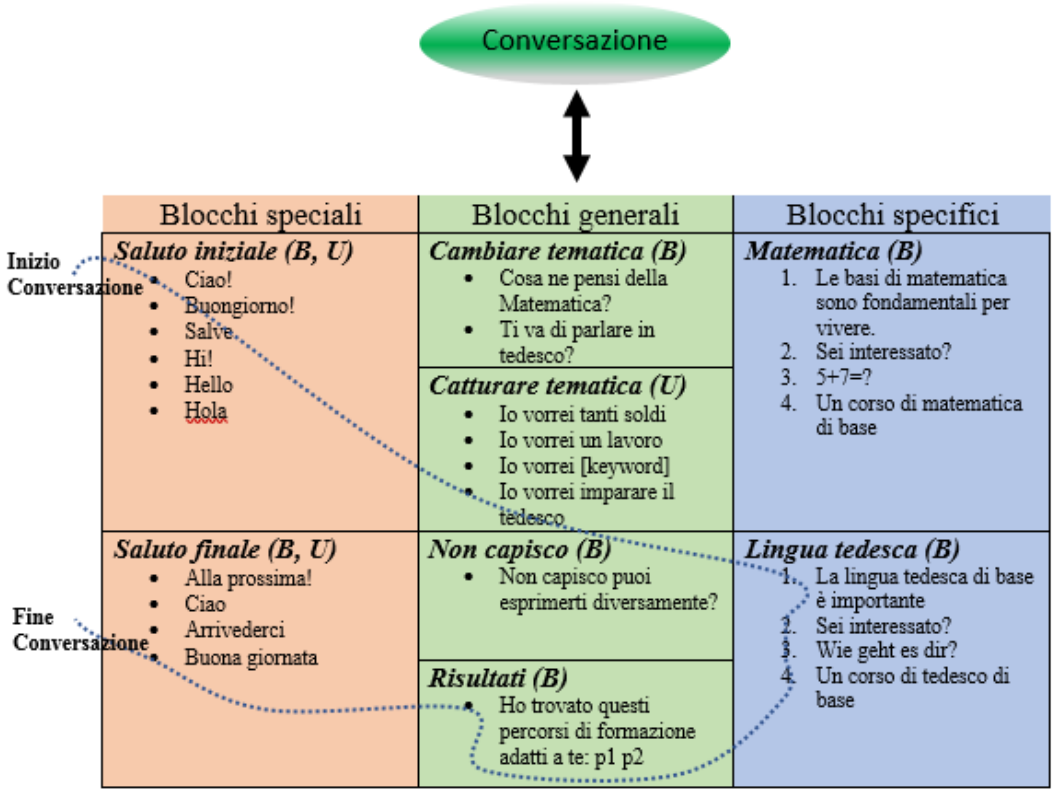


Figura 32 Mostra la suddivisione della conversazione in blocchi speciali, generali e specifici. Ogni blocco ha un proprio nome gli agenti che scrivono le frasi, B per bot U per utente. Gli elenchi puntati indicano che le frasi sono opzionali, gli elenchi numerati indicano che le frasi sono in successione. La linea tratteggiata mostra un esempio di conversazione fra le varie possibili.

Ogni blocco è composto da frasi con un proprio obiettivo. Esistono vari tipi di blocchi:

1) **Blocchi speciali**, sono i blocchi che si ripetono quasi sempre come i saluti di inizio conversazione o fine conversazione.

2) **Blocchi generali**, non hanno una tematica principale ma sono utili a diverse funzioni, come collegare dei blocchi specifici, intercettare le intenzioni dell'utente, gestire i casi sconosciuti, ecc.
3) **Blocchi specifici,** hanno una tematica principale che può essere organizzata secondo vari template.

I blocchi racchiudono varie frasi che possono essere utilizzate dal bot (B) e/o dall'utente (U). Ci sono frasi opzionali, le quali vengono mostrate singolarmente in maniera random all'utente ogni volta che la conversazione attraversa quel determinato blocco. Alcuni blocchi invece seguono delle sequenze logiche ben determinate costituite da più frasi mappate su degli alberi conversazionali.[34]
Tutte le conversazioni possibili attraversando i vari blocchi possono essere immaginate su un grande insieme di alberi conversazionali connessi fra loro.

Nei blocchi specifici abbiamo modellato un "blocco bisogno" necessario a eseguire tutti gli step richiesti dall'apprendimento auto-direzionato. I bisogni che un rifugiato può avere sono stati mappati in step.[35]

Un esempio utilizzato nel prototipo è il bisogno del tedesco, dove il bot, non appena la conversazione entra nel blocco di riferimento, utilizza le seguenti domande ed affermazioni suddivise per fasi:
Presentazione del bisogno: "*Many Germans unfortunately don't speak your native language but they would like to help you! Start to learn German to get many results in every field here in Germany!"*
Valutare il desiderio: *"Are you interested in German Language?"*
Prendere le misure*: "Cool! Can I ask some question to understand better your knowledge?"*
Offrire una soluzione: "*All right!! Let me check what I can do for you! Here are the best courses for learning German. Make your choice considering the possibility of a certificate*"

.

Come nell'immagine seguente (Figura 33) il blocco bisogno è composto da sottoblocchi. I sottoblocchi sono sequenziali ma non tutti sono obbligatori durante una conversazione. Un rifugiato può chiedere "Sono interessato ad eventi culturali" ed automaticamente

iniziare la conversazione direttamente dalla terza fase, oppure dire che non è interessato e saltare le fasi 3 e 4 e, magari, entrare in un nuovo blocco che si occupa di gestire la motivazione del disinteresse, può darsi che il rifugiato sia già competente in quella disciplina. L'ideale sarebbe memorizzare tali informazioni in modo strutturato al fine di evitare di ripetere le stesse domande ogni volta che l'utente usa il chatbot e gestire l'avanzamento nel tempo delle informazioni. Ciò non verrà effettuato per rispettare i requisiti sui dati di privatezza.

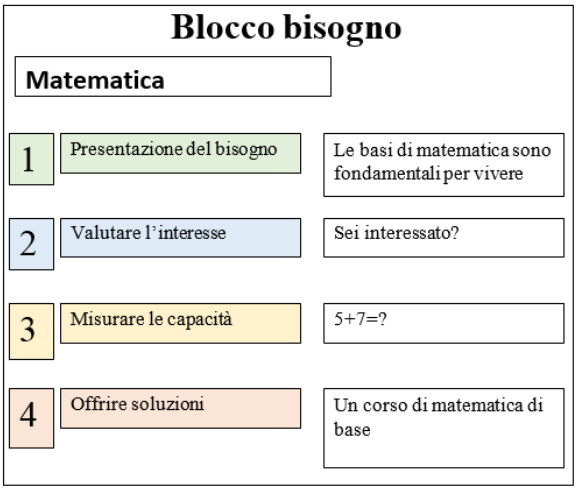


Figura 33 Modellazione blocco bisogno con 4 sottoblocchi utilizzato per modellare la matematica.

3.5.3 Scrivere e disegnare i dialoghi

Esistono vari strumenti digitali per scrivere e disegnare dialoghi, alcuni creati esclusivamente per chatbot che permettono di avere una vista grafica dei dialoghi utilizzando gli elementi tipici delle piattaforme di messaggistica come tutti quelli descritti nel capitolo 2.

Ne abbiamo alcuni degni di nota:

- Arctouch template[27], è un template per un programma di grafica e design chiamato Sketch[28]. Si basa sul concetto di "drag e drop" degli elementi grafici, pulsanti, messaggi di testo, carosello, ecc. è ottimo per descrivere brevi dialoghi efficacemente. Di contro, è necessario il sistema operativo del mac per utilizzarlo, imparare ad usare il programma che è a pagamento (offre una versione di prova per un mese), leggere la documentazione del template, e dedicare vari minuti per ogni elemento creato.
- Twinery[29], è un programma open source per creare delle storie interattive non lineari. È possibile creare le storie direttamente online ed esportarne il codice sorgente. Tuttavia richiede molto tempo per disegnare grandi dialoghi.

Degli strumenti inizialmente provati nessuno è più efficiente del metodo "carta e penna". Metodo immediato, semplice, di basso costo, adatto al lavoro di gruppo e scalabile, sebbene non comunichi efficacemente come gli strumenti informatici.
La nostra progettazione dei dialoghi è avvenuta inizialmente con questo metodo, ma subito si ci è accorti dei suoi punti deboli: nessun correttore di ortografia e grammatica, difficile la ricerca delle parole ma soprattutto lento nell'implementazione dei dialoghi. Tutto ciò che è scritto a mano, poi deve essere ricopiato nel sistema. Per ciò alla fine la scrittura dei dialoghi è avvenuta con il noto editor di testo Word[30]. Sulla rete molti consigliano di utilizzare i software per le mappe mentali ai fini di disegnare i dialoghi, e sicuramente dal punto di vista espressivo sono migliori di Word, ma hanno il problema di organizzare i dialoghi in modo standard, inefficiente per la successiva trascrizione nel sistema. Word, come vedremo nel prossimo capitolo, è stato davvero efficace ed efficiente nell'implementazione, risulta inoltre molto utile dal punto di vista linguistico, grazie al correttore di ortografia e grammatica, la funzione trova e sostituisci, gli elenchi puntati, ecc.

---

[27] https://arctouch.com/bot-design-ux-template/?ref=sketchhunt
[28] https://www.sketchapp.com/
[29] http://twinery.org/
[30] https://products.office.com/en-us/word

L'utilizzo di Word è stato essenzialmente quello di implementare un elenco numerato dove ogni riga coincide con un messaggio dell'utente o del bot. La figura seguente mostra il processo della trascrizione dei dialoghi.

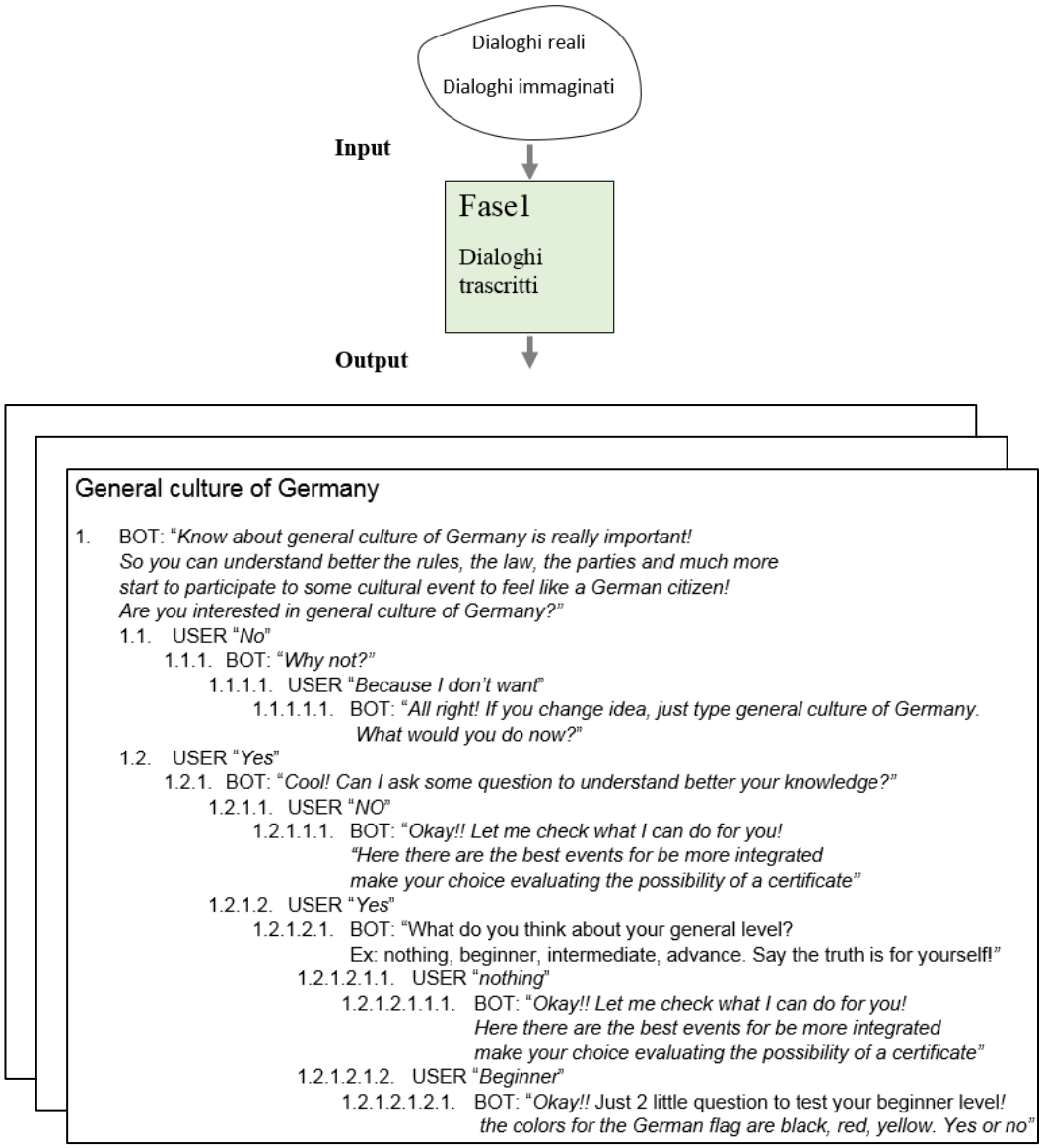


Figura 34 Fase di trascrizione dei dialoghi con in ingresso dei dialoghi in qualsiasi formato, audio, video, testo, vignette, idee, ecc. ed in uscita abbiamo i file di testo con gli alberi conversazionali mappati su elenchi numerati.

3.6 II - Progetto di dati

Oggi ci sono varie soluzioni per memorizzare i dati, a partire dai più comuni DBMS relazionali ai recenti NOSQL DBMS. A livello progettuale è molto utile utilizzare la metodologia adatta alla soluzione adottata.

I nostri requisiti sui dati mostrano due richieste chiave ai fini della nostra scelta:

1) Il database deve memorizzare delle meta-informazioni che non hanno una struttura fissa relative alle conversazioni.
2) I percorsi di formazione sono essenzialmente dei grafi, dove i passi rappresentano i nodi.

Per rispettare la prima richiesta, i DBMS relazionali non sarebbero ottimali in quanto è molto difficile modellare entità che non hanno una struttura fissa e richiederebbe molto sforzo a livello applicativo (si dovrebbe usare il tipo blob).

La seconda richiesta invece ha bisogno di una soluzione per gestire dei dati connessi. Si potrebbero utilizzare anche più DBMS, in base ai tipi di dati da raccogliere con i vari pro e contro dell'interazione fra i database. Ad esempio, i dati del profilo fb, sono strutturati in modo ottimale per un DBMS relazionale, mentre i dati relativi alle conversazioni si adattano meglio ad un DBMS a grafo.

Molto importante è anche la gestione del sistema di raccomandazione, non ben definito nelle specifiche di progetto, e la possibilità futura di memorizzare i dati degli utenti ai fini di migliorare le raccomandazioni.

Tutte queste osservazioni ci hanno portato a scegliere un DBMS a grafo.

3.6.1 Modellazione di un grafo

In questa sezione ci occupiamo di modellare un Labeled Property Graph secondo la metodologia più accreditata, descritta anche sul sito ufficiale di Neo4j[31], che permette di ottenere un grafo predisposto ad essere interrogato in Cypher. [23]
Parte della modellazione è stata tratta dal *Graphgists organization learning,* cioè un esempio di caso d'uso di Neo4j.[36]

1) **Obiettivi:** Descriviamo gli obiettivi del cliente o dell'utente finale che motivano il nostro modello.

   In questa fase abbiamo trasformato i requisiti generali sui dati nell'ottica dell'utente finale, il rifugiato. La seguente modellazione è da considerarsi focalizzata sui dati ed alcuni concetti vengono modificati rispetto al funzionamento reale di tutto il sistema. Ad esempio un rifugiato la prima volta che utilizzerà il sistema non chiederà esplicitamente di avere un percorso di formazione personalizzato, ma piuttosto un aiuto per avere un lavoro. Sarà il chatbot a spiegare che ai fini di ottenere un lavoro sono necessarie delle abilità e/o certificazioni acquisibili con determinati percorsi di formazione.

   Come rifugiato:
   - Voglio dei percorsi di formazione adatti alle mie esigenze
   - Voglio ottenere una certificazione
   - Voglio ottenere un determinato lavoro

   Dominio da modellare
   - Esistono delle categorie anche dette tematiche (es. "Tedesco", "Matematica", "Cultura Generale", ecc.) le quali possono esser suddivise

[31] Si veda https://neo4j.com/developer/guide-data-modeling/ per la versione sintetica della metodologia oppure http://www.slideshare.net/neo4j/data-modeling-with-neo4j-25767444?next_slideshow=1 per una versione illustrata della metodologia. La versione dettagliata è presente sul libro di riferimento disponibile gratuitamente in versione ebook sul sito di neo4j.

in sottocategorie a seconda della tipologia di categoria (es. Il tedesco si può suddividere in livello “base”, “intermedio” e “avanzato”)

- Sono presenti attività (es. corsi, visite guidate, ecc.) certificate e non. Ogni attività è associata a una sola categoria o sottocategoria. (es. l’attività “corso base di tedesco della biblioteca comunale” è certificata ed è associata alla sottocategoria “tedesco livello base”)
- Ci sono delle certificazioni con diversi percorsi di formazione associati. Un percorso di formazione è formato da una sequenza ordinata di elementi di formazione. (es. la certificazione A può essere raggiunta con due percorsi di formazione LP1 con gli elementi P1 P2 e P3, LP2 con gli elementi P4 e P5)
- Ogni elemento di formazione può appartenere ad una categoria o sottocategoria (es P1 appartiene alla sottocategoria “Tedesco di base”).
- Un elemento di formazione coincide con un insieme di attività certificate che appartengo alla sua stessa categoria o sottocategoria (es. P1 contiene le attività “corso base di tedesco della biblioteca comunale” e “corso base di tedesco dell’associazione abc”) oppure ad un'unica attività specificata.
- Per ottenere un certificato è necessario scegliere un percorso di formazione fra quelli offerti e completare ogni elemento di formazione svolgendo una delle attività in esso contenute.
- Sono presenti anche dei lavori che necessitano dei certificati per essere svolti.

Le seguenti due figure illustrano concettualmente il dominio appena descritto.

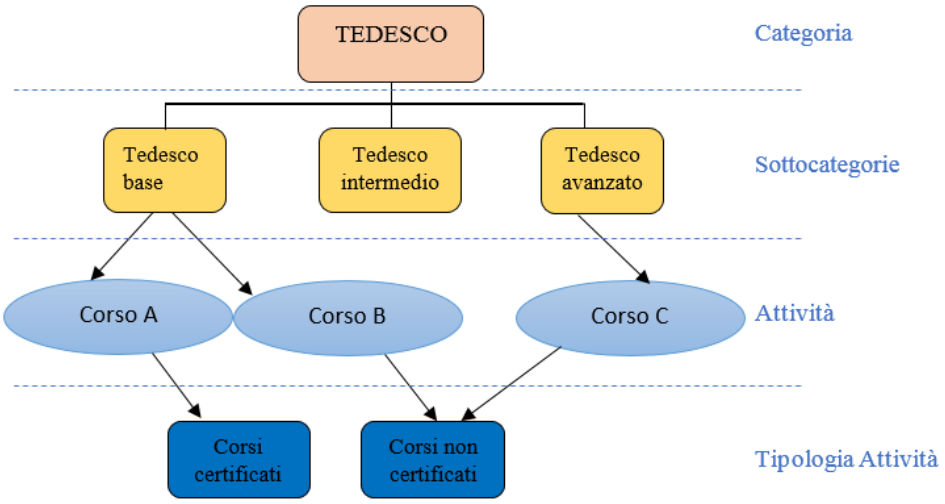


Figura 35 Esempio della categoria tedesco suddivisa in tre sottocategorie le quali possono avere dei corsi certificati e non.

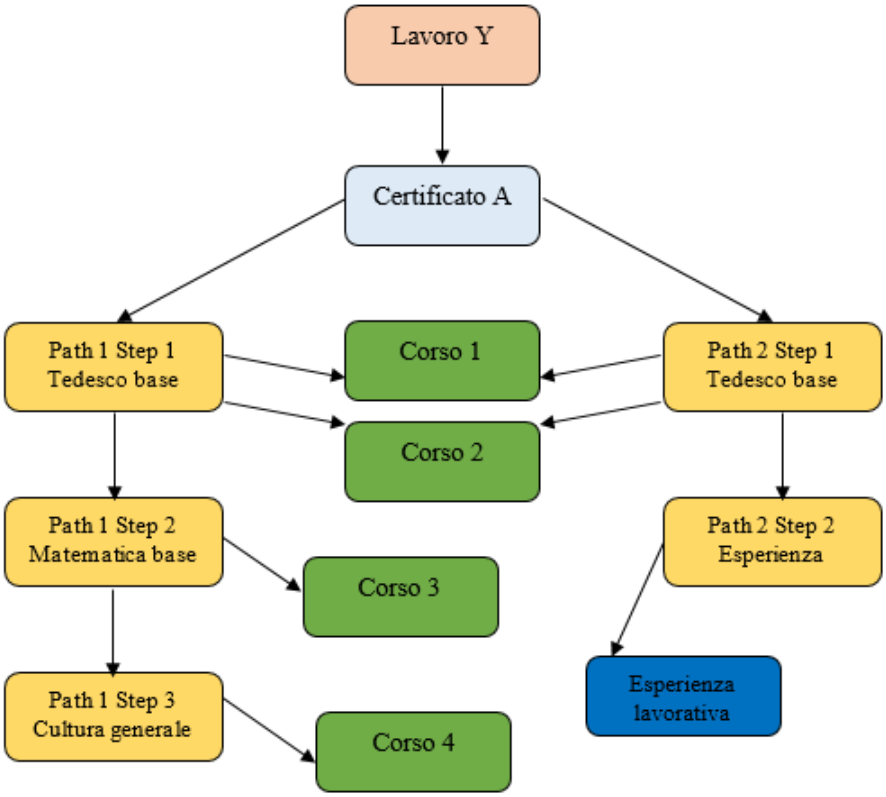


Figura 36 Esempio concettuale del dominio. Il lavoro Y richiede il certificato A il quale può essere ottenuto percorrendo uno dei due percorsi e superando una sola attività collegata ad ogni passo.

Esclusioni dal modello

Le abilità dei rifugiati, il costo delle attività, re-certificazioni, istituzioni, insegnanti, ecc. vengono esclusi dalla modellazione del prototipo.

2) **Domande:** Riscriviamo gli obiettivi come domande da chiedere al nostro dominio

Come rifugiato:

1) Quali sono le attività delle categorie che rispecchiano le mie esigenze?
2) Quali sono i percorsi di formazione per ottenere il certificato X?
3) Vorrei lavorare per la posizione Y, quali sono le certificazioni richieste?

3) **Entità e Relazioni**: Identifichiamo le entità e le relazioni che appaiono nelle domande e nel dominio descritto.

Entità:

Certificazione, Elemento di formazione, Categoria, Sottocategoria, Lavoro con l'attributo nome.

Attività, con l'attributo nome e tipologia che può essere certificata o non.

Relazioni:

Una certificazione ha uno o più percorsi di formazione.

Un percorso di formazione è costituito da uno o più elementi di formazione ordinati.

Un elemento di formazione può appartenere ad una sola categoria o sottocategoria.

Se appartiene ad una sottocategoria allora appartiene anche alla categoria della sottocategoria.

Un elemento di formazione può corrispondere ad un'attività specifica.

Una sottocategoria appartiene a una sola categoria.

Un'attività appartiene a una sola categoria o sottocategoria.

Un lavoro richiede uno o più certificazioni.

Dal momento che una certificazione può avere più percorsi formati da più elementi è necessario identificare quali elementi costituiscono ogni percorso e in che ordine, per cui il percorso di formazione è modellato come una catena di elementi collegati fra loro con la relazione “successivo", la quale deve essere qualificata con il nome del percorso di formazione a cui appartiene. La relazione “successivo” verrà utilizzata per esprimere la relazione fra certificazione e percorso di formazione, indicando la relazione fra certificazione ed il primo elemento di formazione. Questo per ottenere interrogazioni più semplici.

4) **Traduzione:** Traduciamo le entità e le relazioni in espressioni in linguaggio Cypher. In questa fase utilizzando Cypher abbiamo descritto il nostro modello in termini di nodi, etichette, relazioni e proprietà.

Espressioni Cypher:
(:Certificato) - [:SUCCESSIVO] - > (:Elemento)
(:Elemento) - [:SUCCESSIVO] - > (:Elemento)
(:Elemento) - [:APPARTIENE] - > (:Categoria)
(:Elemento) - [:APPARTIENE] - > (:Sottocategoria)
(:Elemento) - [:CORRISPONDE] - > (:Attività)
(:Sottocategoria) - [:DELLA] - > (:Categoria)
(:Attività) - [:HA] - > (:Categoria)
(:Attività) - [:HA] - > (:Sottocategoria)
(:Lavoro) - [:RICHIEDE] - > (:Certificato)

Etichette dei Nodi:
(:Certificato) (:Elemento) (:Attività) (:Certificata) (:Categoria) (:Sottocategoria) (:Lavoro)
Tipi di relazioni:
[:SUCCESSIVO] [:APPARTIENE] [:DELLA] [:HA] [:RICHIEDE] [:CORRISPONDE]

Nella seguente figura è possibile visualizzare graficamente le nostre espressioni.

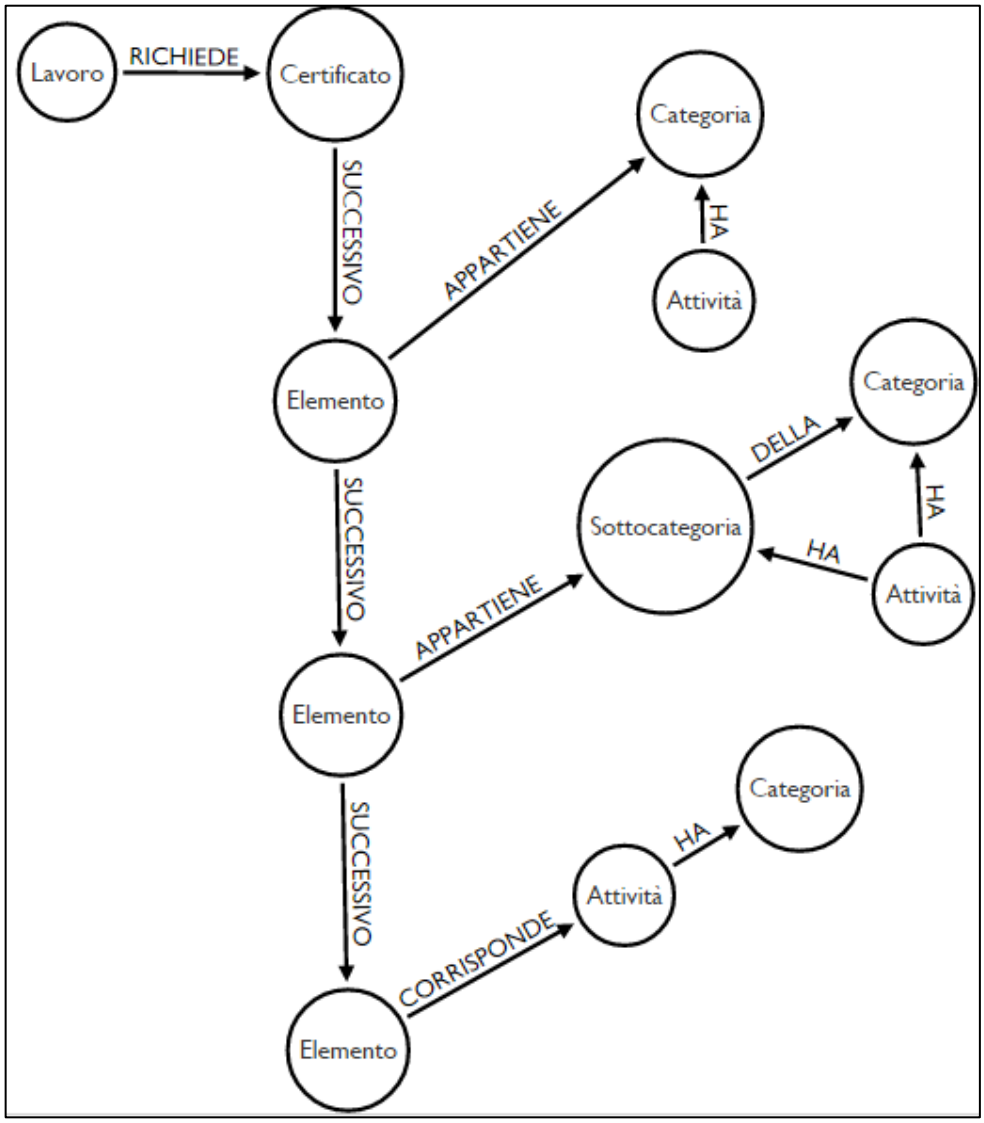


Figura 37 Visualizzazione grafica del codice Cypher che descrive il nostro modello.

Il grafico è stato realizzato con una applicazione web[32] davvero efficace. Permette di creare velocemente un grafo senza scrivere codice, molto intuitivo, non richiede registrazioni e permette di esportare il grafo in vari formati, fra cui il Cypher eseguibile direttamente nella console online di Neo4j[33].

[32] Tool per disegnare grafi http://www.apcjones.com/arrows/#
[33] Console online Neo4j http://console.neo4j.org/

La Figura 37 rappresenta un grafo con un insieme di nodi con all'interno il nome dell'etichetta associata e le relazioni con il loro nome e la loro direzione.

L'utilizzo dei simboli grafici è diverso rispetto a quello utilizzato nella visualizzazione della console di Neo4j in quanto questo grafo ha uno scopo prettamente illustrativo, ottimo per presentazioni.

Nelle successive figure è possibile vedere un esempio specifico di un'istanza del database disegnato rispettivamente con il tool grafico e la console di Neo4j dove è possibile vedere la differenza di notazione usata fra i due strumenti.

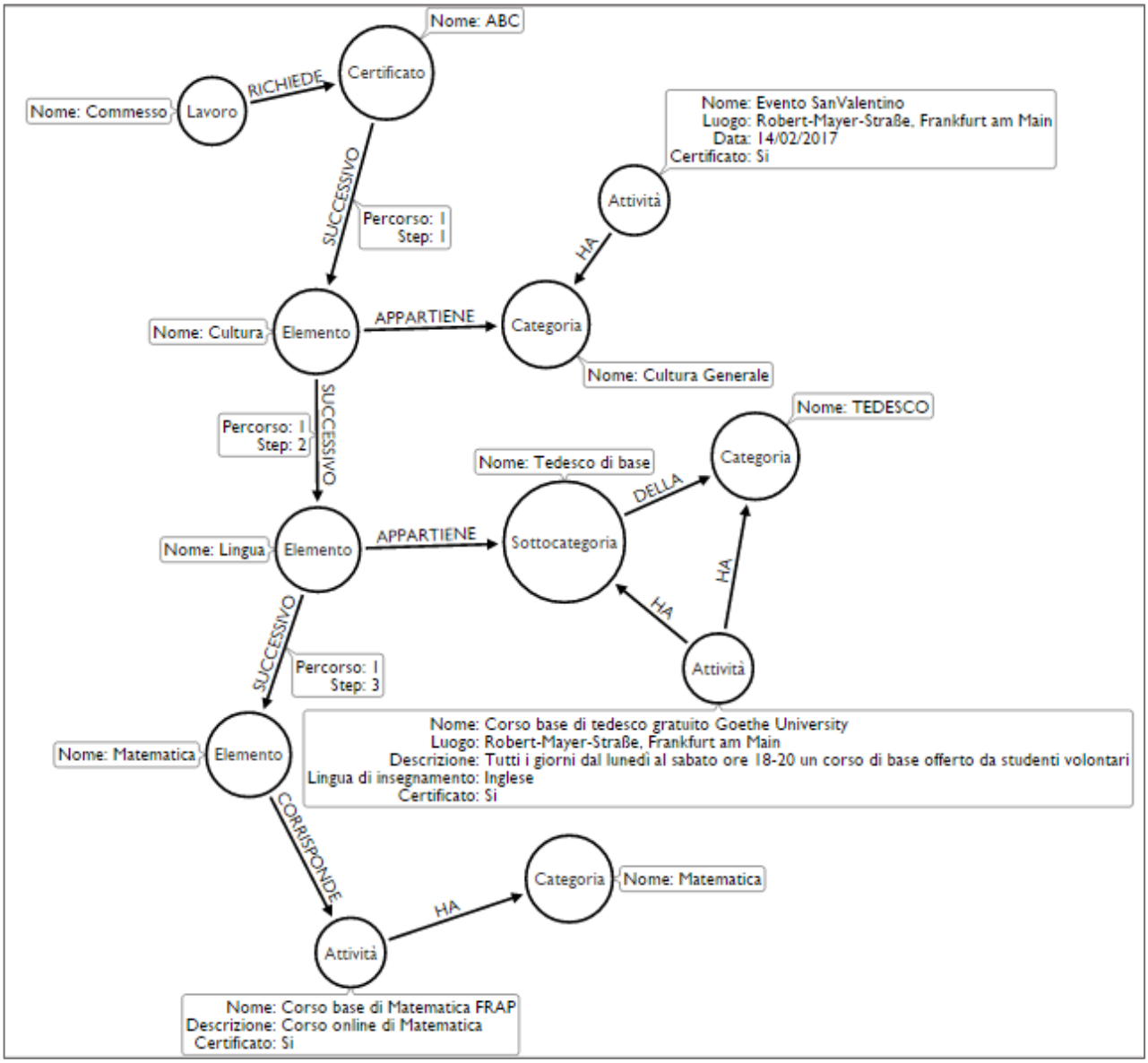


Figura 38 Visualizzazione grafica di un esempio specifico effettuata con il tool grafico.

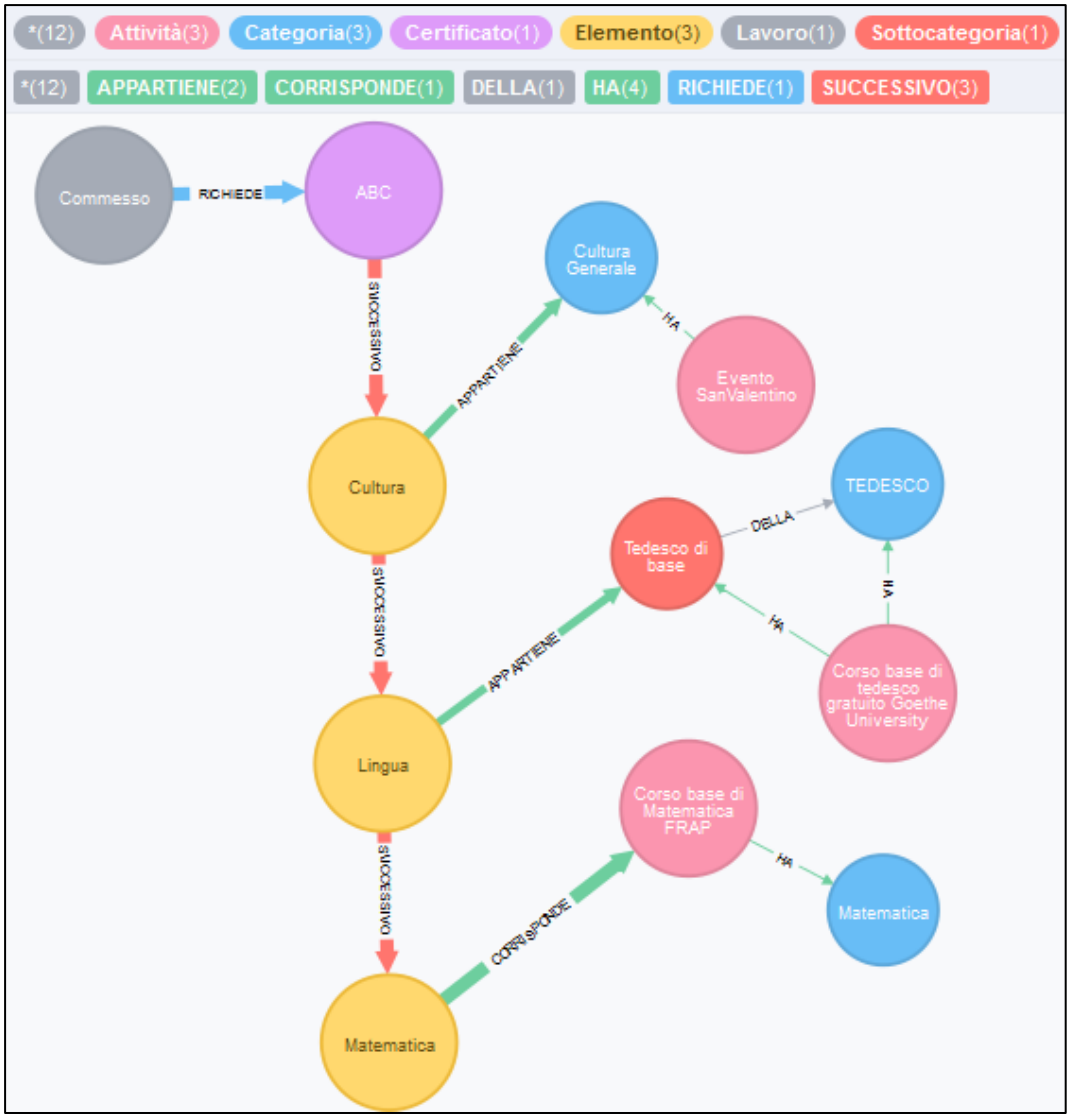


Figura 39 Visualizzazione grafica dell'esempio precedente con la console di Neo4j.

Come è possibile notare da quest'ultima figura, la console di Neo4j aggiunge dei colori per identificare meglio i nodi con le stesse etichette ed inoltre mostra il valore di una proprietà all'interno di ogni nodo e non l'etichetta come nel tool grafico. Ciò porta ad avere difficoltà nella lettura delle relazioni se esse sono espresse in base alle etichette e non alle istanze vere e proprie (Es. Tedesco di base della Tedesco).

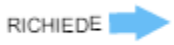
Di seguito vengono riportate due tabelle per descrivere i nodi e le relazioni utilizzate, con le varie etichette e tipi di relazioni, si tralasciano le proprietà.

Viene inserita una loro descrizione astratta affinché lo stesso schema può essere riutilizzato per rappresentare altre realtà di interesse simili.
I colori e le proporzioni dei nodi e delle relazioni sono gli stessi definiti nel foglio di stile presente in appendice D. I concetti di astrazione sono formulati sulle primitive di astrazione: classificazione, aggregazione, generalizzazione.[31]

Nodo	Simbolo	Definizione	Astrazione
	**N1**	Lavoro che necessita di certificati per essere svolto. **Etichette:** Lavoro **Relazioni in entrata:** **Relazioni in uscita:** R1 per associare un certificato, N2.	Classificazione: Classe di elementi del tipo N2.
	**N2**	Certificato che può ottenuto sostenendo dei percorsi di formazione composti da elementi di formazione, N3. **Etichette:** Certificato **Relazioni in entrata:** R1 è la relazione che lo associa ad un lavoro, N1, **Relazioni in uscita:** R2, che collega ogni primo elemento di formazione, N3, dei percorsi di formazione associati ad esso.	Classificazione: Classe di sequenze ordinate di elementi del tipo N3.
	**N3**	Elemento di formazione che può appartenere ad una categoria, sottocategoria. Un elemento di formazione coincide con un insieme di attività certificate che appartengo alla sua stessa categoria o sottocategoria oppure ad un'unica attività specificata. **Etichette:** Elemento **Relazioni in entrata:** R2 collegano al precedente elemento di formazione, N3, o in caso che l'elemento in considerazione è il primo passo del percorso di formazione allora verrà collegato a un certificato, N2. **Relazioni in uscita:** R3 per indicare l'appartenenza a una categoria, N4, o a una sottocategoria, N5. R4 per indicare la corrispondenza di N3 con un'attività N6.	Classificazione: Classe di elementi del tipo N6.
	**N4**	Categorie, anche dette tematiche servono a classificare le attività, Possono essere suddivise in sottocategorie. **Etichette:** Categoria **Relazioni in entrata:** R5 per collegare le attività N6. R6 per collegare le sottocategorie N5. R3 per collegare gli elementi di formazione N3. **Relazioni in uscita:**	Generalizzazione: Classe padre (superclasse) di classi figlie (sottoclassi) del tipo N5. Proprietà di copertura non definita. Classificazione: Classe di elementi del tipo N6.

	**N5**	Sottocategorie, servono a classificare le attività. **Etichette:** Sottocategoria **Relazioni in entrata:** R5 per collegare le attività N6. R3 per collegare gli elementi di formazione N3. **Relazioni in uscita:** R6 per collegare la categoria N4.	Generalizzazione: Classe figlia (sottoclasse) della classe padre (superclasse) del tipo N4. Proprietà di copertura non definita. Classificazione: Classe di elementi del tipo N6.
	**N6**	Attività che possono essere certificate o non. Ciò viene indicato con l'etichetta "Certificata" Ogni attività è associata a una sola categoria o sottocategoria. Se è associata ad una sottocategoria, allora è associata anche alla categoria della sottocategoria. **Etichette:** Attività, Certificata (opzionale) **Relazioni in entrata:** R4 per collegare gli elementi di formazione N3. **Relazioni in uscita:** R5 per collegare la categoria N4 o la sottocategoria N5.	Classificazione: Elemento che può appartenere alle classi del tipo: N3 mediante R4, N4 mediante R5, N5 mediante R5.

Tabella 2 Nodi dello schema a grafo.

Relazioni	Simbolo	Definizione	Astrazione
	**R1**	Indica se un lavoro richiede un certificato per essere svolto. Un lavoro può richiedere più certificati per essere svolto. In questa modellazione i certificati richiesti sono tutti obbligatori per svolgere un dato lavoro. Ciò non permette di avere certificati equivalenti. **Configurazioni** **Nodo partenza-nodo arrivo:** N1-N2	Relazione del tipo *member_of* che unisce la classe N1 ai suoi elementi N2.
	**R2**	Collega gli elementi di formazione per avere una sequenza ordinata. Per motivi implementativi, viene utilizzata anche per collegare un certificato all'inizio dei percorsi di formazione. Ogni elemento può avere un solo elemento precedente ed un solo elemento successivo. Un certificato può avere più percorsi di formazione. **Configurazioni** **Nodo partenza-nodo arrivo:** N2-N3 N3-N3	Relazione del tipo "*next*" che dato un elemento di partenza, indica il suo elemento successivo, elemento di arrivo, in una lista di elementi di tipo N3. La stessa relazione viene utilizzata per esprimere un relazione del tipo *member_of* in cui la classe è di tipo N2 e i suoi elementi sono di tipo N3.
	**R3**	Gestisce l'appartenenza di un elemento di formazione ad una categoria o sottocategoria. Un elemento può appartenere a una sola categoria o sottocategoria ma non entrambe. Se è presente questa relazione non può essere presente la relazione R4 di corrispondenza **Configurazioni** **Nodo partenza-nodo arrivo:** N3-N4 N3-N5	Relazione del tipo *member_of* che unisce la classe N3 ai suoi elementi N6 classificati con le classi N4 ed N5.

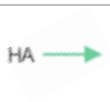
	**R4**	Gestisce la corrispondenza fra un elemento di formazione ed una attività specifica. Un elemento può corrispondere a una sola attività. Se è presente questa relazione non può essere presente la relazione R3 di appartenenza. **Configurazioni** **Nodo partenza-nodo arrivo:** N3-N6	Relazione del tipo *member_of* che associa la classe N3 ad un unico elemento N6
	**R5**	Gestisce l'appartenenza di un'attività ad una categoria o sottocategoria. Un'attività può essere associata ad una sola categoria o sottocategoria. **Configurazioni** **Nodo partenza-nodo arrivo:** N6-N4 N6-N5	Relazione del tipo *member_of* in cui gli elementi sono N6 e le classi possono essere di tipo N5 e N4.
	**R6**	Gestisce l'appartenenza di una sottocategoria a una categoria. Una sottocategoria appartiene ad una sola categoria. **Configurazioni** **Nodo partenza-nodo arrivo:** N5-N4	Relazione di *generalizzazione* fra la classe padre N4, e le classi figlie N5.

Tabella 3 Relazioni dello schema a grafo.

5) **Pattern:** Esprimiamo le precedenti domande come dei pattern grafici.

1) Quali sono le attività delle categorie che rispecchiano le mie esigenze?
   Questa domanda si può riformulare come: quali sono le attività di una determinata categoria o di una determinata sottocategoria?

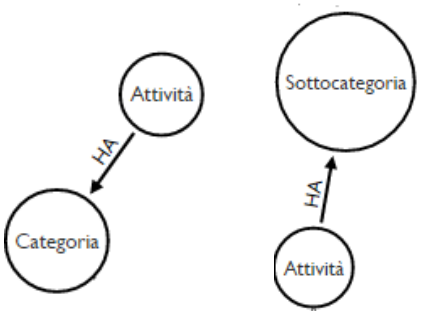


Figura 40 Pattern relativi alla domanda 1.

2) Quali sono i percorsi di formazione per ottenere il certificato X?

Figura 41 Pattern relativo alla domanda 2.

3) Vorrei lavorare per la posizione Y quali sono le certificazioni richieste?

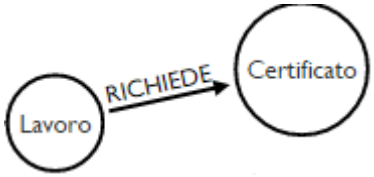


Figura 42 Pattern relativo alla domanda 3.

3.6.2 Modifiche del grafo

La metodologia di progettazione seguita, tuttavia, non prende in considerazione le operazioni che modificano la struttura del grafo. Se da un lato un database a grafo permette di inserire elementi senza problemi, dall'altro, la modellazione del grafo dovrebbe considerare le operazioni di aggiornamento (inserimento, modifica e cancellazione) e non solo le interrogazioni. Seppure Ibot è focalizzato sull'interazione fra un database a grafo ed un chatbot riteniamo che sia importante introdurre dei concetti di base per gestire l'evoluzione del database che in futuro dovrà essere gestita da un'applicazione. Questa tematica è stata affrontata cercando di esprimere i concetti ad alto livello utilizzando una analogia con i database ad oggetti.[37]

Di seguito, mostriamo una lista di possibili aggiornamenti di base per modificare la struttura di uno schema a grafo. Possiamo suddividere gli aggiornamenti in tre categorie: aggiornamenti che modificano la struttura di un nodo, la struttura di una relazione, e nodi e relazioni nel complesso.

1. Cambiamenti nella struttura di un nodo

Siccome un nodo può essere arbitrariamente complesso ci sono diversi modi per cambiare la struttura di un nodo. Possiamo pensare un aggiornamento $\boldsymbol{a}$, il quale modifica la struttura $\boldsymbol{S}$ del nodo $\boldsymbol{N}$, come un mapping tra strutture, $\boldsymbol{a: S \rightarrow S'}$. Queste modifiche si possono dividere in due categorie: quelle per cui a:
$\boldsymbol{S' \leq S}$ (che chiameremo a struttura conservata) e quelli $\boldsymbol{S' \not\leq S}$ (che chiameremo a struttura non conservata). Di tutta la lista dei possibili tipi di modifiche riportiamo sono quelli più di base:

1.1. Aggiungere una proprietà (chiave e valore) ad un nodo.
1.2. Eliminare una delle proprietà all'interno del nodo
1.3. Cambiare il nome di una proprietà (della chiave)
1.4. Cambiare il tipo di una proprietà (del valore)
1.5. Aggiungere un'etichetta ad un nodo.
1.6. Eliminare un'etichetta da un nodo.

Gli aggiornamenti di tipo *1.1, 1.3, 1.5* sono a struttura conservata; aggiornamenti *1.2, 1.4, 1.6* sono a struttura non conservata.

2. Cambiamenti nella struttura di una relazione

Anche la struttura di una relazione può essere arbitrariamente complessa. Come visto per i nodi vi sono diversi modi per cambiarne la sua struttura.

Tipi di modifiche principali:

2.1. Aggiungere una proprietà ad una relazione.
2.2. Eliminare una delle proprietà all'interno di una relazione
2.3. Cambiare il nome di una proprietà
2.4. Cambiare il tipo di una proprietà

Gli aggiornamenti di tipo *2.1* e *2.3* sono a struttura conservata; aggiornamenti *2.2* e *2.4* sono a struttura non conservata.

3. Cambiamenti nella struttura composta da nodi e relazioni
   3.1. Inserimento di nodo
   3.2. Inserimento di una relazione
   3.3. Rimozione di un nodo
   3.4. Rimozione di una relazione

È possibile estendere o ridurre questa lista dal punto di vista teorico inserendo o rimuovendo modifiche che si possono esprimere come sequenze di alcuni aggiornamenti di base.

Esempio di riduzione

> 1.3 = <1.1, 1.2> Per cambiare il nome di una proprietà si può aggiungere una nuova proprietà con un nuovo nome di chiave e lo stesso valore precedente, successivamente eliminare la proprietà relativa.

Esempio di estensione

> Cambiare la tipologia di una relazione = 4.1 = <3.2, 3.4>

Una volta definita una lista definitiva mappabile con il linguaggio Cypher, si può procedere analizzando quali saranno le modifiche al nostro modello a grafo in termini delle operazioni appena descritte. Ad esempio:

1) Inserimento di un'attività,
    a. inserire il nodo etichettato attività
    b. inserire tutte le relazioni di tipo ha alle eventuali categorie o sottocategorie

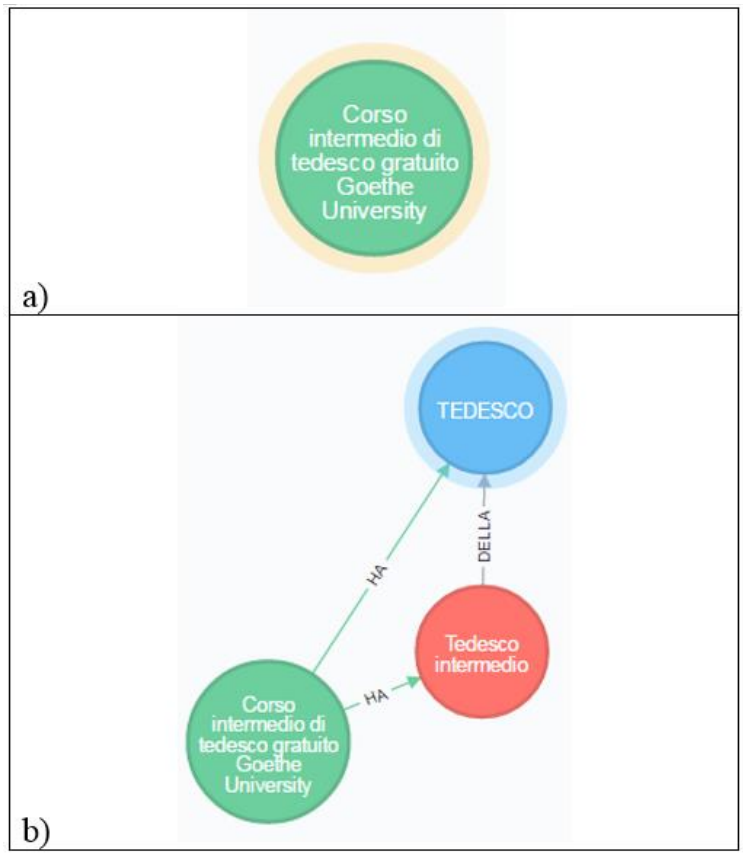


Figura 43 Esempio di inserimento di una nuova attività all'interno del grafo.

2) Inserimento di una sottocategoria
    a. Inserire il nodo etichettato sottocategoria
    b. Inserire la relazione di tipo della con la categoria
    c. Inserire le relazioni di tipo ha con tutte le attività appartenenti ad essa
3) Ecc..

In fine dovremmo ottenere un modello di grafo che permette di eseguire non solo le interrogazioni richieste ma anche i vari aggiornamenti richiesti. Sarà necessario analizzare i requisiti specifici delle singole operazioni e cercare le soluzioni migliori, tuttavia questa tesi non analizzerà ulteriormente questo aspetto perché non rilevante ai fini del prototipo.

3.7 III – Progetto Interazione

Questa fase si occupa di descrivere l'interazione tra il progetto dei dialoghi e dei dati necessaria affinché il sistema rispetti i requisiti funzionali. È da ritenersi il contributo davvero innovativo di questa tesi.

3.7.1 Modellare l'interazione

L'interazione è basata su alcune meta-informazioni memorizzate nel database relative ai dialoghi. La loro struttura nel nostro modello è in funzione della tipologia di blocco conversazionale. Ogni blocco della conversazione può avere delle meta-informazioni associate che si occupano di eseguire determinati compiti. Ad esempio il blocco che si occupa dei saluti non ha nessuna meta-informazione associata, mentre i blocchi che valutano le abilità linguistiche o matematiche hanno varie meta-informazioni fra cui le soluzioni delle domande a quiz effettuate.
L'interazione può essere vista come un insieme di 3 elementi comunicanti. Un blocco conversazionale, un nodo del grafo contenente tutte le meta-informazioni e una parte di codice che viene attivata per gestire un determinato blocco conversazionale. Lo schema seguente mostra il flusso completo del funzionamento di tutto il sistema. Tale flusso è composto da 6 fasi dove l'output di ogni fase coincide con l'input della fase seguente:

1) L'utente dialoga con il sistema attraversando i vari blocchi finché arriverà ad un blocco che necessita informazioni dal database.
   Output:
   Lista di parametri forniti dall'utente durante la conversazione e parametri interni dovuti alla gestione della conversazione (contesti, azioni, intenti, ecc.).
2) L'applicazione riceve i parametri della fase precedente e li elabora secondo la parte di codice specifica alla gestione del blocco che ha generato la richiesta.
   Output:
   Query sul database per ottenere le meta-informazioni del blocco in gestione.
3) Il Graph DBMS esegue la query ricevuta.
   Output:
   Risultato della query eseguita.

4) L'applicazione riceve le meta-informazioni necessarie e le elabora secondo la parte di codice specifica alla gestione del blocco.
   Output:
   Query sul database per ottenere le informazioni richieste dal blocco in gestione.
5) Il Graph DBMS esegue la query ricevuta.
   Output:
   Risultato della query eseguita.
6) L'applicazione riceve le informazioni richieste dal blocco, le elabora e le invia all'utente.
   Output:
   Messaggio di risposta.

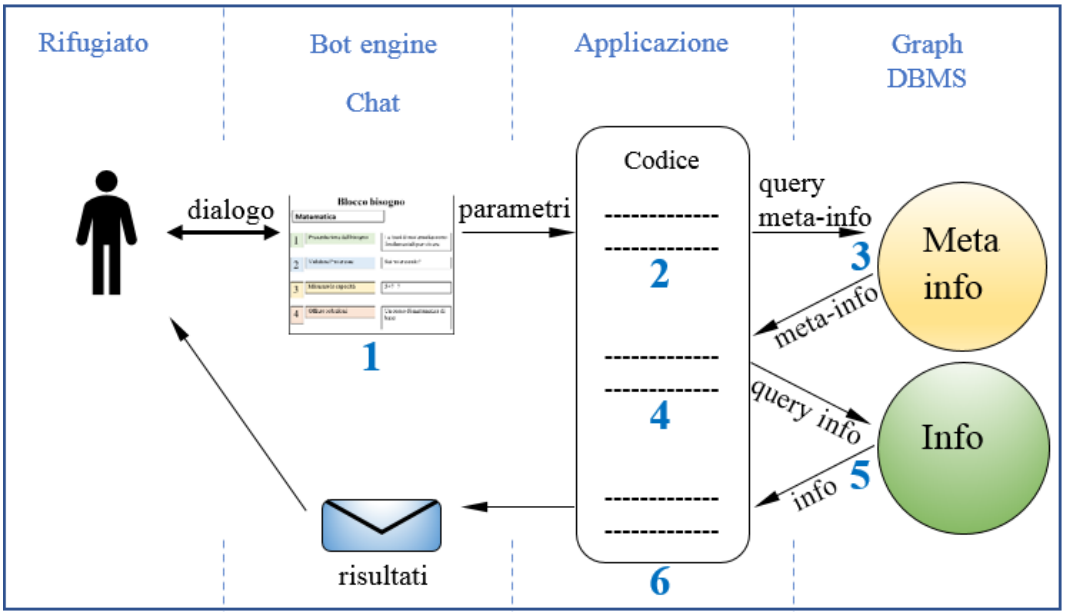


Figura 44 Flusso di funzionamento del sistema mappato sull'architettura a 3 livelli scelta diviso in 6 fasi sequenziali.

3.7.2 Modellare i dialoghi per l'interazione

I dialoghi, al fine di poter comunicare al sistema i parametri necessari per effettuare le richieste al database, dovranno essere progettati considerando non solo i testi ma anche la logica relativa ad essi. Questa fase può essere effettuata senza conoscenze informatiche, ma è strettamente necessaria per la fase dell'implementazione in quanto non sempre è

possibile ricavare la logica partendo dal testo. Ad esempio, se ci sono delle domande quiz bisogna descrivere come devono essere valutate le soluzioni, oltre ad includere le soluzioni stesse.

La logica che verrà implementata nell'applicazione fra la chat e il database, può essere descritta in qualsiasi forma, diagrammi di flusso, tabelle, testi, purché comprensibile nella fase dell'implementazione.

3.7.3 Modellare il grafo per l'interazione

Fin ora, lo schema a grafo è stato progettato non considerando i dialoghi e il loro modello a blocchi. Il collegamento diretto che avviene fra i due progetti consiste nell'associare un nodo di tipo "categoria" a ogni blocco conversazionale che ne abbia bisogno.

Una volta che tali blocchi sono documentati con il loro modo di funzionare e i loro dati di supporto, meta-informazioni, è necessario aggiungere quest'ultimi al database.

Le meta-informazioni posso essere memorizzate semplicemente come proprietà del nodo "categoria" relativo, sebbene in caso di complessità elevate è possibile utilizzare strutture a grafo, grazie alla proprietà schema-free dei database a grafo. In realtà esse potrebbero anche essere incorporate nel codice dell'applicazione, ciò aumenterebbe le performance risparmiando una query al database, ma successivamente ogni modifica che deve essere effettuata ad esse porterebbe a modificare il codice del sistema. Tuttavia se in fase di test l'esecuzione risultasse troppo lenta, si dovranno valutare altre soluzioni ed effettuare le modifiche opportune al progetto.

Sempre mediante la documentazione dei blocchi sarà possibile definire eventuali sottocategorie.

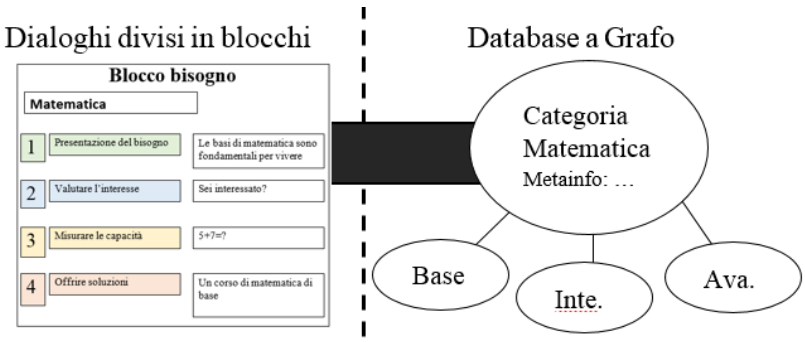


Figura 45 Interazione fra un blocco conversazionale e un nodo categoria dello schema a grafo.

<table>
<tr><th>Documentazione Dialoghi</th><th>Applicazione</th><th>Database a grafo</th></tr>
<tr><td rowspan="2">Testo blocco Matematica
Le basi della matematica sono importanti per vivere
Sei interessato?
5+7=?</td><td>Pseudocodice 1
Se blocco matematica:
leggi meta-info matematica</td><td rowspan="2">Matematica meta-informazioni
Risposta_1: 12</td></tr>
<tr><td>Pseudocodice 2
Se input_utente_1==Risposta_1:
LIVELLO = “intermedio”
altrimenti:
LIVELLO = “base”</td></tr>
<tr><td>Logica blocco Matematica
Presenta una domanda che richiede un parametro numerico intero. La soluzione corretta è 12. Se il parametro inserito è uguale alla soluzione allora l’utente è di livello intermedio altrimenti è di livello base</td><td>Pseudocodice 3
Switch(LIVELLO):
Case(“base”):
esegui query 1
Case(“intermedio”):
esegui query 2</td><td>Matematica informazioni
Un corso di base
Nessun corso intermedio</td></tr>
</table>

Figura 46 Esempio di una documentazione relativa alla logica di un blocco conversazionale, lo pseudocodice e le meta-informazioni riferite ad essa.

3.7.4 Raccomandazione: percorsi di formazione dinamici

L’interazione fra i dialoghi ed il database ci permette di realizzare il nostro requisito funzionale per quanto riguarda la raccomandazione di percorsi di formazione. Esistono diversi sistemi di raccomandazione, molti dei quali immediatamente applicabili ad un database a grafo, ma per rispettare il requisito di non memorizzare i dati degli utenti ne possiamo utilizzare solo pochi. In particolare la nostra raccomandazione è basata su due concetti:

- raccomandazione sulla logica di ogni blocco conversazionale
- raccomandazione sulla forma dei testi.

Ad esempio, il blocco che raccomanda l’apprendimento della lingua può svolgere dei quiz e consigliare i migliori corsi in base alle risposte ricevute (raccomandazione sulla logica). Inoltre, può essere inserito come un percorso obbligatorio del dialogo, magari nelle parti della conversazione dove il rifugiato ha un’attenzione maggiore (raccomandazione sui testi).

L'idea di percorsi di formazione dinamici, ovvero percorsi non già definiti nel sistema ma che vengono costruiti per ogni specifico utente, si realizza mediante l'evoluzione della conversazione. Nel dialogo per determinati blocchi conversazionali attraversati vengono suggeriti degli elementi di formazione. Il rifugiato alla fine di una conversazione visualizzerà un percorso di formazione che non è necessariamente memorizzato nel database a grafo. L'immagine seguente mostra un esempio di conversazione che dinamicamente crea un percorso di formazione per l'utente. Con le linee a puntini vengono mostrate tutte le 9 combinazioni di attività disponibili affinché il percorso possa essere completato. I blocchi sono disposti sfasati perché la conversazione si svolge percorrendo i rami di un albero conversazionale. La linea in rosso indica il percorso scelto dal rifugiato.

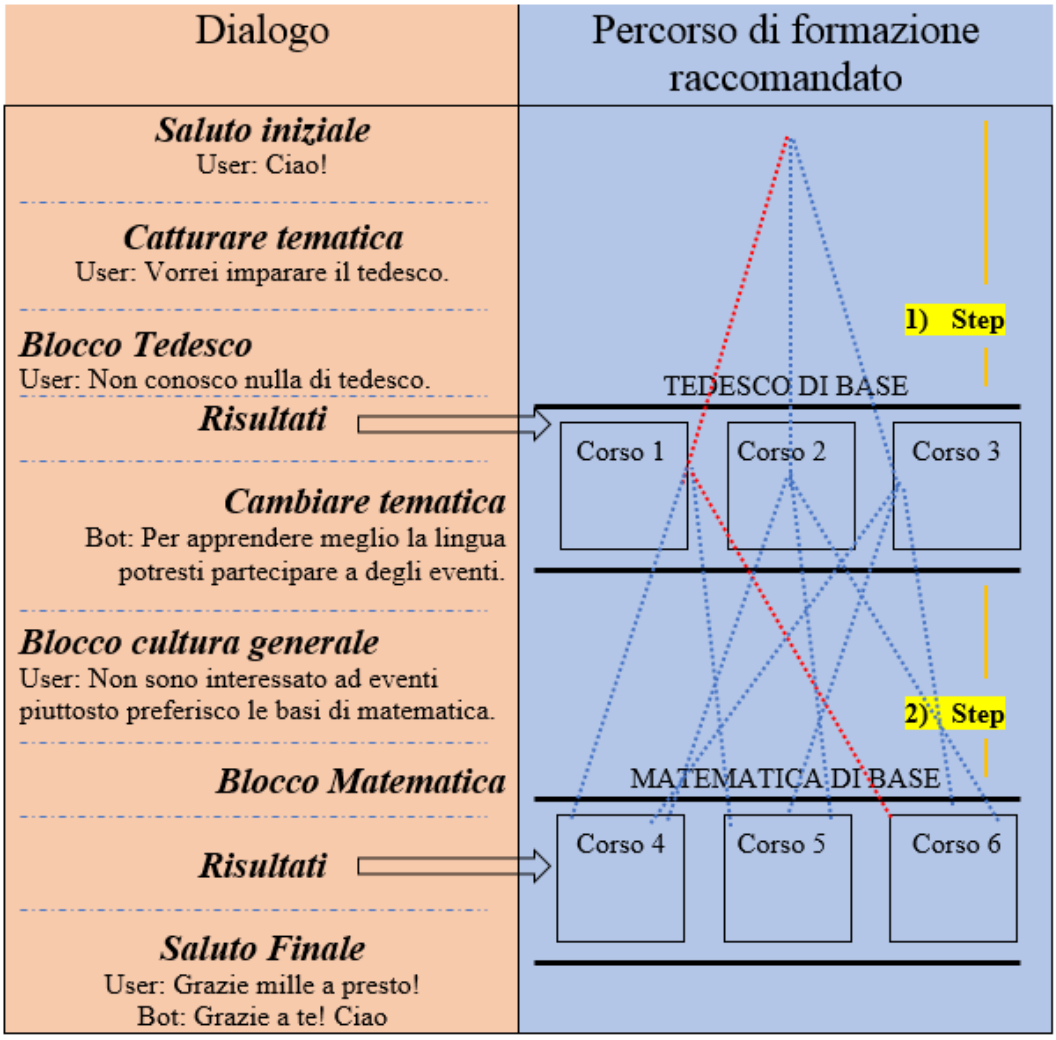


Figura 47 Esempio di un percorso di formazione dinamico creato sulla base del dialogo.

3.7.5 Evoluzione nel tempo: Gestire le modifiche.

Nel sistema ci sono due gruppi di operazioni degni di nota necessari all'evoluzione del chatbot:

- Modifiche ai dialoghi, apportate dal gestore del sistema (FRAP).
- Modifiche agli elementi del database, dovute agli erogatori di attività.

Le modifiche ai dialoghi

Esse sono meno frequenti e solitamente più complesse delle altre, comprendono tre tipi principali di operazioni:

1) Inserimento di un nuovo blocco conversazionale.
2) Modifiche ad un blocco già esistente.
3) Eliminazione di un blocco.

L'inserimento comporta l'introduzione di nuovi blocchi conversazionali all'interno del bot engine, eventuale codice nell'applicazione, eventuali strutture nel database a grafo. Per effettuare questa operazione è necessario verificare che non siano già presenti blocchi simili o uguali a quello da inserire. Dal punto di vista del dialogo bisogna valutare la possibilità che il blocco inserito possa essere effettivamente attraversato durante una conversazione. Per permettere al sistema di inserire blocchi senza causare malfunzionamenti è necessario inserire prima le strutture nel database, grazie alla flessibilità dei database a grafo questa fase non comporta problemi, successivamente si inserisce il codice nell'applicazione ed infine i dialoghi nel bot engine. L'inserimento di ogni blocco conversazionale comporta la creazione di un nodo categoria ed eventuali nodi sottocategoria. In questa fase sarebbe utile collegare eventuali attività già presenti alle nuove categorie inserite.

L'eliminazione, al contrario dell'inserimento, rimuove i blocchi conversazionali e gli elementi associati ad essi. Questa operazione prima di essere attuata deve verificare i possibili effetti collaterali sia dal punto di vista del dialogo, dove influisce su altri blocchi conversazionali, sia dal punto di vista dei dati. La sequenza di azioni corretta per eseguire l'eliminazione inizia rimuovendo i dialoghi dal bot engine e cambiando gli altri dialoghi influenzati dall'eliminazione, successivamente viene modificato il database nel modo opportuno ed infine il codice dell'applicazione. L'eliminazione di un blocco

conversazionale si riflette sul database eliminando un nodo categoria ed eventuali sottocategorie con tutte le loro relazioni. Se sono presenti delle attività collegate a quest'ultime si può decidere se eliminarle, collegarle ad altre categorie o lasciarle in sospeso finché non verrà inserito un nuovo dialogo che le possa gestire.

Ci sono varie modifiche inerenti ai blocchi conversazionali.
Alcune riguardano:

1) Solo il bot engine per la modifica dei dialoghi
   Es: correggere errori grammaticali in un testo.
   Solitamente richiedono poco tempo ed hanno un effetto immediato. Si realizzano tramite l'interfaccia grafica del bot engine.
2) Solo l'applicazione per la modifica della logica del dialogo
   Es: aggiungere un controllo dei parametri in ingresso.
   Queste modifiche possono essere molto delicate, in quanto richiedono dei cambiamenti nel codice dell'applicazione e possono causare l'instabilità di tutto il sistema.
3) Solo il database
   Es: Modificare le meta-informazioni di una categoria.
   A seconda dell'operazione sono di complessità differenti. Vengono apportate mediante delle query Cypher al database a grafo.
4) Combinazioni delle modifiche precedenti.
   Es 1 – 2: Rimuovere una domanda e il relativo codice di controllo.
   Es 1 – 3: Aggiornare una domanda a risposta multipla e le relative soluzioni presenti nelle meta-informazioni senza modificarne la logica di valutazione.
   Es 2 – 3: Semplificare un query cambiando la struttura del database.
   Es 1 – 2 – 3: Cambiare un ramo della conversazione che richiede una nuova sottocategoria e una logica per gestirla.
   Sono le modifiche più critiche in quanto influiscono su più elementi dell'architettura. Come nell'inserimento ed eliminazione di un blocco del dialogo è necessario seguire un determinato ordine delle modifiche affinché il sistema continui a funzionare regolarmente durante l'implementazione delle modifiche.

Le modifiche agli elementi del database, dovute agli erogatori di attività.

Il database dovrà permettere ad altre applicazioni di gestire le attività:

1) Inserimento
2) Modifica
3) Eliminazione.

L'inserimento consiste nella creazione di un nodo attività. È possibile inserire l'attività direttamente all'interno di una determinata categoria in modo che sia immediatamente raggiungibile tramite dei blocchi conversazionali, oppure senza nessuna categoria.

L'eliminazione equivale alla rimozione del relativo nodo.

La modifica può essere di due tipi, interna modificando le proprietà del nodo e le varie etichette, esterna modificando le relazioni del nodo, ovvero la categoria o sottocategoria di appartenenza dell'attività.

Nella figura seguente è possibile vedere come la raccomandazione delle attività varia nel tempo in base alla situazione del database, in un determinato istante, a parità di dialogo sostenuto dal rifugiato.

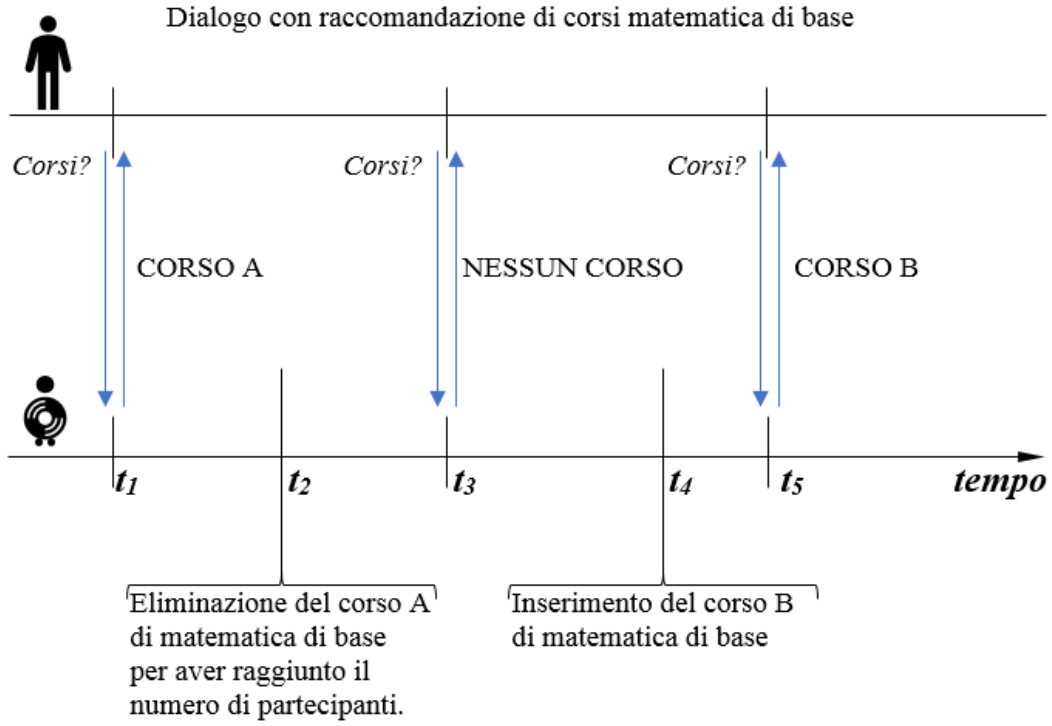


Figura 48 Evoluzione temporale della raccomandazione di attività ad un rifugiato che sostiene sempre lo stesso dialogo in diversi istanti di tempo.

Capitolo 4

Implementazione

Questo capitolo si occupa di descrivere l'implementazione del bot. Tutto ciò che è fornito dalla fase di progettazione verrà implementato con le tecnologie descritte nel capitolo 2. Come mostrato in figura seguente, dopo aver analizzato gli aspetti generali implementativi e le motivazioni delle varie scelte implementative, l'implementazione viene divisa in tre fasi, l'implementazione dei dialoghi, del database e dell'interazione fra essi.

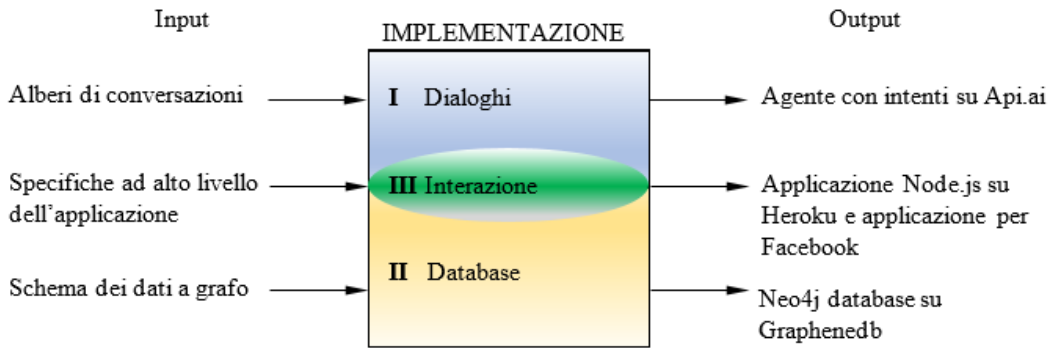


Figura 49 Fasi dell'implementazione.

4.1 Architettura implementata

Come anticipato, la quantità di tecnologie presenti sul mercato per realizzare la nostra architettura è davvero elevata. Per filtrare le possibili soluzioni abbiamo effettuato una serie di scelte. Le varie tecnologie descritte nel capitolo 2 ci hanno permesso di implementare l'architettura progettata. Nella figura seguente viene mostrata l'architettura implementata con il flusso dei messaggi semplificato rispetto a quello reale che verrà descritto in seguito.

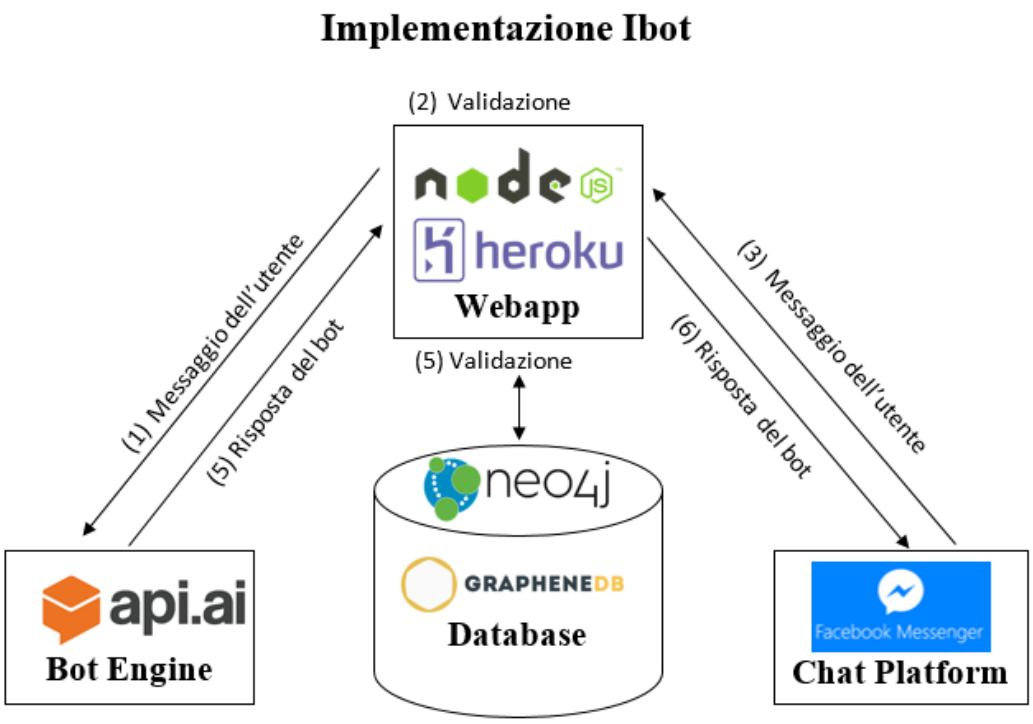


Figura 50 Architettura a 3 livelli implementata

Le principali scelte implementative sono:

- Facebook Messenger, come piattaforma di messaggistica;
- Api.ai, come servizio per elaborare il linguaggio naturale;
- Neo4j come database a grafo;
- Graphenedb come hosting per Neo4j;
- Heroku per distribuire l'applicazione online.

4.2 Motivazioni delle scelte implementative

La webapp è stata realizzata con Node.js[34] ed ospitata sui server di Heroku. Scelte legate alla velocità di realizzazione del prototipo. Tale velocità è dovuta principalmente a due fattori. Il primo è la notevole quantità di codice presente nella documentazione di Messenger Platform, Heroku, Graphenedb, Github[35], Api.ai, il secondo fattore riguarda la possibilità di ottenere immediatamente i servizi da loro offerti.

4.2.1 Scelta di Messenger

La scelta implementativa di usare Messenger e non un'altra chat è legata principalmente alla sua popolarità. Secondo i dati dell'F8:

- 900 milioni di utenti attivi su M. ogni mese (11% della popolazione mondiale).
- 50 milioni di aziende sono già presenti su M.
- 9.5 miliardi di foto vengono inviate ogni mese su M.
- Il 72% degli utenti di M. acquista online regolarmente.[12]

Moltissimi sono i rifugiati già presenti su Facebook. Per i profughi il social network svolge un ruolo molto importante, sia a livello comunicativo, che nella ricerca attiva dei membri mancanti della loro famiglia. Questa scelta ci permette di interagire con loro senza che scarichino un'apposita app, indipendentemente dal dispositivo utilizzato.

4.2.2 Scelta di API.ai

La scelta del bot engine ricadeva principalmente tra due opzioni, Wit.ai di Facebook e Api.ai di Google. In definitiva il bot engine è stato realizzato con Api.ai perché ha un'interfaccia davvero intuitiva e sembrerebbe il più popolare fra gli sviluppatori e le aziende. Api.ai ha molte caratteristiche utili ed offre la possibilità di costruire le funzionalità richieste senza necessariamente programmare. In questo prototipo è stata

[34] https://nodejs.org/it/
[35] https://github.com/

scelta la versione gratuita di Api.ai, non solo per motivazioni economiche, ma anche perché le funzionalità a noi necessarie sono tutte presenti.

4.2.3 Scelta di Neo4j

È stato scelto Neo4j come database perché risulta essere il più popolare fra i database a grafo. La sua enorme diffusione ha dato vita ad una vera e propria community che offre tantissime risorse in vari campi di applicazione.

4.2.4 Scelta di Graphenedb

Ci sono vari modi per utilizzare Neo4j in un'applicazione, sul suo sito ufficiale vi è una guida completa alle varie soluzioni cloud disponibili[36]. È stata scelta una soluzione cloud che rispettasse il requisito di sistema riferito alla possibilità del prototipo di rimanere sempre a disposizione di chiunque voglia continuare il progetto. Scelta utile anche per evitare di dover importare ed esportare i dati fra un database in locale ed uno in produzione. La nostra prima prova di Neo4j in cloud prevedeva l'utilizzo di Amazon EC2[37]. Abbiamo seguito una guida[38], ma l'esito di questo test è stato negativo; abbiamo riscontrato vari problemi, fra cui la necessità di utilizzare specifiche porte del firewall bloccate per motivi di sicurezza all'interno del nostro ambiente di lavoro e l'esaurimento rapido delle quote gratuite offerte da Amazon. Quest'ultimo motivo, in particolare, ci ha portato a valutare altre opzioni e possibili soluzioni.
Avendo scelto precedentemente di utilizzare Heroku come piattaforma per distribuire la nostra applicazione, è possibile utilizzare un componente aggiuntivo per dotare l'applicazione, di Neo4j utilizzando Graphenedb come database as a service. Seppure il componente offre una quota gratuita richiede la verifica delle credenziali utente con l'inserimento di una carta di credito. Tuttavia analizzando direttamente il sito ufficiale di Graphenedb è esplicito che si può utilizzare, nella sua versione di prova, senza carta di

[36] https://neo4j.com/developer/guide-cloud-deployment/
[37] https://aws.amazon.com/it/
[38] https://dzone.com/articles/how-deploy-neo4j-instance

credito in qualsiasi applicazione, inserendo gli opportuni driver per la comunicazione con esso. Tale implementazione verrà descritta inseguito.

4.3 Connessione del bot con Facebook e flusso dei messaggi

Per gestire la comunicazione fra il Bot-engine e la piattaforma di messaggistica, in particolare Api.ai e Facebook, è necessario configurare alcuni parametri. In una prima fase dello sviluppo dell'applicazione, durante l'implementazione dei dialoghi, è stato utile usare l'architettura semplice a due livelli. Questo ci ha permesso di testare subito i dialoghi su smartphone senza scrivere codice. Nel caso di architettura a due livelli
gli elementi per realizzare la connessione sono solo 3, una pagina fb, una applicazione fb, un agente Api.ai. La pagina rappresenta l'identità del nostro bot ed un punto di ingresso per far sì che il nostro bot venga trovato. L'applicazione fb, intesa come un'applicazione all'interno di fb, è necessaria affinché l'agente possa comunicare con Messenger Platform. La figura seguente mostra la prima architettura implementata. Nell'appendice B è possibile trovare una mini guida per implementare un chatbot con architettura semplice in pochi minuti.

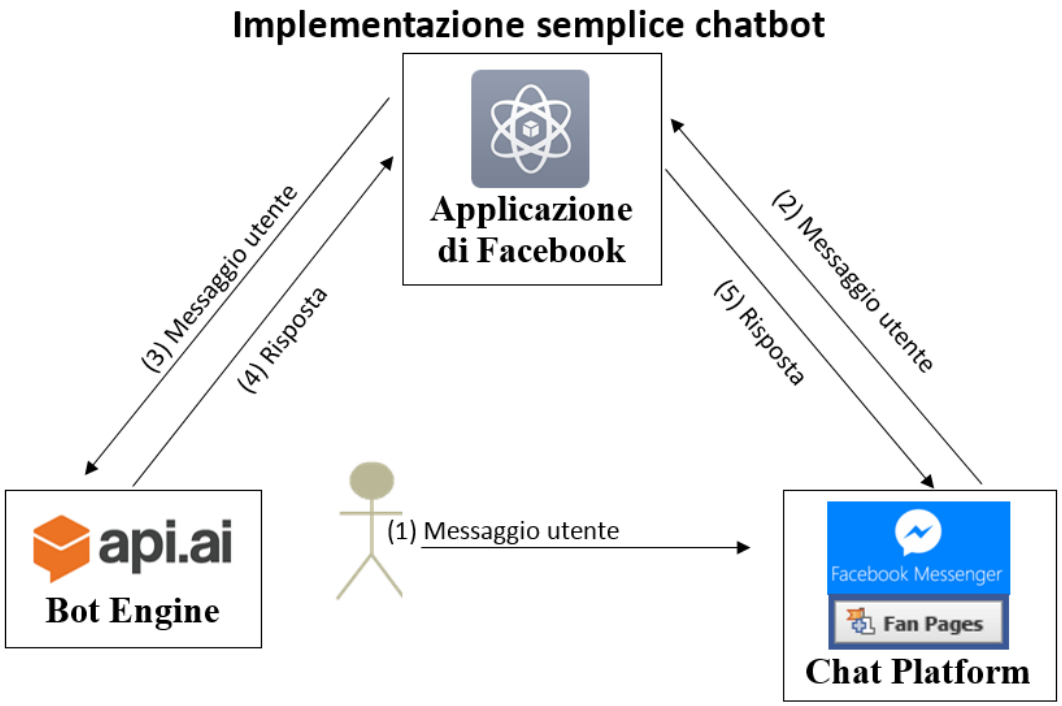


Figura 51 Prima implementazione di Ibot con architettura semplice.

Tutti i messaggi inviati dall'utente tramite Facebook o Messenger alla nostra fan page saranno inviati alla nostra applicazione fb che, a sua volta, li inoltrerà a Api.ai.

Api.ai analizzerà i dati, controllerà gli intenti, le azioni, gli eventuali parametri, se sono presenti dei contesti precedenti e genererà una delle possibili risposte per quello specifico intento. La risposta sarà inviata all'applicazione fb che la mostrerà nella chat all'utente.

Successivamente per collegare il sistema al database, con alcune modifiche all'architettura precedente, si è ottenuta l'architettura tipica a 3 livelli, definitiva per il nostro progetto. La figura seguente ne mostra tutti i componenti ed il flusso reale dei messaggi.

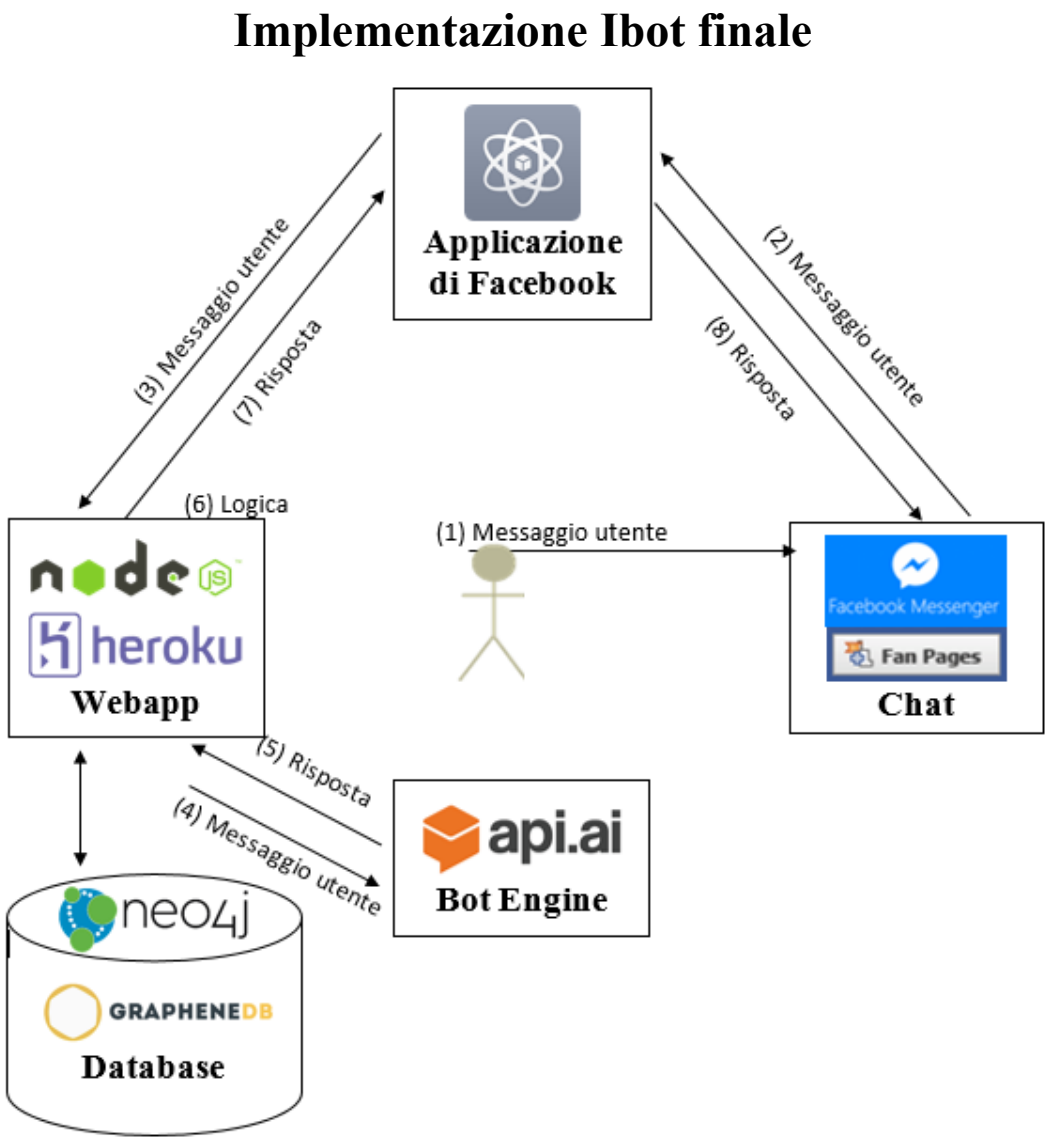


Figura 52 Implementazione finale Ibot con l'intero flusso di messaggi.

4.4 Fasi dell'implementazione

Come anticipato all'inizio del capitolo, l'implementazione viene divisa in tre fasi, l'implementazione dei dialoghi, del database e dell'interazione fra essi. La prima fase si occupa delle conversazioni che saranno gestite interamente Api.ai. La seconda fase della creazione del database a grafo Neo4j su Graphenedb. La terza fase dell'implementazione dell'applicazione in Node.js su Heroku.

4.5 I - Implementazione dei dialoghi

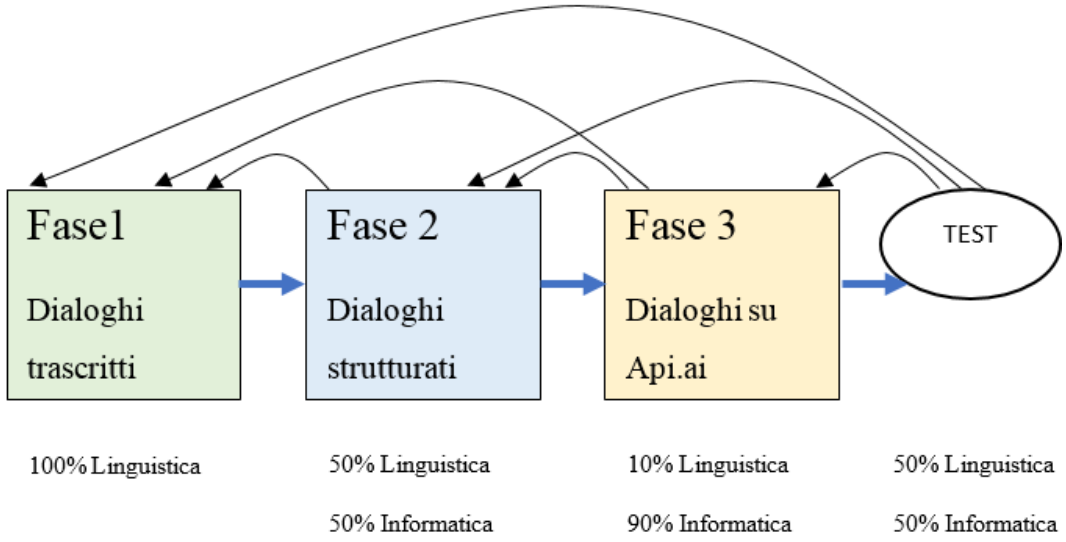


Figura 53 fasi per l'implementazione dei dialoghi su api.ai

Il nostro processo per trasformare semplici frasi in un sistema in grado di utilizzarle in modo simile ad un umano è composto da tre fasi:

1) La Fase 1 si è svolta durante la progettazione ed il suo output è stato un insieme di dialoghi, trascritti con una documentazione logica allegata. In tale fase non sono necessarie competenze informatiche e i dialoghi prodotti possono essere implementati in qualsiasi sistema che elabora il linguaggio naturale.

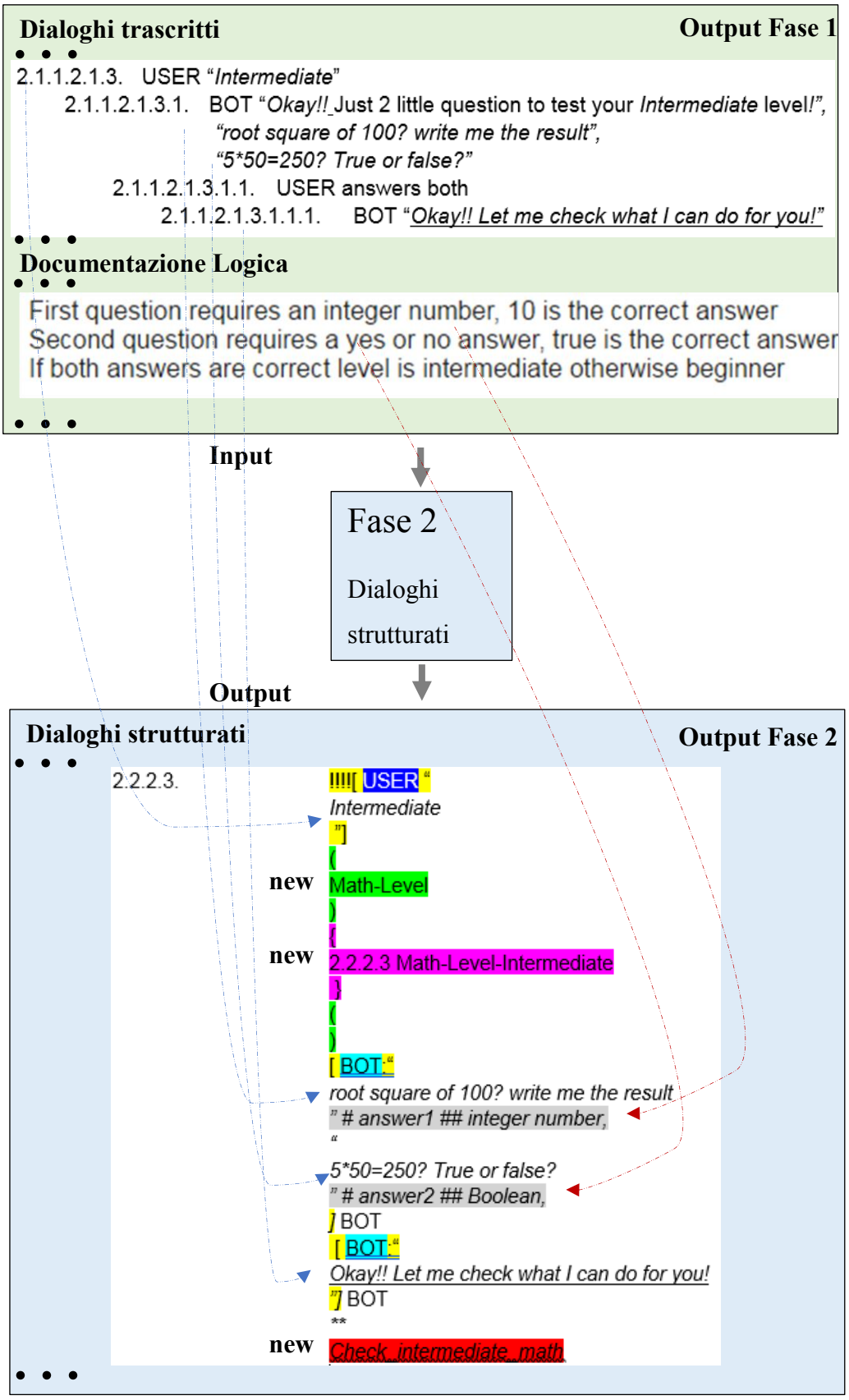


Figura 54 Esecuzione della seconda fase per implementare i dialoghi.

2) La fase 2 rappresenta la ristrutturazione del testo nella semantica del bot engine di riferimento, in questo caso Api.ai[39]. Durante questa fase una semplice frase si trasforma in intenti, contesti, parametri, azioni e risposte, ovvero gli elementi descritti precedentemente nel capitolo 2. Questa fase può sembrare superficiale in quanto si potrebbero mappare i dialoghi direttamente sulla piattaforma di Api.ai, ma in realtà è l'unica possibilità di gestire la complessità di un bot che ha conversazioni con vari rami. Per svolgere questa fase abbiamo creato una nostra "sintassi". Quest'ultima è stata modificata varie volte per ottenere una versione performante in grado di mappare la maggior parte degli elementi necessari e favorire un trasferimento veloce nella fase 3.

Descrizione della sintassi

Per ogni intento si ha un punto dell'elenco numerato contenente una struttura del tipo:

X.X.X.X !!!! [USER " *Cosa dice l'utente* "]/ Sinonimo di ciò che dice l'utente / altro sinonimo (Contesto-Entrata) { X.X.X.X.Nome-intento } (Contesto-1-uscita X, Contesto-2-uscita X) [BOT: "Cosa dice il bot*" # nome-parametro-1 ## tipo parametro 1,"* Cosa dice il bot *" nome-parametro-2 ## tipo parametro 2,]* BOT [BOT:"*Cosa dice alla fine il bot"]* BOT**Nome-azione*

Il carattere "X" rappresenta un numero intero qualsiasi. La sintassi è sequenziale cioè ogni elemento segue l'ordine presentato nella struttura. L'indentazione non è riportata, ma segue la regola generale che tutte le frasi da ricopiare devono essere riportate su un'unica riga senza elementi di contorno come asterischi, parentesi, ecc. La sottolineatura è opzionale, normalmente indica quando una conversazione arriva ad una foglia, ovvero dove termina un ramo di un albero conversazionale.

[39] In realtà la semantica dei vari bot engine è molto simile fra loro se non pressoché uguale, per cui l'output della fase 2 non è esclusiva per Api.ai

Nella seguente descrizione indichiamo con "+" gli elementi della sintassi obbligatori, con "–" quelli opzionali:

X.X.X.X

- \+ Rappresenta il numero dell'elenco numerato, utile sia a modellare i rami della conversazione che come identificatore univoco per il nome dell'intento. La numerazione non coincide con la numerazione prodotta dalla fase 1 di progettazione dei dialoghi, in quanto solitamente ogni intento racchiude più frasi della conversazione. La numerazione inizia 1 e continua in base all'evoluzione della conversazione. È utile modellare ogni blocco conversazionale con un nuovo numero iniziale così che nella fase 3 dell'implementazione si avranno tutti gli intenti implementati ordinati per tale numero.

!!!!

- \- Se presente vuol dire che l'intento è stato già implementato. Se sono presenti solo "!!" vuol dire che è iniziata l'implementazione di un intento ma non è ancora terminata.

[USER " *Cosa dice l'utente* "]

- \+ Rappresenta ciò che l'utente dice ed innesca l'attivazione dell'intento. Il testo può contenere esempi di particolari istanze, parole chiave che, successivamente, verranno mappate su delle entità. Ad esempio la frase "Yes" può essere riferita ad un entità di risposta affermativa

/ Sinonimo di ciò che dice l'utente

- \- Può essere inserita una lista di frasi separati da "/" per indicare le frasi alternative che l'utente può digitare con lo stesso intento della frase precedentemente descritta.

(Contesto-Entrata)

- \- Rappresenta il contesto in entrata della frase. Viene generato come output dagli intenti precedenti. Serve per collegare i vari rami conversazionali.

{ X.X.X.X.Nome-intento }

- \+ È l'identificatore di ogni intento. Deve essere univoco ed espressivo ai fini della ricerca. Inserire il numero dell'elenco al suo interno ne aiuta la

gestione.

(Contesto-1-uscita X, Contesto-2-uscita X)

- - Rappresenta i vari contesti in uscita di un intento. Serve per collegare i vari rami conversazionali. È possibile definirne diversi separati da una virgola, ognuno di essi può avere un intero positivo associato che ne indica il tempo di vita, ciò rappresenta per quante interazioni con l'utente il contesto rimarrà attivo. Di default Api.ai gli associa il valore 5, tuttavia nella nostra sintassi se non viene specificato è uguale ad 1. È importante che tale valore sia sufficientemente alto da permettere al bot di non perdere il contesto, nel caso in cui l'utente digiti qualcosa di sconosciuto al bot stesso.

[BOT: "Cosa dice il bot" *# nome-parametro-1 ## tipo parametro 1]* BOT

- - In base alla tipologia di intento può essere necessario richiedere dei parametri all'utente. Questo può essere effettuato mediante specifiche frasi. Il nome di ogni parametro viene preceduto dal "#", il tipo da "##", se il parametro è opzionale verrà esplicitamente indicato in quest'ultimo.

[BOT:"*Cosa dice alla fine il bot"]* BOT

- - Il bot per ogni intento può avere una risposta finale contenente i vari elementi descritti nel capitolo 2, carosello, card, bottoni, ecc.

BOT**_Nome-azione_

- - Il bot può segnalare delle azioni che altre applicazioni eseguiranno.

Non è una sintassi rigida, il sistema continua a funzionare correttamente anche se la sintassi non viene rispettata completamente, questo perché la fase successiva viene effettuata "a mano", quindi pur di inserire maggior capacità espressiva si possono non seguire tutte le linee guida appena descritte.
I colori e le parentesi sono molto utili per attirare velocemente l'attenzione su alcuni particolari e strutturare meglio i testi:

	[]	Indica l'inizio e la fine di una frase dell'utente o del bot
USER		Precede ciò che dice l'utente
	()	Segnala i contesti in entrata e in uscita
	{}	Contiene il nome dell'intento

BOT	Precede ciò che dice il bot
	Indica le caratteristiche dei parametri
	Si riferisce all'azione da prendere

Un esempio della sintassi è mostrato nella seguente figura.

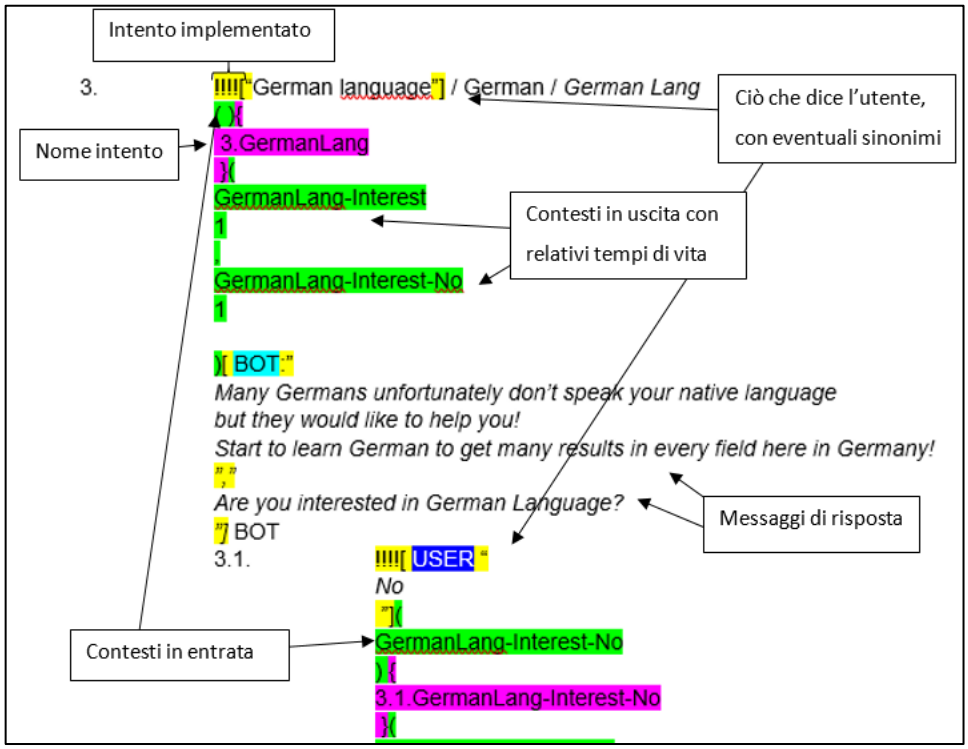


Figura 55 Esempio di intento strutturato secondo la nostra sintassi definita nella fase 2 dell'implementazione dei dialoghi.

L'esempio in figura mostra l'intento "3.GermanLang" che si aziona quando l'utente digita "German language" o i suoi sinonimi "German", "German Lang", o frasi simili del tipo "tell me about German" che il sistema Api.ai riconosce automaticamente basandosi sulla similarità con le frasi da noi definite.
Una volta attivato l'intento, il sistema invia due messaggi all'utente "Many Germans unfortunately…" e "Are you interested in German Language?", produce due contesti in uscita "GermanLang-Interest" e "GermanLang-Interest-No", che rimarranno in vita solo per il prossimo messaggio, a meno che non vengono ricreati nel successivo evento che viene attivato.

In questa fase e nella successiva si notano le potenzialità di Word. L'elenco puntato ci permette di avere una struttura logica da seguire e permette di avere nomi univoci

per gli intenti, favorendone la gestione, soprattutto in Api.ai. Le frasi possono essere formattate esattamente per come devono essere mostrate all'utente, molto importante perché sull'interfaccia grafica di Api.ai il tasto invio viene usato per dare conferma in automatico e non si riescono ad inserire gli accapo nei messaggi. Scrivendo le frasi su righe separate si perde di leggibilità, ma così è possibile selezionare esattamente il contenuto necessario con 3 click del mouse, per poi copiarlo e incollarlo su Api.ai molto velocemente.
Per gestire le dimensioni degli elenchi puntati nel documento Word è stato impostato un foglio di dimensioni personalizzate 50 cm x 50 cm, una dimensione caratteri su 3, zoom della pagina 500%.

3) La fase 3 come mostra di seguito la Figura 56 è un "copia e incolla" dal file testuale strutturato con la nostra sintassi, alla piattaforma Api.ai.
È stato utilizzato Google Chrome come browser in modalità incognito perché è più veloce di quella standard. Questa fase potrebbe essere semplificata utilizzando un compilatore che prende in input i file word da noi generati e restituisce i file JSON strutturati per Api.ai.

Il procedimento appena visto è simile allo sviluppo di un algoritmo che passa prima dall'idea poi al pseudolinguaggio, per arrivare al vero e proprio codice sorgente. Come mostrato dalle frecce scure nello schema in Figura 49 Fasi dell'implementazione. Queste fasi possono comunicare fra loro anche all'indietro, proprio come un sistema dotato di retroazione.
La principale retrazione avviene quando in fase di test si ci accorge che il chatbot non risponde correttamente, oppure dalla fase 2 alla fase 1, quando si capisce che il chatbot entrerà in confusione per gli stessi motivi descritti nel capitolo 2, in questo caso sarà necessario riformulare la frase diversamente.
Un'osservazione potrebbe essere, perché non codificare direttamente in json gli intenti, ovvero nel loro formato finale e non con la nostra sintassi. La risposta è nella complessità di gestire ogni file json per Api.ai che segue una struttura più complicata rispetto a quella da noi creata, in quanto contiene altre informazioni in più, utili alla loro gestione in Api.ai e che vengono generate automaticamente.

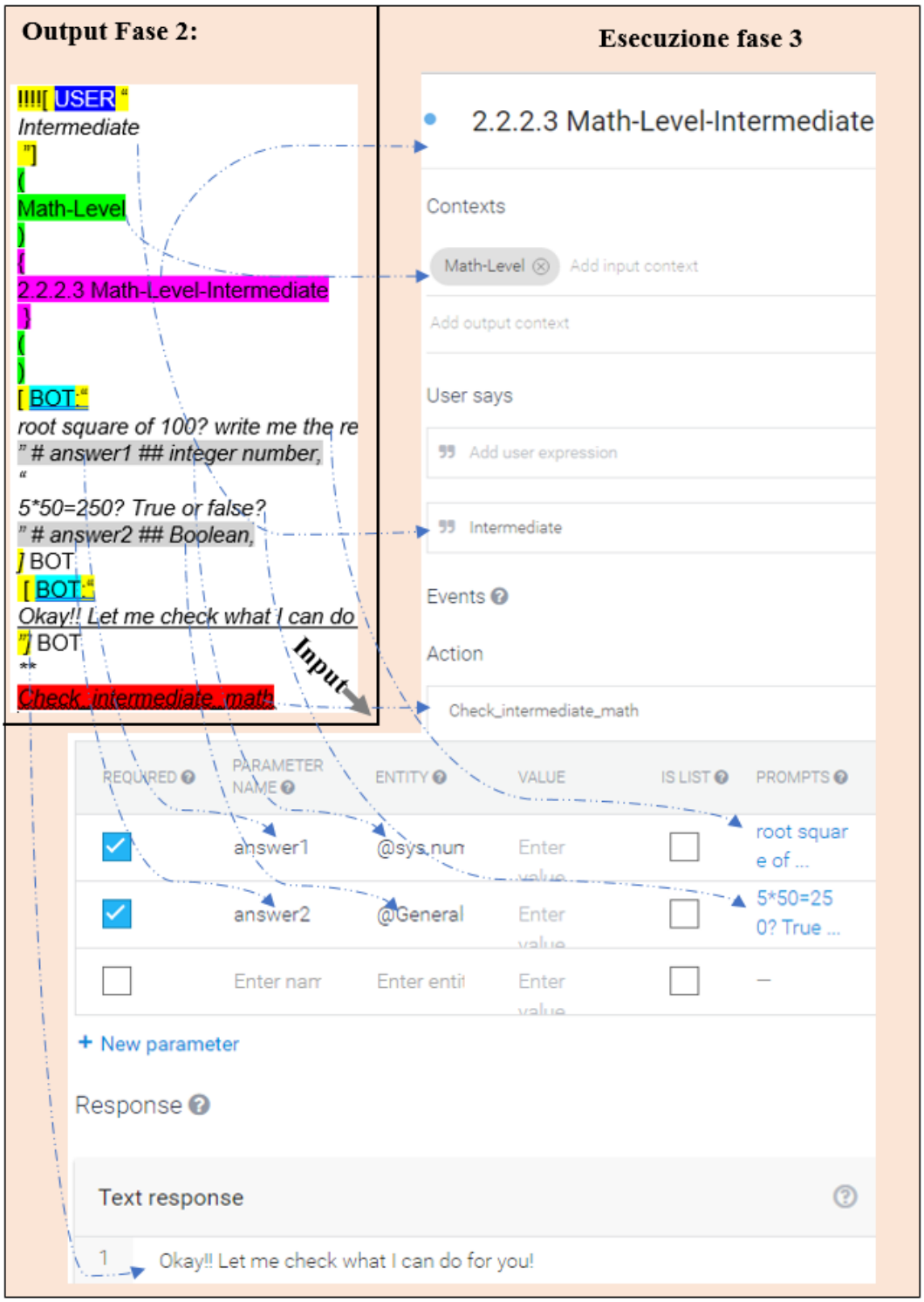


Figura 56 Fase 3 dell'implementazione dei dialoghi, tutto ciò presente nei file Word viene riportato su Api.ai mediante la sua interfaccia grafica.

Output Fase 3


```
{
  "templates": [
    "Intermediate"
  ],
  "userSays": [
    {
      "id": "dbc359c0-ca9d-4db1-a805-a5a8f0126351",
      "data": [
        {
          "text": "Intermediate"
        }
      ],
      "isTemplate": false,
      "count": 0
    }
  ],
  "id": "c1094a8c-9ee4-43e0-96ff-f7c1dcd601e1",
  "name": "2.2.2.3 Math-Level-Intermediate",
  "auto": true,
  "contexts": [
    "Math-Level"
  ],
  "responses": [
    {
      "resetContexts": false,
      "action": "Check_intermediate_math",
      "affectedContexts": [],
      "parameters": [
        {
          "required": true,
          "dataType": "@sys.number-integer",
          "name": "answer1",
          "prompts": [
            "root square of 100? write me the result"
          ]
        },
        {
          "required": true,
          "dataType": "@General-boolean",
          "name": "answer2",
          "prompts": [
            "5*50\u003d250? True or false?"
          ]
        }
      ],
      "messages": [
        {
          "type": 0,
          "speech": "Okay!! Let me check what I can do for you!"
        }
      ]
    }
  ],
  "priority": 500000,
  "cortanaCommand": {
    "navigateOrService": "NAVIGATE",
    "target": ""
  },
  "webhookUsed": false,
  "webhookForSlotFilling": false,
  "fallbackIntent": false,
  "events": []
}
```


Figura 57 Output fase 3 implementazione dei dialoghi, mostra l'intento in file JSON equivalente all'intento descritto secondo la nostra sintassi in un file Word.

Per velocizzare questo lunghissimo processo, nella realizzazione dei vari blocchi bisogno, si è evitata la fase 1 scrivendo i dialoghi direttamente strutturati con il principio dei "template". Allo stesso modo anche la fase 3 ha avuto delle agevolazioni, modificando il "template" di un altro blocco bisogno già implementato piuttosto che ricreare da zero tutti gli intenti.

4.5.1 Implementazione di Api.ai

Tutti gli intenti implementati sono stati inseriti all'interno di un agente chiamato "Ibot". Come lingua è stata scelta l'inglese. Una scelta molto forte visto che può essere assegnata una sola lingua per ogni agente e non può essere cambiata successivamente.
È di tipo privato perché al momento non vogliamo che altri sviluppatori esterni possano contribuire o usarlo nelle proprie app.

4.5.2 Problematiche riscontrate in Api.ai

L'idea di creare dei template da modificare in base all'esigenze per rendere veloce la realizzazione di nuovi percorsi è stata leggermente ostacolata durante l'implementazione. Fra le funzionalità offerte da Api.ai, vi è la possibilità di caricare un intero agente con tutti i suoi intenti automaticamente su un altro agente in modo da avere la fusione dei due agenti, in caso di intenti con lo stesso nome, quelli già esistenti vengono sovrascritti.
Tuttavia quando:

1) un agente A1 viene esportato
2) importato nuovamente su Api.ai in un nuovo agente (A2), modificato adeguatamente in tutti gli intenti
3) esportato in A3
4) importato nuovamente in A1

Succede che gli intenti vengono sovrascritti pur avendo un nome differente. Probabilmente nella fusione dei due agenti, il caricamento degli intenti controlla i vari identificatori interni nella struttura json con i quali gli intenti vengono esportati. Essendo questi uguali, dato che le modifiche di un intento non modificano anche l'identificatore, risultano gli stessi intenti e quindi l'ultimo caricato sovrascrive quello precedente. Questo procedimento è l'ideale

per chatbot reali i quali devono funzionare continuamente, per cui le modifiche da apportare avvengono prima su una copia per lo sviluppo e poi vengono aggiunte a quella reale.
La soluzione da noi trovata, probabilmente la più banale, è caricare gli intenti singolarmente mediante l'apposita funzione. Tale funzione infatti non crea nessun problema e carica gli intenti nel modo aspettato, se hanno nomi diversi sono intenti diversi, infatti cambia l'identificatore interno in modo automatico. Ma è chiaro che caricare un intento alla volta comporta un aumento di tempo e della possibilità di errori di distrazione all'aumentare del numero di intenti da caricare. Tale problematica è stata segnalata. Il team di Api.ai ci ha proposto una nuova soluzione diversa per caricare gli intenti a blocchi senza creare un nuovo agente, tuttavia sembra avere la stessa problematica. Sicuramente in futuro sarà presente la stessa funzione con la possibilità di caricare più intenti contemporaneamente pur avendo gli stessi identificatori interni.

4.6 II – Implementazione del database

L'implementazione del database a grafo Neo4j è avvenuta su Graphenedb.
La versione di Neo4j utilizzata è la 2.3.9, sebbene al momento sia presente anche la 3.1.1, perché, fra i vari driver disponibili per Node.js[40], abbiamo scelto node-neo4j[41] il quale non supporta le versioni 3.x. Quest'ultima scelta è dovuta ad un'applicazione template[42] sul sito di Heroku[43], che permette di usare tali driver per uno sviluppo rapido e probabilmente sicuro. Questi driver, come la maggior parte dei driver per Neo4j, permettono di utilizzare Cypher come interfaccia.

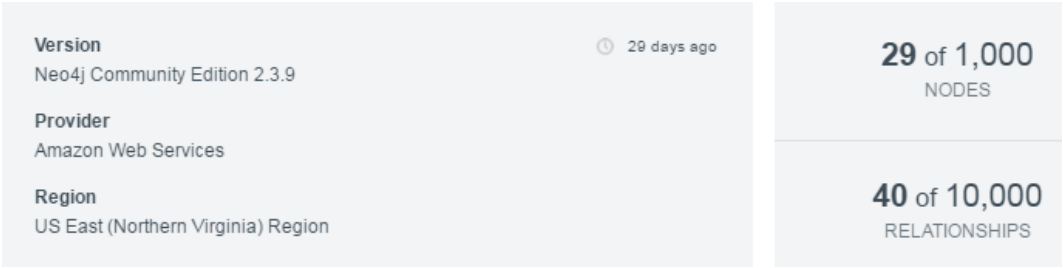


Figura 58 Screenshot Console Graphenedb il 26/02/2017.

[40] https://devcenter.heroku.com/articles/graphenedb#using-with-node-js-and-node-neo4j
[41] https://github.com/thingdom/node-neo4j
[42] https://github.com/aseemk/node-neo4j-template
[43] https://elements.heroku.com/buttons/aseemk/node-neo4j-template

Come mostrato in Figura 58, è stata utilizzata la versione gratuita, Hobby. Questa offre di gestire fino a 1000 nodi e 10.000 relazioni, su server nella regione AWS US East (Northen Virginia). È presente anche una regione AWS EU (Frankfurt), ma è a pagamento. Il database è stato creato su un'istanza di Neo4j in locale, poi è stato importato su Graphenedb attraverso un'espressione Cypher, presente in Appendice D.

Nella Figura 59 viene mostrata un'istanza del nostro database. La grafica è stata modificata, cambiando i colori e le dimensioni degli elementi del grafo. Il file che la gestisce (.grass) è riportato nell'appendice D.

Figura 59 Istanza del database visualizzata tramite la console di Neo4j. In alto, sono mostrate le varie etichette ed i tipi di relazione con i colori associati. In basso, si può notare un nodo lavoro, "Commesso", che richiede 2 certificati, "A" e "B", con i vari percorsi di formazione associati.

4.7 III – Implementazione dell'interazione fra i dialoghi ed il database

La webapp, realizzata con Node.js ed ospitata sui server di Heroku, è stata sviluppata partendo da un progetto Github[44]. Quest'ultimo, è un insieme dei vari blocchi di codice presenti nelle documentazioni ufficiali di Messenger Platform ed Heroku. La sua documentazione è un videocorso su Udemy[45]. Come ambiente di sviluppo è stato utilizzato Webstorm[46].

4.7.1 Gestione connessione al database

Come descritto sul sito di Graphene[47], la prima connessione fra la webapp e il database è avvenuta in 3 step:

1) Installazione di node-neo4j con npm[48] (package manager per JavaScript)

   Da terminale

   ```
   $ npm install neo4j --save
   ```

   Nel package.json[49] avremo `"neo4j": "^2.0.0-RC2"` all'interno di `"dependencies`

2) Impostazione dei parametri necessari alla connessione dell'istanza di Neo4j su GrapheneDB.

   Javascript

   ```
   var neo4j = require('neo4j');
   var db = new neo4j.GraphDatabase("http://... Url della nostra istanza
   su graphenedb  ... com:24789");
   ```

[44] https://github.com/jbergant/chatbot-facebook-nodejs.git
[45] https://www.udemy.com/chatbots/
[46] https://www.jetbrains.com/webstorm/
[47] http://docs.graphenedb.com/docs/nodejs#section-node-neo4j
[48] https://www.npmjs.com/
[49] https://docs.npmjs.com/getting-started/using-a-package.json

3) Verifica della connessione con un codice di esempio.

Javascript

```
db.cypher({
    query: 'CREATE (n:Person {name: {personName}}) RETURN n',
    params: {
        personName: 'Bob'
    }
}, function(err, results){
    var result = results[0];
    if (err) {
        console.error('Error saving new node to database:', err);
    } else {
        console.log('Node saved to database with id:', result['n']['_id'
]);
    }
});
```

4.7.2 Gestione degli intenti

Gli intenti implementati sono più di 30. Sono stati implementati tre tipi di blocco bisogno: Matematica, Cultura generale, Tedesco. Per ognuno di essi sono stati necessari 11 intenti, organizzati come descritto precedentemente. Essi interagiscono con l'applicazione utilizzando strutture JSON, in particolare mediante "azioni", "parametri". Nella tabella seguente vengono mostrati gli intenti del blocco "Matematica".

Nome intento	Azione e parametri
2.Math	
2.1.Math-Interest-No	
2.1.1.Math-Interest-No-Motivation	
2.1.2.Math-Interest-Yes-Motivation	**Parametro 1:** motivation-not-interested **Tipo parametro 1:** @sys.any
2.2.Math-Interest-Yes	
2.2.1 Math-Questions-No	**Azione:** c_MATEMATICA **Descrizione azione:** cerca su db tutte le attività appartenenti alla categoria (c) Matematica, indipendentemente dalle sottocategorie esistenti.
2.2.2.Math-Questions-Yes	
2.2.2.1Math-Level-Nothing	**Azione:** s_MATEMATICA-4 **Descrizione azione:** cerca su db tutte le attività appartenenti alla sottocategoria (s) della categoria Matematica identificata dalla proprietà idSC con valore 4.

2.2.2.2 Math-Level-Beginner	**Azione:** q_s-MATEMATICA-4 **Parametro 1:** answer1 **Tipo parametro 1:** @General-boolean **Parametro 2:** answer2 **Tipo parametro 2:** @sys.number-integer **Descrizione azione:** cerca su db tutte le attività appartenenti alla sottocategoria della categoria Matematica identificata dalla proprietà idSC con valore 4. Esegue la verifica delle risposte, memorizzare nei parametri answer1 e answer2, di un quiz (q) di due domande.
2.2.2.3 Math-Level-Intermediate	**Azione:** q_s-MATEMATICA-5 **Parametro 1:** answer1 **Tipo parametro 1:** @sys.number-integer **Parametro 2:** answer2 **Tipo parametro 2:** @General-boolean **Descrizione azione:** analoga all'azione precedente(q_s-MATEMATICA-4) con idSC=5.
2.2.2.4.Math-Level-Advanced	**Azione:** q_s-MATEMATICA-6 **Parametro 1:** answer1 **Tipo parametro 1:** @General-boolean **Parametro 2:** answer2 **Tipo parametro 2:** @sys.number-integer **Descrizione azione:** analoga all'azione q_s-MATEMATICA-4 con idSC=6

Tabella 4 Intenti del blocco bisogno Matematica con azioni e parametri.

Come si può notare non tutti gli intenti hanno dei parametri o delle azioni. Il tipo di ogni parametro varia sia nel formato (stringhe, numero, ecc.), che nella semantica ("Milano" e "venerdì" sono entrambe stringhe di caratteri ma rappresentano due concetti diversi, rispettivamente città e giorno della settimana). Come descritto nella sezione 2.4.2, è possibile utilizzare delle entità definite da Api.ai o dallo sviluppatore, per mappare i parametri richiesti.

Entità Api.ai

Sono state definite due nuove entità:

1. General-levels, per mappare il livello di competenze (base, intermedio, ecc.)

2. General-boolean, per definire il tipo booleano per la lingua inglese. Esso restituisce
   "yes" per yes, true, y, Y, yep, of course, absolutely.
   "no" per no, false, impossible, n, not.

Controllo parametri

La scelta dell'entità è rilevante per il controllo dei parametri. Le opzioni principali per controllare i parametri inseriti dall'utente sono 3:

1. Livello linguaggio (api.ai)
2. Livello applicazione (node.js)
3. Livello di database (neo4j)

È possibile combinare queste opzioni per ottenere il controllo desiderato. Ad esempio nel blocco "Cultura generale", per il parametro con semantica "città" sono stati usati tutti e tre:

1. Entità del tipo @sys.geo-city , che permette di accettare le città gestite da Api.ai. In questo caso le città espresse in lingua inglese (es. Munich).
2. Non distinguiamo le maiuscole dalle minuscole, ma possono essere implementate altre logiche di similarità, ad esempio per evitare errori di battitura.
3. Inseriamo diversi valori per lo stesso concetto (es. Munich, München, Monaco di Baviera).

Azioni Api.ai e relative Cypher query

Le azioni definite seguono una sintassi del tipo:
[tipo azione]+"_"+[informazioni per eseguire l'azione]
I tipi di azioni possibili sono 4:

- "c" esegue una query su una categoria identificata dal nome per ottenere tutte le attività ad essa associate.
  Es. "c_MATEMATICA"

  ```
  MATCH (c:Categoria)-[:HA]-(a:Attività)
  WHERE c.Nome = {categoryName}
  RETURN a
  ```

- "s" esegue query su una sottocategoria identificata dal suo codice idSC per ottenere tutte le attività ad essa associate.

Es. "s-MATEMATICA-4"

```
MATCH (c:Categoria)-[:DELLA]-(s:Sottocategoria)-[:HA]-(a:Attività)
WHERE c.Nome = {categoryName} and s.idSC = {subcategoryID}
RETURN a
```

- "lp" esegue una query sui percorsi di formazione per ottenere tutte le attività relative al loro primo passo di formazione. I percorsi vengono individuati utilizzando l'identificatore del certificato IdCe,
  Es. "lp_2"

```
MATCH (c:Certificato)-[s:SUCCESSIVO]-(e:Elemento)-
[:APPARTIENE]->(csc)<-[:HA]-(a:Attività)
WHERE c.IdCe = {IdCeName}
RETURN a, id(e) as ids
UNION
MATCH (c:Certificato)-[s:SUCCESSIVO]-(e:Elemento)-
[:CORRISPONDE]->(a:Attività)
WHERE c.IdCe = {IdCeName}
RETURN a, id(e) as ids
```

- "q" esegue una query su una categoria identificata dal nome, necessaria per ottenere le risposte corrette di un quiz riferita ad essa,
  Es. "q_s-MATEMATICA-6"
  Cypher query uguale al tipo 1, "c".

I fattori principali che hanno influenzato questa sintassi sono due: leggibilità e codice di supporto. Si sarebbero potuti utilizzare solo gli identificatori delle categorie al posto del loro nome; ciò avrebbe portato ad azioni indicate da stringhe incomprensibili senza l'aiuto di tabelle di rifermento. Il nome della categoria nelle azioni poteva essere del tutto omesso, visto che è presente anche in altri parametri (nome intento, contesti, ecc) ma ciò avrebbe richiesto molto più codice per manipolare tale informazione.
La leggibilità e quantità del codice si sono dovute confrontare anche con i vincoli delle varie piattaforme. Le azioni, per come sono definite in Api.ai, non possono avere spazi

vuoti per cui, o sono stati rimossi ("Cultura generale"->"Culturagenerale") o sono stati sostituiti con caratteri separatori, "-", "_", e gestiti con apposito codice.

Quiz

I quiz sono un insieme di domande, nel nostro prototipo solo 2, che permettono al rifugiato di valutare le sue competenze. La loro implementazione prevede diversi tipi di domande, sia a risposta aperta (Es. "Capitale della Germania?", "Radice quadrata di 625?") che a risposta chiusa ("6*5 = 30 ? "). Tutte le domande prevedono che l'utente scriva le risposte e non clicchi su dei bottoni (es. bottoni nei template, che generano eventi del tipo messaging-postbacks). Quest'ultimo, in modo che la comunicazione con Ibot sia il più possibile simile a quella che avviene con un umano. Tuttavia, quest'ultima cosa comporta maggior lentezza nelle conversazioni, più errori e del codice di supporto.
La verifica delle risposte corrette avviene in tutti i casi mediante un controllo di uguaglianza fra stringhe, senza distinzione di maiuscole e minuscole, fra la risposta inserita dall'utente e tutte le risposte corrette possibili inserite nel database.

4.8 Visualizzazione dei percorsi di formazione

Una volta definiti nel database i precorsi di formazione necessari per ottenere un dato certificato essi devono essere mostrati all'utente. A questo punto del lavoro sono possibili numerose alternative per decidere quali informazioni mostrare e come mostrarle. Ad esempio, la soluzione più facile da implementare sarebbe sicuramente mostrare tutte le attività relative ad ogni percorso di formazione insieme. Soluzione che richiede una sola query al database e poco codice per la sua gestione, non richiede l'interazione con l'utente ed è quasi incomprensibile per l'utente finale. La soluzione adottata ragiona per passi:

- Viene mostrato un solo passo, step, alla volta.
- Ogni passo è composto da tutte le attività possibili per completarlo.
- Le attività di un passo sono disposte su un carosello, H-scroll (descritto in sezione 2.3.3). Ciò indica che le attività disposte sulla stessa riga sono opzionali, ne basta una sola per completare lo step.
- Per mostrare le attività dello step successivo, l'utente deve premere su una attività di quelle mostrate nello step presente sullo schermo.

Questa soluzione è davvero comoda per l'utente, che può scegliere un percorso di formazione in funzione dei singoli step. Se invece l'utente deve decidere il suo percorso di formazione basandosi su un'informazione complessiva, questa diventa molto scomoda.

Ad esempio, ritornando all'analogia con Google Maps, se si vuole scegliere un percorso valutando il percorso più economico o più breve o più veloce, la nostra modalità di visualizzare i risultati sarebbe molto scomoda. Se invece il percorso che si vuole scegliere prevede di effettuare delle tappe nelle città più belle, allora la nostra soluzione potrebbe aiutare la scelta.

L'implementazione di questa soluzione prevede una query, mostrata precedentemente nel caso al tipo "lp" di azioni Api.ai (query A), per mostrare tutte le attività del primo step e la seguente (query B), per mostrare le attività di uno step successivo:

```
MATCH (e:Elemento)-[:SUCCESSIVO*2..2]->(:Elemento)-[:CORRISPONDE]-(a)
WHERE  id(e) in [57,59]
RETURN a , id(e) as ids
UNION
MATCH (e:Elemento)-[:SUCCESSIVO*2..2]->(:Elemento)-[:APPARTIENE]->(csc)<-
[:HA]-(a)
WHERE id(e) in [57,59]
return a , id(e) as ids
```

Nella query sono presenti 3 concetti non descritti precedentemente:

1. [:SUCCESSIVO*2..2], permette di definire il numero minimo e massimo di relazioni, del tipo specificato, che devono essere presenti nel pattern cercato. Per indicare un numero esatto di relazioni, massimo e minimo coincidono.[50]
2. UNION, permette di unire i risultati delle due query.[51]
3. id(e) in [57,59], la condizione è vera se il valore è presente nella lista.[52]

[50] http://neo4j.com/docs/2.3.9/query-match.html#match-variable-length-relationships
[51] http://neo4j.com/docs/2.3.9/query-union.html
[52] http://neo4j.com/docs/2.3.9/query-where.html#_collections

La query prende in ingresso due parametri:

1. Una lista di identificatori interni di Neo4j (Es. [57,59], due percorsi) che rappresentano il primo step di ogni percorso di formazione che si può effettuare.
2. Il numero di step precedente a quello desiderato (Es. 2..2, cerchiamo il terzo step).

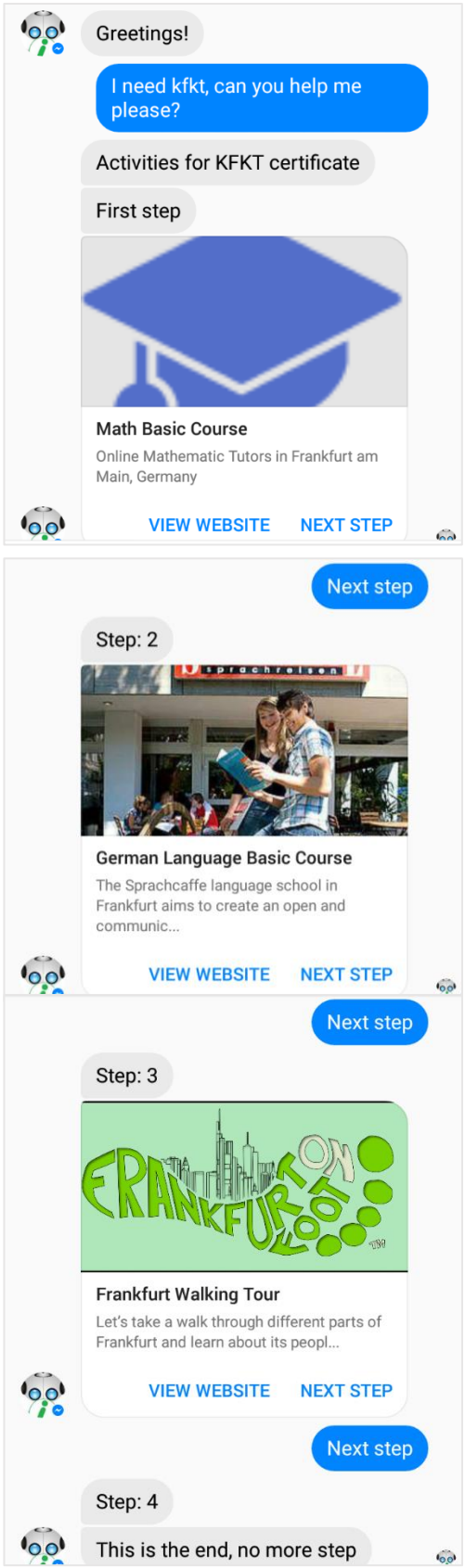
Entrambe le due query, A e B, restituiscono una lista contente le informazioni di ogni attività insieme al percorso di formazione a cui appartengono, indentificato dal nodo elemento di partenza e indicato con "ids". Un esempio di risposta è la seguente:


```
[
  {
    "a": {
      "_id": 48,
      "labels": [
        "Attività",
        "Certificata"
      ],
      "properties": {
        "idA": 11,
        "Nome": "Corso intermedio di matematica gratuito Goethe University"
      }
    },
    "ids": 59
  },
  {
    "a": {
      "_id": 45,
      "labels": [
        "Attività",
        "Certificata"
      ],
      "properties": {
        "idA": 10,
        "Nome": "Corso base di matematica gratuito Goethe University"
      }
    },
    "ids": 59
  }
]
```


La formulazione di tutte le query ha dovuto considerare due opzioni:

1. Quantità e complessità del codice di supporto alla query.
2. Vincoli del linguaggio Cypher.

Per formulare la query B sono state necessari diversi giorni di lavoro in cui sono state analizzate varie soluzioni. Alcune funzionalità che in SQL sarebbero banali, in particolare quelle riguardanti l'aggregazione in Cypher, diventano molto complesse, se non impossibili. Ad esempio, non è possibile effettuare operazioni di aggregazione su result set ottenuti mediante la clausola UNION. Ciò comporta l'utilizzo dei workaroud ai fini di ottenere risultati simili[38](La soluzione proposta sul blog ufficiale di Neo4j ha un problema, nei commenti vi è la correzione). Per cui nella soluzione implementata, ad esempio, l'eliminazione dei duplicati ottenuti a causa di ids, avviene a livello di applicazione.



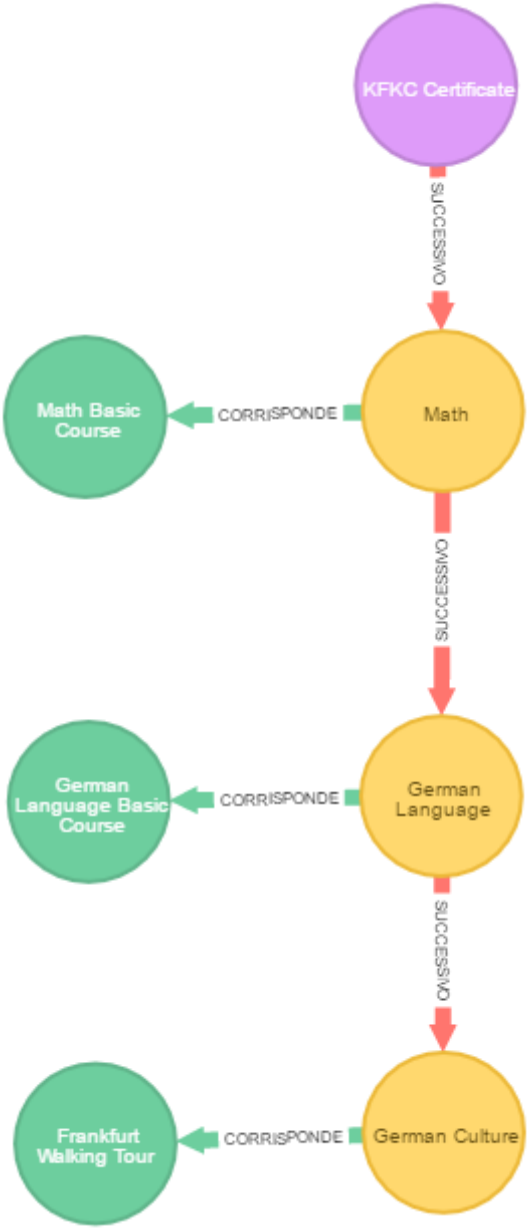


Figura 60 Visualizzazione di un percorso di formazione, a sinistra mediante screenshot su smartphone Android, a destra nella console di Neo4j.

Nella Figura 60 vengono mostrati gli screenshot dell'interazione di ibot con un utente. L'utente cerca le informazioni per ottenere il certificato KFKC e le ottiene sotto forma di messaggi testuali e card. Ogni card è composta da 5 elementi: Immagine, titolo e descrizione riferiti ad un'attività, un bottone (VIEW WEBSITE) per aprire il sito web da cui sono state prelevate le informazioni, ed un bottone (NEXT STEP) per visualizzare il prossimo elemento di formazione.

Tutte le informazioni che riguardano il numero del prossimo step, l'identificatore del certificato in esame, l'identificatore dell'attività scelta e i percorsi di formazione percorribili, vengono memorizzati come un'unica stringa, nel payload del bottone NEXT STEP che è di tipo postback. (Es. "Ns_1_2_3_4,5,6" indica step corrente 1, certificato con IdCe 2, attività della card con idA 3, identificatori interni di neo4j, per gli elementi di formazione in testa ai percorsi di formazione percorribili 4,5,6)

L'esempio in Figura 60 mostra un caso base dove un certificato che si può ottenere con un solo percorso di formazione. Nel caso di un certificato con più percorsi, il funzionamento logico di Ibot per la scelta di un unico percorso di formazione, consiste nel generare un insieme di percorsi percorribili, memorizzati come appena descritto. L'utente scegliendo le attività con il bottone NEXT STEP ad esse collegate, elimina i percorsi non più percorribili dall'insieme. Alla fine l'insieme conterrà un solo elemento che indica il percorso di formazione finale scelto dall'utente. Ad esempio, se viene scelta una attività presente in tutti i percorsi, allora non verrà eliminato nessun percorso dall'insieme. Se invece, l'attività scelta è presente in un unico percorso, allora tutti gli altri percorsi finora percorribili vengono rimossi.

4.9 Codice sorgente

Il progetto è presente su Github al seguente indirizzo: https://github.com/ginvodka/ibot . Esso contiene l'intero codice sorgente dell'applicazione e il backup completo dell'agente di Api.ai con tutti gli intenti. Il codice dell'applicazione, partendo da un progetto open source, contiene funzionalità non utilizzate ma utili per eventuali usi futuri.

4.10 Esempio di conversazione con Ibot

Per concludere questo capitolo viene mostrato, in figura, un esempio di conversazione con Ibot che prevede alcune funzionalità descritte precedentemente.

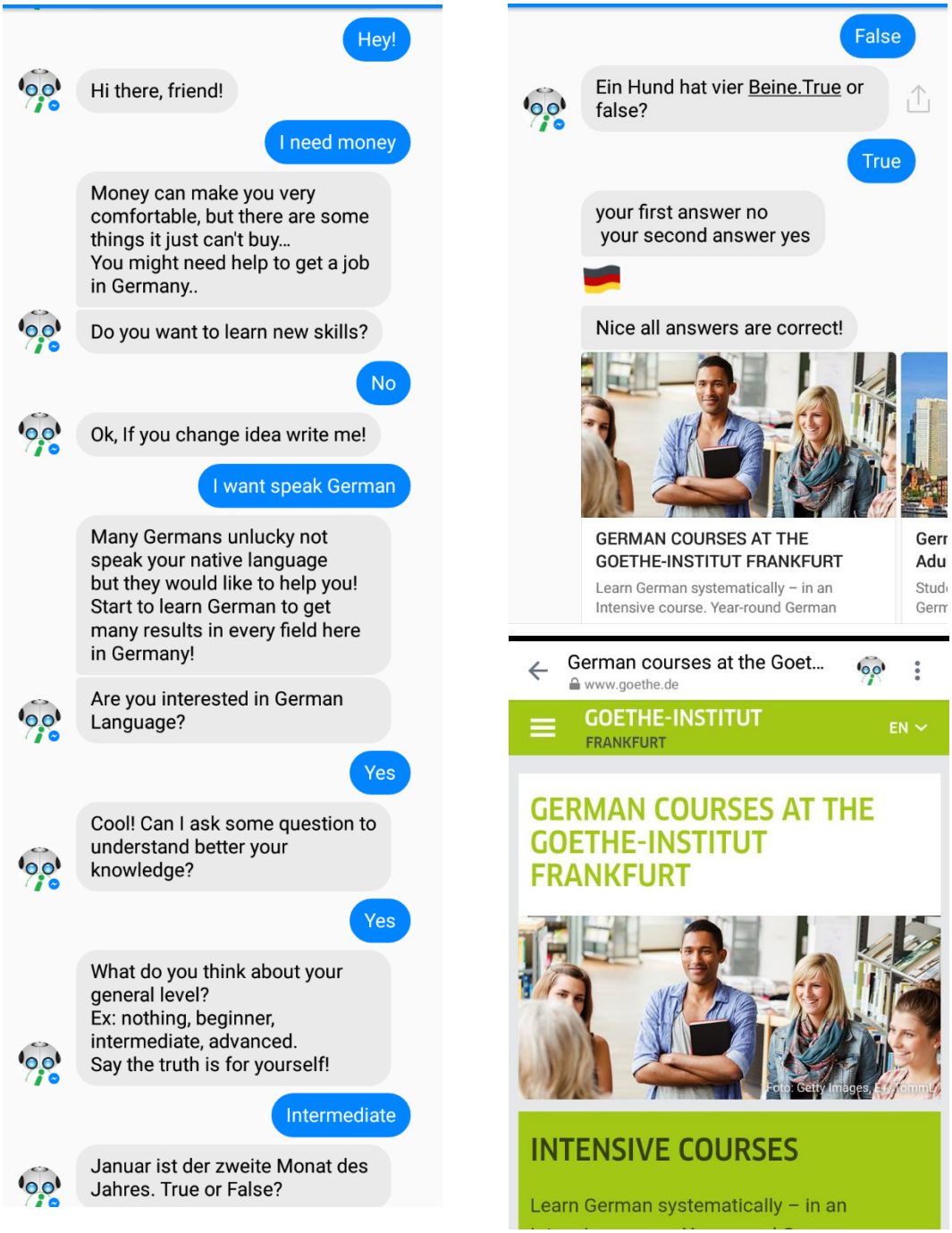


Figura 61 Esempio di conversazione con Ibot, con quiz per valutare il livello di tedesco e visualizzazione del sito web che offre il corso da Ibot suggerito.

Capitolo 5

Conclusioni e lavoro futuro

Per la stesura di questa tesi ci siamo occupati della realizzazione di alcune parti di un progetto finalizzato ad aiutare i rifugiati della città di Francoforte sul Meno. In particolare modo la realizzazione di un prototipo per l'apprendimento direzionato. Ci siamo serviti di nuove tecnologie e metodologie per la creazione di un chatbot che interagisce con un DBMS a grafo.

5.1 Limitazioni: Costi, privacy e sicurezza

La tesi non ha analizzato alcuni aspetti importanti, in particolare i costi e il trattamento dei dati personali. L'insieme dei costi relativi al progetto, presentato mediante un prototipo, è davvero esteso. Una prima classificazione può essere effettuata in base alla loro natura tecnica (server, programmatori, energia elettrica, ecc.) e non tecnica (avvocati, traduttori, marketing per la sua promozione, ecc.). Per cui, in futuro, saranno necessari dei fondi che finanziano il progetto, affinché questo possa continuare nel tempo.
Il progetto finale dovrà occuparsi di rispettare la privacy, in particolare quella dei rifugiati, utilizzando strumenti di sicurezza adatti, seguendo dei principi, rispettando un codice etico rigoroso e, molto importante, ricordando che i dati riguardano delle persone.

5.2 Raccomandazioni per lavoro futuro

Alcuni punti chiave per eventuali lavori futuri:

Usabilità
• Il chatbot deve parlare più lingue. • La necessità di personale qualificato in comunicazione, sia nell'interazione con i rifugiati che con gli altri stakeholder. • Lo sviluppo di interfaccia visuale user-friendly per inserire le informazioni nel database. • Fase di testing che analizzi sia il comportamento umano che del sistema. Nel primo caso, si potrebbero effettuare dei test con delle vere persone che suggeriscono i percorsi di formazione ai rifugiati al posto del bot.
Tecnologia
• L'impiego e/o sviluppo di nuovi strumenti, che permettono di caricare i dialoghi nel sistema in modo più efficiente. • Approfondire la gestione del database, in particolare le modifiche. • Utilizzo di altre soluzioni implementative. Sono state utilizzate tutte versioni gratuite, ottime per la prototipazione, ma non per la vera distribuzione. In realtà, dietro un corrispettivo monetario, tutte le tecnologie utilizzate offrono la possibilità di aumentare le risorse in modo semplice e immediato. • Valutazione su come migliorare il sistema di raccomandazione, ad esempio utilizzando lo stesso approccio del progetto *EDISON Data Science Competence Framework*, che permette di analizzare le competenze delle persone mappandole su unità di apprendimento, al fine di creare dei percorsi di formazione personalizzati.[39] • L'integrazione con altre applicazioni. Questo è uno dei punti chiave più importarti per collaborare con i fornitori di contenuti. Come mostrato nel capitolo 3, lo scopo dell'applicazione non è raccogliere da zero le attività bensì utilizzare meccanismi (API, web scraping, ecc.) per servirsi delle attività già presenti in altre applicazioni.
Legalità
• Studio del trattamento dei dati per poter memorizzare i dati sui rifugiati nel sistema.

Tabella 5 Punti chiave per lavori futuri.

5.3 Esperienza vissuta a Francoforte

L'esperienza è iniziata il 06.09.2016. Sono arrivato all'aeroporto di Frankfurt-Hahn che dista circa 2 ore dalla città di Francoforte sul Meno, collegato tramite un autobus. Le prime notti le ho trascorse in ostello, il "Five Elements Hostel" in attesa di trovare un alloggio. Ostello economico, 19 euro per notte, molto carino, con varie attività di intrattenimento. Ha il vantaggio e lo svantaggio di essere vicino alla stazione centrale, comodo per i trasporti ma è situato in una delle peggiori zone della città. Lì è iniziata l'avventura.

Attività progettuale

L'attività progettuale è stata svolta presso il Frankfurt Big Data Lab (Università Goethe di Francoforte sul Meno) per un periodo di 6 mesi. Fin dal primo giorno, l'ambiente di lavoro è stato molto positivo. Mi è stata affidata una postazione in un ufficio condiviso con due colleghe, Kim Hee e Concha Sanchez. Il laboratorio è dotato di vari confort: sala meeting, stanza fotocopie, cucina, un divanetto, ecc. Il team del lab è composto da varie persone interessanti di diverse nazioni: Italia, Spagna, Bulgaria, Germania, India, ecc. Con loro non era solo un rapporto di lavoro ma anche di amicizia.
Nei primi tre mesi, sono stato seguito da Concha, project manager del progetto Integreat.
È stata una fase di analisi dei requisiti e di sviluppo di un'applicazione web.

Durante quel periodo ho avuto l'occasione di apprendere competenze di ogni genere: nuove tecnologie, nuove metodologie, come si lavora in gruppo, come si gestisce un progetto con più persone a distanza, come strutturare le presentazioni in modo professionale, come comunicare nei meeting, ecc. Ho avuto l'occasione di partecipare ad alcune lezioni del "*Web Business: Data Challenges WS 2016"*, un corso universitario da loro sostenuto.

L'applicazione web realizzata è stata molto utile per raccogliere i requisiti del progetto Integreat in maniera iterativa, seguendo un ciclo di tre fasi:
raccolta requisiti – sviluppo – presentazione.
La webapp non è stata documentata in questa tesi perché, sebbene necessaria a capire a fondo le problematiche, non ricopre un ruolo davvero innovativo.

Nei mesi successivi, 3 in Germania, 1 in Italia, sono stato seguito direttamente dal Prof. Roberto V. Zicari che, con meeting regolari, organizzava e valutava il mio lavoro. Lì avvenivano discussioni stimolanti che mi trasmettevano entusiasmo, motivazione e conoscenze.

Città e Persone

La città è davvero carina. Ci sono grattacieli altissimi. È molto pulita e trasmette una sensazione di sicurezza. Ci sono parchi, musei, centri commerciali, ecc.
La popolazione è molto mista. Ci sono persone di tutte le parti del mondo.

Trasporti

La città è davvero ben collegata. Tutti gli studenti dell'università Goethe hanno a disposizione una *student card* che permette di viaggiare gratuitamente con determinati autobus, treni, metro e tram. Nonostante il clima, molti si muovono in bicicletta. Ci sono delle biciclette a noleggio agevolate per gli studenti.

Lingua

La lingua ufficiale è il tedesco ma la maggior parte delle persone parla anche inglese. Pur non parlando il tedesco non ho avuto problemi di comunicazione. L'università Goethe offre vari corsi di Tedesco gratuiti.

Alloggio

L'università attraverso una società, Studentenwerk Frankfurt, si occupa di assegnare degli alloggi in dormitori agli studenti che fanno richiesta. Io ingenuamente non ho fatto richiesta perché volevo immergermi nella vera cultura tedesca. Risultato: ho cambiato 4 alloggi nei primi tre mesi. Il mio alloggio definitivo è stato il dormitorio della "Frankfurt School of Finance & Management". Il costo sostenuto è stato di 495 euro al mese. Un ambiente molto accogliente e pieno di persone in gamba.

Borsa Erasmus e Tasse universitarie

La mia borsa Erasmus è stata di 345 euro al mese. L'unica tassa universitaria sostenuta è di 307.60 euro

Eventi

In città vi sono numerose iniziative. Insieme alla mia collega Kim ho partecipato a vari meetup sui Big Data, organizzati dal "Frankfurt Analytics", ed a un hackathon IBM. Anche l'associazione di studenti universitari ESN, Erasmus Student Network, è molto attiva, con numerosi eventi culturali e ricreativi.

Attività sportive

Le varie università offrono una lista di attività sportive molto ampia. Ho frequentato il corso intermedio di salsa e un corso di yoga a soli 20 euro per tutto il semestre. Durante il corso di yoga ho applicato io stesso i principi dell'apprendimento auto-direzionato, chiedendo al maestro di insegnarci qualche esercizio di coppia, ai fini di socializzare, ed esercizi per migliorare la postura scorretta creatasi dopo le numerose ore trascorse al computer.

5.4 Conclusioni

Se molti associano il termine prototipo a qualcosa di non perfettamente funzionante, ben pochi conoscono la storia del Golden Path[40].

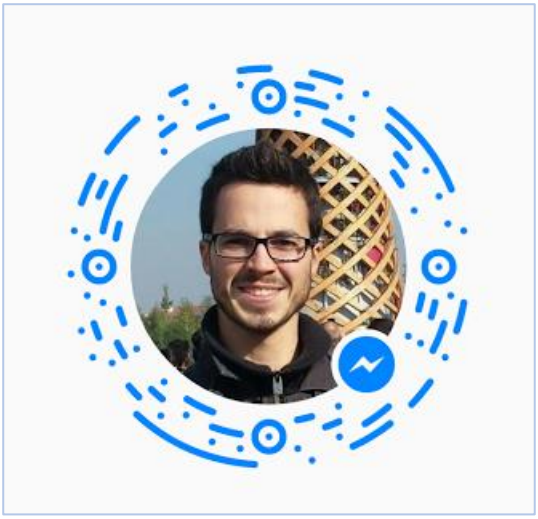
Il primo iPhone (Apple iPhone 2G) quando è stato presentato era un prototipo con vari problemi, ma Steve Jobs, seguendo una specifica sequenza delle operazioni da effettuare, il percorso d'oro, ha mostrato che il dispositivo funzionava perfettamente e nessuno nella sala si è accorto di nulla.
Quel prototipo ha dato inizio alla storia dell'iPhone.

Prototipo è quindi anche sinonimo del primo passo per grandi opportunità. Nel nostro caso specifico, se il prototipo otterrà dei successi, la sua utilità potrebbe essere sfruttata anche per aiutare i rifugiati in Italia, o in qualsiasi parte del mondo.

Il concetto di interazione fra un chatbot e un DMBS a grafo è un'idea molto innovativa che potrebbe affacciarsi su campi di applicazioni diversi da quello qui presentato. Ad esempio, essendo un'infrastruttura neutra, può essere implementata per un sistema di raccomandazione interattiva per eventi nelle vicinanze, che in realtà, maschera un sistema per combattere la criminalità.

Il prototipo sviluppato con questo progetto di tesi non è quindi un punto di arrivo, ma un punto di partenza per un potenziale, futuro, aiuto nei confronti dell'intera società.

Marco Orlando



Marco Orlando ha studiato presso l'Università di Modena e Reggio Emilia nel corso di laurea triennale (sett. 11 – lug. 14), e magistrale in Ingegneria Informatica (sett. 14 – apr. 17).

Nell'ambito del progetto Erasmus ha svolto due esperienze all'estero, Università di Lleida (Spagna, sett. 14 – feb. 15), Università Goethe di Francoforte sul Meno (Germania, sett. 16 – feb. 17).

Il suo principale hobby è la lettura di libri, principalmente di manualistica, psicologia ed economia. Balla Salsa e Bachata, ama tanto viaggiare ed incontrare nuove persone.

Ha ricevuto preziosi consigli durante il suo percorso formativo e ha il piacere a far lo stesso per chiunque abbia bisogno.

Facebook	https://facebook.com/orlamar
Linkedin	https://it.linkedin.com/in/marco-orlando-9791386b
Email	168962@studenti.unimore.it

Appendice A

Risorse consigliate per Chatbot

La quantità di risorse per creare i bot sono già tante. Fra le principali:

- Smooch.io , Abbastanza completo e ben documentato
- Api.ai, Ha un'interfaccia davvero intuitiva ed è molto popolare
- Motion.ai, Estremamente visuale, non è necessario scrivere alcun codice
- Wit.ai, Adatto per bot molto complessi
- Botpages.com , è una raccolta di differenti bot che girano su piattaforme diverse, utilissimo per trovare ispirazioni.
- botlist.co un'ottima raccolta di bot con le recensioni degli utenti

Il web è pieno di risorse per approfondire l'argomento chatbot, fra le risorse utilizzate per svolgere questa tesi vi è un ottimo corso online su Udemy di Jana Bergant [53] .
Il corso è ben strutturato ed in continuo aggiornamento, in quanto le tecnologie sono agli inizi e, frequentemente, rilasciano nuove funzioni o migliorano le precedenti.
Nel corso è presente una lista di risorse chatbot ben organizzate per categoria che viene riportata di seguito in versione originale. Piccoli test sono stati effettuati su varie risorse, ma i risultati sono troppo superficiali e soggettivi da non essere qui riportati.

[53] https://www.udemy.com/user/jana-bergant/

ChatBot prototyping

BotSociety allows you to quickly create and iterate a conversation with your future bot. Before it's coded!
Platforms: Facebook Messenger Price:Free

Bot UI kit is a fully customisable Sketch UI Kit for Messenger Platform.
Platforms: Facebook Messenger Price:Free

Messaging Design Kit is a design kit for Sketch, containing everything you need to craft beautiful and functional rich messaging experiences. Platforms: Facebook Messenger Price:Free

BotMock - Design your Facebook Messenger bot conversations with our easy drag and drop editor. No coding required. Export your mocks into a gif or video instantly. Platforms: Facebook Messenger Price:Free + Paid

Botframe is a playground to design & validate bots. Platforms: Facebook Messenger Price:Free

ChatBot Analytics

Botanalytics is a conversational analytics for bots. Platforms: Facebook Messenger, Slack Price:FREE

Bot Metrics provides instant chatbot analytics with fine grained targeting and personalized notifications. It is a secure, private, open source conversational analytics platform. Platforms: Kik, Facebook, Slack, In-app messaging Price:FREE

Dashbot is an actionable bot analytics.Increase user engagement, acquisition, and monetisation through actionable bots analytics. Platforms: Facebook Messenger, Slack, Kik Price:FREE

Botlytics ia an analytics for your bot. Track the messages your bot sends and the conversations your bot has.
Platforms: Restful API Price:FREE

ChatMetrics is an engagement platform for bots. People try your bot every day and leave. Get them to come back. Reactivate up to 27% of inactive users and improve retention. Platforms: Facebook Messenger
Price:FREE

ChatBot Developer Platforms

Api.AI is a conversational user experience platform. Build brand-unique, natural language interactions for bots, applications, services, and devices.
Platforms: Facebook, Slack, Twilio IP Messaging, Twilio, Skype, Tropo, Telegram, Kik, LINE, Spark, Alexa, Cortana, Twitter Price:FREE

Wit.ai makes it easy for developers to build applications and devices that you can talk or text to. Wit.ai learns human language from every interaction, and leverages the community: what's learned is shared across developers.
Platforms: Custom / API Price:FREE

Microdoft Bot Framework - Build a great conversationalist. Build and connect intelligent bots to interact with your users naturally wherever they are.
Platforms: text/SMS, Skype, Slack, Facebook Messenger, Office 365 mail, Kik, Telegram Price:FREE

Octane - The Easiest Way to Create a Chatbot. Drive sales, capture leads, and increase engagement on your

Facebook Page. Ready to launch in minutes.
Platforms: Facebook Messenger Price:FREE

Rebotify helps you build chat bot in a fun way. Build once, publish anywhere. Your bots work with Facebook, WeChat and Slack simultaneously.
Platforms: Facebook Messenger, We Chat, Slack, Web Price:FREE

Chatfuel - Create an AI chatbot to suit your needs. Build & launch a full-featured chatbot in 7 minutes without coding.
Platforms: Facebook Messenger, Telegram Price:FREE

Motion AI - Visually build, train & deploy bots to do just about anything. Chatbots made easy.
Platforms: Webchat, SMS, Messenger, Slack, Email, Custom/API Price:FREE

Recast AI- Create the most advanced bots in the simplest way. Collaborative Bot Platform.
Platforms: Messenger, slack, kik, Microsoft Bot Connector Price:FREE

ManyChat - Send news and content, automate interaction and much more. Easy 2-minute setup. No coding required.
Platforms: Facebook Messenger Price:FREE

Init AI - The most powerful way to build, train, and deploy intelligent conversational apps. Automate conversations, Analyze them for actionable insights,and Assist sales and support staff in communications
Platforms: Custom / API Price:FREE

Gupshup - Build, test and deploy your bot in minutes, across platforms.
Platforms: Facebook Messenger, SMS, Twitter, Telegram, Slack, Hipchat, Skype, Kik, Twillio, Line, Cisco Spark, Teamchat
Price:FREE

Flow XO - Everything you need to create and manage bots. Bot + human messaging in one great platform.
Platforms: Facebook Messenger, Slack, Telegram, SMS Price:FREE + PAID

Smooch lets you manage conversations with your customers from the business systems you already use.
Platforms: Facebook Messenger, Telegram, Twillio SMS, We Chat, Line, Slack, Web Messenger, Email
Price:FREE + PAID

BottrCo is the simplest framework for creating user facing bots.
Platforms: Facebook Messenger, Twillio, Twitter Price:FREE

BOTTR.me is a personal chat bot platform. In one click create your own bot that can simplify anything you want into a conversation and allow anyone (fans, foto talk to your virtual identity.
Platforms: Facebook Messenger Price:FREE

Botkit is the most popular open-source toolkit for creating bots for messaging platforms. It provides a semantic interface to sending and receiving messages so developers can focus on creating novel applications and experiences.
Platforms: Slack, Facebook Messenger, Twilio IP Messaging Price:FREE

ChatBot Stores

Botlist is an app store for bots. It's an official VentureBeat Media Partner.
Platforms: Android, Discord, Email, iOS, Kik, Facebook Messenger, Skype, Slack, SMS, Telegram, Twitter, Web
Price:FREE

ChatBottle is a chatbots search engine. ChatBottle search is powered by ranking algorithms with a full-text search.
Platforms: Facebook Messenger, Slack, Skype, Telegram
Price:FREE

Botpages is an open bot directory to find, discover and promote bots.
Platforms: Slack, Facebook, Telegram, SMS, Twitter, Kik, Skype, WhatsApp, Viber, Line
Price:FREE

Intento aims to provide search and discovery for infostreams, chat bots and public threads across many instant messaging platforms.

ChatBot Marketing

bCRM is a crm for marketers and customer support Staff in the chatbot world. Grow your bot business and increase engagement with powerful analytics and messaging.
Platforms: Facebook Messenger, Slack Price:Free

Bonobo.ai drives personalised user engagement and helps you acquire, grow and retain your chatbot users.
Platforms: Facebook Messenger, Slack, Viber, Telegram, Kik, Allo, Wechat, skype, Line

ChatBot Translators

Cyrano - Make your bot multilingual. Unbabel's Cyrano API instantly enables your bot to chat in more than 20 languages with human quality.
Platforms: API Price:Free

ChatBot Customer service engines

Reply.ai - Enabling one to one conversation with your customers at scale. Automate conversations with your consumers. Have your human agents take over when needed. Payments support on Messenger.
Platforms: Facebook Messenger, Kik, Line, Twitter,Web, SMS, Twillio Price:Free+Paid

Agent.ai - AI-Enhanced Customer Service. Let artificial intelligence resolve your customers' frequently asked questions in seconds, not days.
Platforms: Facebook Messenger, Kik, Telegram,Slack,WE Chat, Skype, Android, IOS Price:Free+Paid

ChatBot Job Boards

BotJobs - A job board focused on the Bot Industry.

Price:FREE

Botgig - Hire a top-tier development team to build your chatbot. Botgig connects you with top chatbot developers from our vetted talent pool. Get a guaranteed, fixed price quote.
Price:FREE

Botbuzz - On-Demand Chatbot Developers. Get a custom AI powered chatbot at a flat rate price.
Price:PAID

ChatBot Magazines

Chatbots Magazine - Top place to learn about chatbots.
Price:FREE

Chatbot's Life - Everything you need to know about chatbots.
Price:FREE

Bot Funded - Track which chatbots are getting the most funding.
Price:FREE

ChatBot NewsLetters

The Messenger - Curated newsletter on messengers, chatbots, industry experts.

Topbots curates a weekly newsletter of the best #chatbot news and events.

Bot Weekly is a weekly round up of the most interesting chatbot and AI news.

Chat Bots Weekly - Stay up to date with market insights, trends and tutorials about chatbots.

Botlist weekly - Get all the best bots delivered to your inbox every week

ChatBot Discussion Forums

Bots - Discuss chatbots, AI, NLP, Facebook Messenger, Slack, Telegram, Discord, Kik, SMS, etc. Talk about what you're working on. Platforms: FaceBook

BOTS - Ask & share knowledge reltaed to chatbots with bots enthusastics.
Platforms: FaceBook

Chatbots News - An intelligent, supportive, and collaborative community for chatbot developers, designers, creators, and enthusiasts. Moderated by the Chatbots Magazine team.

Chatbot podcasts

O'Reilly Bots Podcast covers advances in conversational user interfaces, artificial intelligence, and messaging that are revolutionizing the way we interact with software.
Price:FREE

ChatBot conferences

ChatbotConf - Explore Europe's international conference on chatbots and mobile Messaging #CBC16
Price:PAID

Chatbot Summit is the first international chatbot & artificial intelligence summit to be held in Israel. This summit brings together the leading players of newly formed chatbot economy includingChatbot developers, A.I. and machine learning experts, UX designers, product managers, chatbot platform and tools providers, startups, investors and industry thought leaders.
Price:PAID

Talkabot is a two day convergence where we will explore the past, present and future of bots in commerce, journalism and entertainment. Talkabot is for developers, entrepreneurs and technologists looking to understand more about what it takes to build a successful messaging experience.
Price:PAID

Botness is a two-day gathering in NYC to discuss all things bots: startups, tools, bots for work and play, discovery, and engagement. We will work together to shape the evolving technologies to support innovation & open collaboration.
Price:FREE

Bot Day offers the strategic and technical insight you need to start implementing AI-driven conversational interfaces that can talk to your customers, make your employees more productive, and streamline your business.
Price:FREE

Appendice B

Guida per costruire un chatbot con architettura semplice

Questa è una guida step by step non dettagliata per creare un semplice chatbot in meno di 10 min, utile anche come checklist.

1. È necessario
    a. un account fb https://www.facebook.com/
    b. un account per sviluppatori fb https://developers.facebook.com
    c. un account api.ai https://api.ai/
2. Creare una pagina fb https://www.facebook.com/pages/create/
3. Creare una applicazione fb https://developers.facebook.com (my apps, add new application), lasciate questa pagina aperta vi servirà in seguito
4. Creare un nuovo agente su api.ai https://api.ai/,
    a. integrations tab, enable fb messenger, ora otteniamo il callback url per fb app, lasciate questa scheda aperta e non confermate
5. Su fb app cliaccate add products, cercate messenger, get started, confermate
    a. Generate il token selezionando la pagina creata in precedenza
    b. Copiate il token appena ottenuto
6. Su api.ai nella schermata precedente incollate il token nell'apposito form
    a. Scrivete un proprio token a piacere ad es "alongtoken"
    b. Copiate questo token appena creato
7. Su fb, setupwebhook, incollate il token precedente
    a. Selezionate gli eventi di cui avete bisogno, es messanges
8. Su api.ai copiate l'url della callback
    a. Su fb copiate tale url
    b. Su api.ai cliccate start
    c. Ora api.ai sarà attivo
9. Su fb potete confermare premendo "verify and save"
    a. Comparirà la scritta "complete"
10. Potete testare il vostro chatbot sulla chat che preferite ad esempio m.me/nomepaginacreata

Appendice C

Caratteristiche di Neo4j

Le caratteristiche principali di Neo4j sono:

- **Native Graph storage**, la memorizzazione dei dati avviene con una piattaforma ottimizzata e progettata per la memorizzazione e gestione dei grafi. Ciò assicura la consistenza dei dati e notevoli performance.
- **Native Graph Processing,** ogni nodo del grafo possiede un puntatore diretto ai nodi adiacenti (index-free adjacency) e ciò permette di effettuare milioni di salti da un nodo all'altro al secondo, in real-time.
- **Integrità dei dati**
  Supporta transazioni ACID.
- **Modellazione "Amica della lavagna"**
  Il modello dei dati è molto simile alla realtà esaminata.
- **Un linguaggio di interrogazione potente ed espressivo**
  La quantità di codice richiesto è da 10 a 100 volte minore dell'SQL.
- **Scalabile e High Availability**
  Scalabilità verticale e orizzontale ottimizzata per i grafi.
- **Gestione ETL incorporata**
  Importazione diretta dagli altri database.
- **Integrazioni**
  Driver e API disponibili per i linguaggi più popolari.[24]

I casi d'uso di Neo4j principali sono:

- Raccomandazioni in real time
- Detenzioni di frodi
- Gestione delle identità e degli accessi
- Ricerche basate sui grafi
- Operazioni su reti

Appendice D

Codice sorgente

In questa appendice è presente parte del codice sorgente utilizzato nel progetto. A causa della dimensione del codice sorgente l'intero codice è presente su Github al seguente indirizzo: https://github.com/ginvodka/ibot.

Codice relativo a Neo4j

Codice per popolare il database con un'espressione Cypher.

```
create
(_6:`Categoria` {`Nome`:"TEDESCO",`idC`:1}),
(_7:`Sottocategoria` {`Nome`:"Tedesco di base",`idSC`:1}),
(_8:`Sottocategoria` {`Nome`:"Tedesco intermedio",`idSC`:2}),
(_9:`Sottocategoria` {`Nome`:"Tedesco avanzato",`idSC`:3}),
(_10:`Certificata`:`Attività` {`Nome`:"Corso base di tedesco gratuito Goethe University",`idA`:1}),
(_11:`Certificata`:`Attività` {`Nome`:"Corso base di tedesco gratuito FRAP ",`idA`:2}),
(_12:`Certificata`:`Attività` {`Nome`:"Corso base di tedesco gratuito Accenture Foundation ",`idA`:3}),
(_13:`Attività` {`Nome`:"Corso base di tedesco gratuito studenti volontari ",`idA`:5}),
(_14:`Attività` {`Nome`:"Corso base di tedesco gratuito ex-studenti volontari",`idA`:6}),
(_15:`Certificata`:`Attività` {`Nome`:"Corso intermedio di tedesco gratuito Goethe University",`idA`:7}),
(_16:`Attività` {`Nome`:"Corso intermedio di tedesco gratuito studenti volontari",`idA`:8}),
(_17:`Sottocategoria` {`Nome`:"Matematica di base",`idSC`:4}),
(_18:`Categoria` {`Nome`:"MATEMATICA",`idC`:2}),
(_19:`Certificata`:`Attività` {`Nome`:"Corso base di matematica gratuito Goethe University",`idA`:10}),
(_20:`Sottocategoria` {`Nome`:"Matematica intermedia ",`idSC`:5}),
(_21:`Sottocategoria` {`Nome`:"Matematica avanzata",`idSC`:6}),
(_22:`Certificata`:`Attività` {`Nome`:"Corso intermedio di matematica gratuito Goethe University",`idA`:11}),
(_23:`Certificata`:`Attività` {`Nome`:"Evento culturale per la ricerca de l lavoro",`idA`:12}),
(_24:`Categoria` {`Nome`:"CULTURA GENERALE",`idC`:3}),
(_25:`Certificata`:`Attività` {`Nome`:"Gita turistica per la città di Modena",`idA`:13}),
```

```
(_26:`Lavoro` {`IdL`:1,`Nome`:"Commesso"}),
(_27:`Certificato` {`IdCe`:1,`Nome`:"Certificato A"}),
(_28:`Certificato` {`IdCe`:2,`Nome`:"Certificato B"}),
(_29:`Elemento` {`Nome`:"Matematica"}),
(_30:`Elemento` {`Nome`:"Lingua tedesca"}),
(_31:`Elemento` {`Nome`:"Lingua tedesca"}),
(_32:`Elemento` {`Nome`:"Evento culturale"}),
(_33:`Elemento` {`Nome`:"Lingua tedesca"}),
(_34:`Elemento` {`Nome`:"Evento culturale"}),
(_7)-[:`DELLA`]->(_6),
(_8)-[:`DELLA`]->(_6),
(_9)-[:`DELLA`]->(_6),
(_10)-[:`HA`]->(_7),
(_10)-[:`HA`]->(_6),
(_11)-[:`HA`]->(_7),
(_11)-[:`HA`]->(_6),
(_12)-[:`HA`]->(_7),
(_12)-[:`HA`]->(_6),
(_13)-[:`HA`]->(_7),
(_13)-[:`HA`]->(_6),
(_14)-[:`HA`]->(_7),
(_14)-[:`HA`]->(_6),
(_15)-[:`HA`]->(_8),
(_15)-[:`HA`]->(_6),
(_16)-[:`HA`]->(_6),
(_16)-[:`HA`]->(_8),
(_17)-[:`DELLA`]->(_18),
(_19)-[:`HA`]->(_17),
(_19)-[:`HA`]->(_18),
(_20)-[:`DELLA`]->(_18),
(_21)-[:`DELLA`]->(_18),
(_22)-[:`HA`]->(_20),
(_22)-[:`HA`]->(_18),
(_23)-[:`HA`]->(_24),
(_25)-[:`HA`]->(_24),
(_26)-[:`RICHIEDE`]->(_28),
(_26)-[:`RICHIEDE`]->(_27),
(_27)-[:`SUCCESSIVO` {`Path`:1,`num`:1}]->(_29),
(_28)-[:`SUCCESSIVO` {`Path`:1,`num`:1}]->(_31),
(_28)-[:`SUCCESSIVO` {`Path`:2,`num`:1}]->(_33),
(_29)-[:`APPARTIENE`]->(_17),
(_29)-[:`SUCCESSIVO` {`Path`:1,`num`:2}]->(_30),
(_30)-[:`CORRISPONDE`]->(_10),
(_31)-[:`SUCCESSIVO` {`Path`:1,`num`:2}]->(_32),
(_31)-[:`APPARTIENE`]->(_7),
(_32)-[:`CORRISPONDE`]->(_23),
(_33)-[:`SUCCESSIVO` {`Path`:2,`num`:2}]->(_34),
(_33)-[:`APPARTIENE`]->(_7),
(_34)-[:`CORRISPONDE`]->(_25)
```

Graphstyle.grass: foglio di stile per la visualizzazione del grafo (qui disposto su 3 colonne).

```
node {
  diameter: 50px;
  color: #A5ABB6;
  border-color: #9AA1AC;
  border-width: 2px;
  text-color-internal: #FFFFFF;
  font-size: 10px;
}

relationship {
  color: #A5ABB6;
  shaft-width: 1px;
  font-size: 8px;
  padding: 3px;
  text-color-external: #000000;
  text-color-internal: #FFFFFF;
  caption: '<type>';
}

node.Certificato {
  color: #DE9BF9;
  border-color: #BF85D6;
  text-color-internal: #FFFFFF;
  caption: '{Nome}';
  diameter: 80px;
}

node.Elemento {
  color: #FFD86E;
  border-color: #EDBA39;
  text-color-internal: #604A0E;
  caption: '{Nome}';
  diameter: 80px;
}

node.Categoria {
  color: #68BDF6;
  border-color: #5CA8DB;
  text-color-internal: #FFFFFF;
  caption: '{Nome}';
  diameter: 65px;
}

node.Sottocategoria {
  color: #FF756E;
  border-color: #E06760;
  text-color-internal: #FFFFFF;
  caption: '{Nome}';
  diameter: 65px;
}

node.Attività {
  color: #6DCE9E;
  border-color: #60B58B;
  text-color-internal: #FFFFFF;
  caption: '{Nome}';
  diameter: 80px;
}

node.Lavoro {
  color: #A5ABB6;
  border-color: #9AA1AC;
  text-color-internal: #FFFFFF;
  caption: '{Nome}';
  diameter: 80px;
}

node.prova {
  color: #A5ABB6;
  border-color: #9AA1AC;
  text-color-internal: #FFFFFF;
  caption: '<id>';
}

relationship.HA {
  color: #6DCE9E;
  border-color: #60B58B;
  text-color-internal: #FFFFFF;
}

relationship.RICHIEDE {
  shaft-width: 8px;
  color: #68BDF6;
  border-color: #5CA8DB;
  text-color-internal: #FFFFFF;
}

relationship.SUCCESSIVO {
  shaft-width: 8px;
  color: #FF756E;
  border-color: #E06760;
  text-color-internal: #FFFFFF;
}

relationship.CORRISPONDE {
  shaft-width: 8px;
  color: #6DCE9E;
  border-color: #60B58B;
  text-color-internal: #FFFFFF;
}

relationship.APPARTIENE {
  color: #6DCE9E;
  border-color: #60B58B;
  text-color-internal: #FFFFFF;
  shaft-width: 5px;
}
```

Bibliografia

[1] M. Niño, R.V. Zicari, T. Ivanov, K. Hee, N. Mushtaq, M. Rosselli, C. Sánchez-Ocaña, K. Tolle, J.M. Blanco, A. Illarramendi, J. Besier, H. Underwood, Data Projects for Social Good: Challenges and Opportunities, ICCSS 2017 Int. Conf. Comput. Soc. Sci. Amst. Neth. May 14-15 2017. 3 (2017) 3102.

[2] M.S. Knowles, M. Fedeli, Self-directed learning. Strumenti e strategie per promuoverlo, Franco Angeli, Milano, 2014.

[3] Chat bot, Wikipedia. (2016). https://it.wikipedia.org/w/index.php?title=Chat_bot&oldid=84809172 (accessed February 9, 2017).

[4] Messenger Platform - Documentation, Facebook Dev. (n.d.). https://developers.facebook.com/docs/messenger-platform/ (accessed February 20, 2017).

[5] API.AI, API.AI. (n.d.). https://docs.api.ai/docs.api.ai (accessed February 20, 2017).

[6] Neo4j Documentation, Neo4j Graph Database. (n.d.). https://neo4j.com/docs/ (accessed February 20, 2017).

[7] Getting Started with GrapheneDB · GrapheneDB Documentation, GrapheneDB Doc. (n.d.). http://docs.graphenedb.com/docs.graphenedb.com/docs (accessed February 20, 2017).

[8] Reference | Heroku Dev Center, (n.d.). https://devcenter.heroku.com/categories/reference (accessed February 20, 2017).

[9] M. Scarcia, Chatbot: cosa sono e come usarli per guadagnare di più, SEMrush Blog. (n.d.). https://it.semrush.com/blog/chatbot-cosa-sono/ (accessed February 9, 2017).

[10] Computing Machinery and Intelligence A.M. Turing, (n.d.). http://loebner.net/Prizef/TuringArticle.html (accessed February 24, 2017).

[11] J. Bergant, ChatBots: Messenger ChatBot with API.AI and Node.JS, Udemy. (n.d.). https://www.udemy.com/chatbots/ (accessed February 9, 2017).

[12] Facebook for Developers, F8 2016 Day 1 Keynote - videos, Facebook Dev. (n.d.). https://developers.facebook.com/videos/f8-2016/keynote/ (accessed February 9, 2017).

[13] Facebook for Developers, Building your Messenger Bot - Videos, Facebook Dev. (n.d.). https://developers.facebook.com/videos/f8-2016/building-your-messenger-bot/ (accessed February 9, 2017).

[14] Conversational UX Platform for products and services - API.AI, (n.d.). https://api.ai/ (accessed February 9, 2017).

[15] Writing the Script: Conversation Mapping | Botdesign.ai, (n.d.). http://botdesign.ai/thesis/conversation-mapping/46 (accessed February 9, 2017).

[16] J.P.E. by P. Economy, User Story Mapping: Discover the Whole Story, Build the Right Product, 1st ed., O'Reilly and Associates, Beijing ; Sebastopol, CA, 2014.

[17] E. Mazzilli, the thing that makes me more excited to come to work every day, facebook.com. (n.d.). https://www.facebook.com/emanuelconunaemme/posts/10211532367448441?match=dGhlIHRoaW5nIHRoYXQgbWFrZXMsZW1hbnVlbCBtYXp6aWxsaSxtYXp6aWxsaSxlbWFudWVsLG1ha2VzLHRoaW5n.

[18] C. King, Do all digital assistants suck? Or is it just me?, Inphantry.com. (n.d.). http://www.inphantry.com/blog/2015/do-all-digital-assistants-suck-or-is-it-just-me/ (accessed February 9, 2017).

[19] C. Carylyne, Why does your Chatbot Suck? | UX Magazine, (n.d.). https://uxmag.com/articles/why-does-your-chatbot-suck (accessed February 9, 2017).

[20] A. Tate, I cinque Chatbot di Facebook Messenger a cui ispirarsi, AdEspresso. (2016). https://adespresso.com/it/academy/blog/5-migliori-chatbot-facebook-messenger/

(accessed February 9, 2017).
[21] Base di dati a grafo, Wikipedia. (2016). https://it.wikipedia.org/w/index.php?title=Base_di_dati_a_grafo&oldid=84344998 (accessed February 9, 2017).
[22] Grafo, Wikipedia. (2016). https://it.wikipedia.org/w/index.php?title=Grafo&oldid=82183572 (accessed February 9, 2017).
[23] I. Robinson, J. Webber, E. Eifrem, Graph Databases: New Opportunities for Connected Data, 2 edizione, O'Reilly Media, Beijing, 2015.
[24] Neo4j - The Fastest and Most Scalable Native Graph Database, Intro to Neo4j and Graph Databases, (23:50:41 UTC). http://www.slideshare.net/neo4j/intro-to-neo4j-and-graph-databases (accessed February 9, 2017).
[25] David Fombella Pombal, Neo4j Introduction (Basics, Cypher, RDBMS to GRAPH), (15:35:00 UTC). http://www.slideshare.net/DavidFombellaPombal/neo4j-introduction-basics-cypher-rdbms-to-graph (accessed February 19, 2017).
[26] Neo4j's Cypher queries cheatsheet · GitHub, (n.d.). https://gist.github.com/DaniSancas/1d5265fc159a95ff457b940fc5046887 (accessed February 19, 2017).
[27] torta_matteo_tesi.pdf, (n.d.). http://amslaurea.unibo.it/7410/1/torta_matteo_tesi.pdf (accessed February 20, 2017).
[28] Hosting Neo4j in the Cloud, Neo4j Graph Database. (n.d.). https://neo4j.com/developer/guide-cloud-deployment/ (accessed February 10, 2017).
[29] Heroku, Wikipedia. (2016). https://it.wikipedia.org/w/index.php?title=Heroku&oldid=83428339 (accessed February 10, 2017).
[30] Cloud Application Platform | Heroku, (n.d.). https://www.heroku.com/ (accessed February 10, 2017).
[31] D. Beneventano, S. Bergamaschi, F. Guerra, Progetto di basi di dati relazionali. Lezioni ed esercizi, 2 edizione, Pitagora, Bologna, 2007.
[32] R. Carli, R.M. Paniccia, L'analisi emozionale del testo. Uno strumento psicologico per leggere testi e discorsi, FrancoAngeli, 2004.
[33] B.R. Marshall, Le parole sono finestre (oppure muri). Introduzione alla comunicazione nonviolenta, Esserci, Reggio Emilia, 2015.
[34] Dialog tree, Wikipedia. (2017). https://en.wikipedia.org/w/index.php?title=Dialog_tree&oldid=763521859 (accessed February 11, 2017).
[35] G. Dezza, G. Narrante, Arte di vendere, arte di investigare, FrancoAngeli Milano. (2010).
[36] graphgist, Neo4j Graph Database. (n.d.). https://neo4j.com/graphgist/ (accessed February 12, 2017).
[37] F. Bancilhon, ed., Building an object-oriented database system: the story of O 2, Morgan Kaufmann Publ, San Mateo, Calif, 1992.
[38] R. Kharwar, M. Hunger, N.T.| M. 17, 2016, Cypher: How to Rewrite a UNION Query Using a COLLECT Clause, Neo4j Graph Database. (2016). https://neo4j.com/blog/cypher-union-query-using-collect-clause/.
[39] K. Hee, R.V. Zicari, K. Tolle, Tailored Data Science Education using Gamification, in: 3rd RDA Workshop Curricula Teach. Methods Cloud Comput. Big Data Data Sci. Part 8th IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2016 Luxemb. Dec. 12 2016, 2016. http://2016cloudcom.ux.uis.no/conf/workshops/data_teaching.html.
[40] P. Wallbank, Steve Jobs' golden path, Decod. New Econ. (2013). http://paulwallbank.com/2013/10/11/steve-jobs-golden-path/ (accessed February 26, 2017).