

Chatbot with Watson

Agenda

- Sistemi Cognitivi
- Watson API
- ChatBot e Agenti Virtuali
- IBM Cloud
- Flow-Based Programming e Node-Red
- Watson Assistant
- Telegram Bot (Tutorial)

Emerging Patterns for Artificial Intelligence adoption

Text Mining is a class of functions for parsing and identifying significant words in language (NLP) as well as understand the semantic of a textual content



Multi-Media Mining is a class of function for analyzing visual content such as images or videos



Speech Mining is a class of functions for analyzing audio signals including speech to such as ability



<XXX> Mining is class of large specialized functions for analyzing "digital representation" in a specific domain → e.g., Bioinformatics, Financial Analytics, etc.



Rule-based Semantic Search and Analytics

systems use statistical techniques for detecting patterns or detect trends within data, yield an understanding of historical or current state from which to draw conclusions

Cognitive Systems

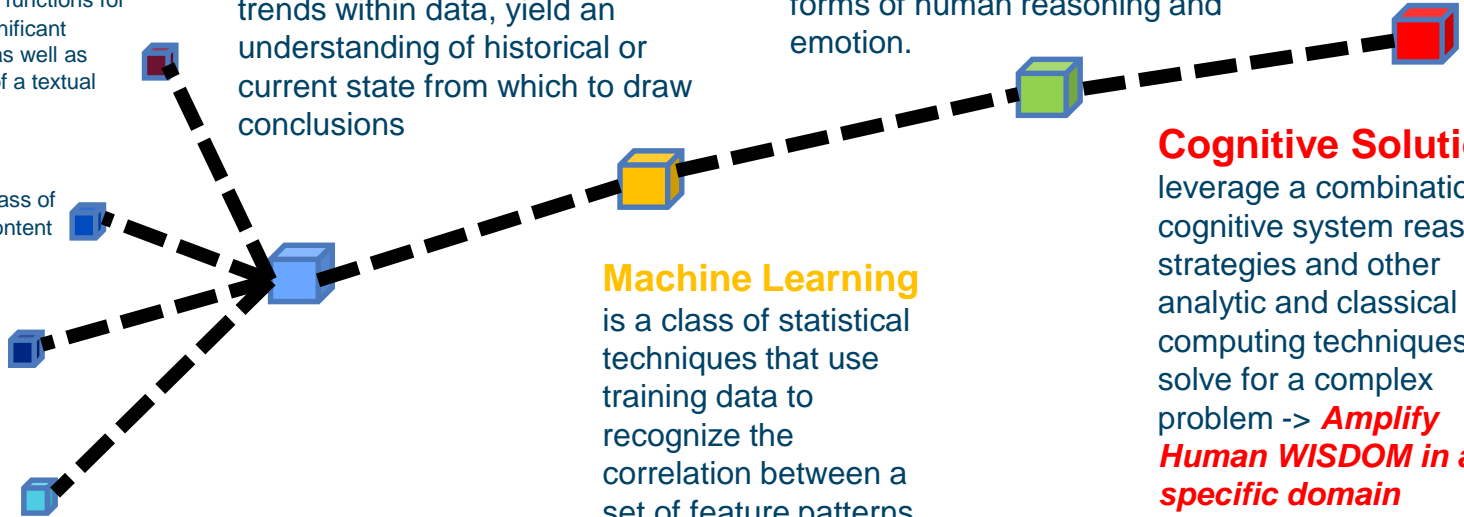
leverage machine learning to predict meaning in features of human language (spoken, written, visual) and related forms of human reasoning and emotion.

Machine Learning

is a class of statistical techniques that use training data to recognize the correlation between a set of feature patterns and outcomes.

Cognitive Solutions

leverage a combination of cognitive system reasoning strategies and other analytic and classical computing techniques to solve for a complex problem -> **Amplify Human WISDOM in a specific domain**



I sistemi cognitivi

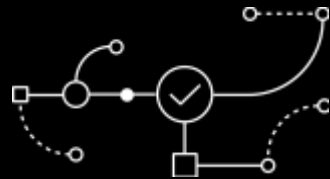
Le capacità che differenziano i i sistemi cognitivi dai sistemi tradizionali sono le seguenti:

CAPIRE



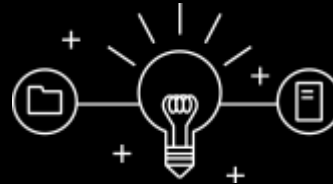
I sistemi cognitive sono in grado di capire il linguaggio umano, immagini fisse ed in movimento e ogni altra forma di dato non strutturato, esattamente **come un essere umano**

RAGIONARE



Possono **estrarre elementi significativi**, cogliere concetti, formulare ipotesi e **proporre soluzioni**.

IMPARARE



Nel corso di ogni interazione sono in grado di affinare e sviluppare le proprie competenze, in pratica **non smettono mai di imparare**.

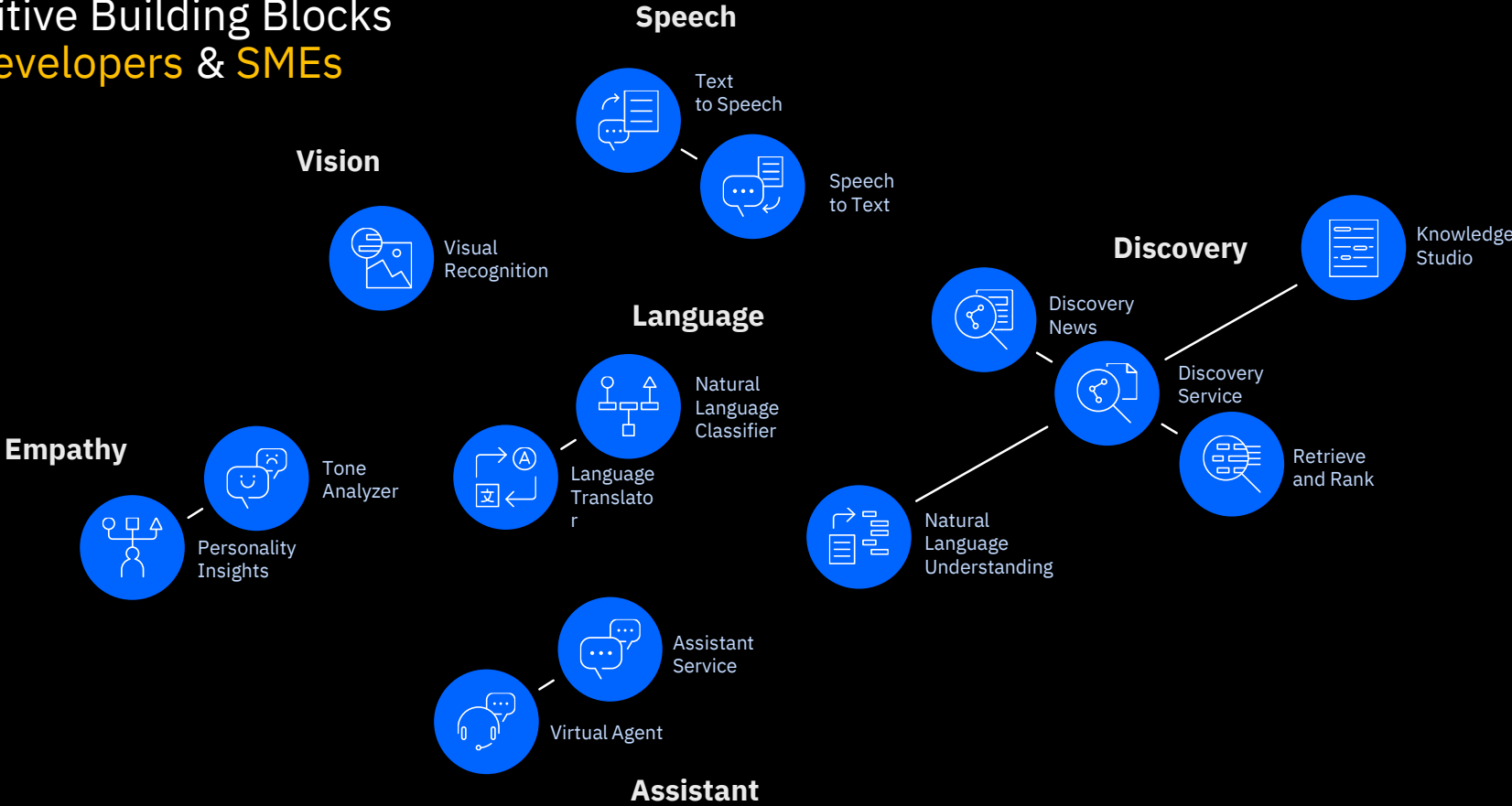
INTERAGIRE



Grazie alle proprie capacità di vedere, ascoltare e parlare, I sistemi cognitive sono in grado di **interagire con le persone in modo del tutto naturale**

Watson APIs:

Cognitive Building Blocks for **Developers** & **SMEs**

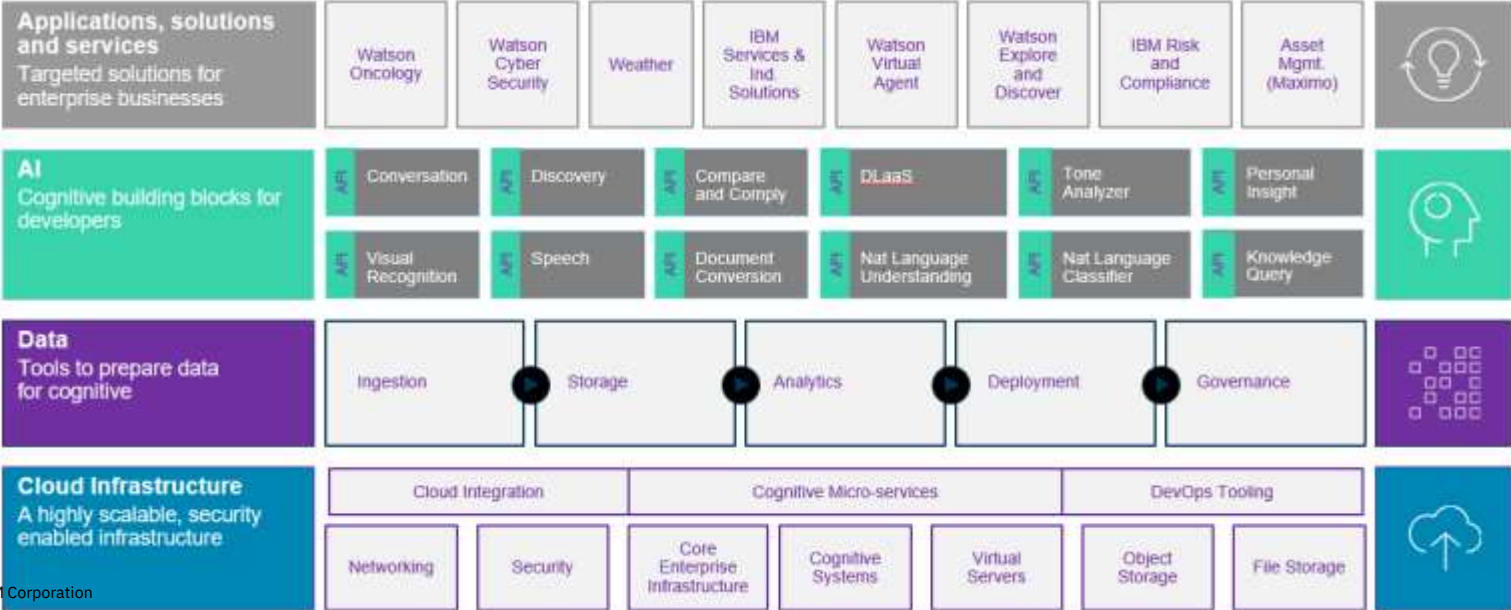


IBM Cloud



IBM Cloud è una piattaforma ibrida fornita da IBM e basata su CloudFoundry

Unendo platform e infratructure services, IBM Cloud supporta vari linguaggi di programmazione (Java, Node.js, Go, PHP, Python, Ruby on Rails) ed offre un ricco assortimento di servizi IBM e di terze parti in continua espansione.



Che cos'è un chatbot?

Gli agenti virtuali e i chatbot sono **applicazioni** che consentono di creare **dialoghi interattivi con linguaggio naturale**.

Attraverso le chatbot è possibile soddisfare puntualmente le richieste dell'utente creando una **conversazione in linguaggio naturale** e nello stesso tempo 'raccogliere' tutte le informazioni inserite dall'utente per indirizzare al meglio le reazioni del sistema

Flow-Based Programming

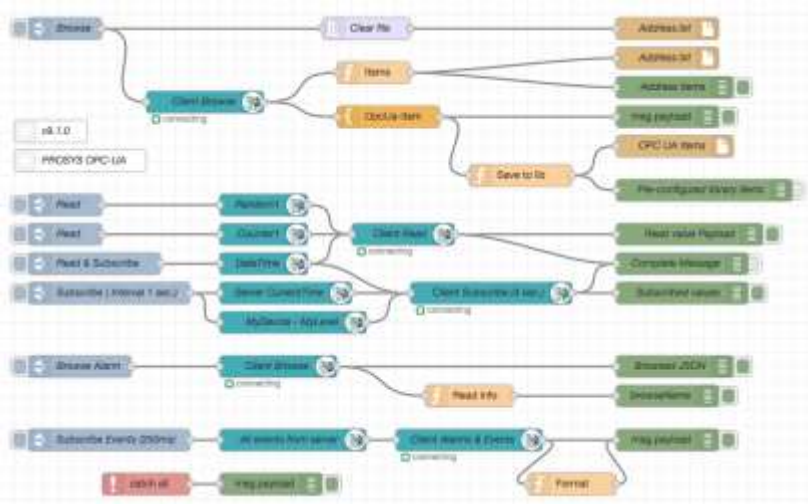
Negli ultimi anni si è diffuso un nuovo paradigma chiamato **Flow-Based Programming** (Programmazione basata su flussi) che consente di realizzare un programma in modo visuale collegando tra loro dei box.

In questo paradigma **un programma è costituito da una serie di blocchi** (black box) di cui si conosce il tipo di elaborazione eseguita (quindi la funzionalità realizzata e le specifiche di input e output) ma non l'implementazione interna.

Un programma diventa quindi non una sequenza di istruzioni come nel caso dei paradigmi classici, ma un insieme di flussi di dati che vengono scambiati tra i blocchi

Node-Red

Lo strumento più noto nell'ambito del Flow-Based Programming è chiamato **Node-RED**: sviluppato in node js, può essere utilizzato in locale o sulla piattaforma IBM Cloud.



Documentazione: <https://nodered.org/docs/getting-started/>

Watson Assistant

Tra le API di Watson disponibili su IBM Cloud, è presente Watson Assistant, uno strumento grafico flessibile e facile da usare per realizzare dialoghi interattivi con l'utente.

- **Api Reference:**
<https://console.bluemix.net/apidocs/assistant>
- **Documentazione:**
<https://console.bluemix.net/docs/services/assistant/index.html#about>



Tutorial

Telegram Bot

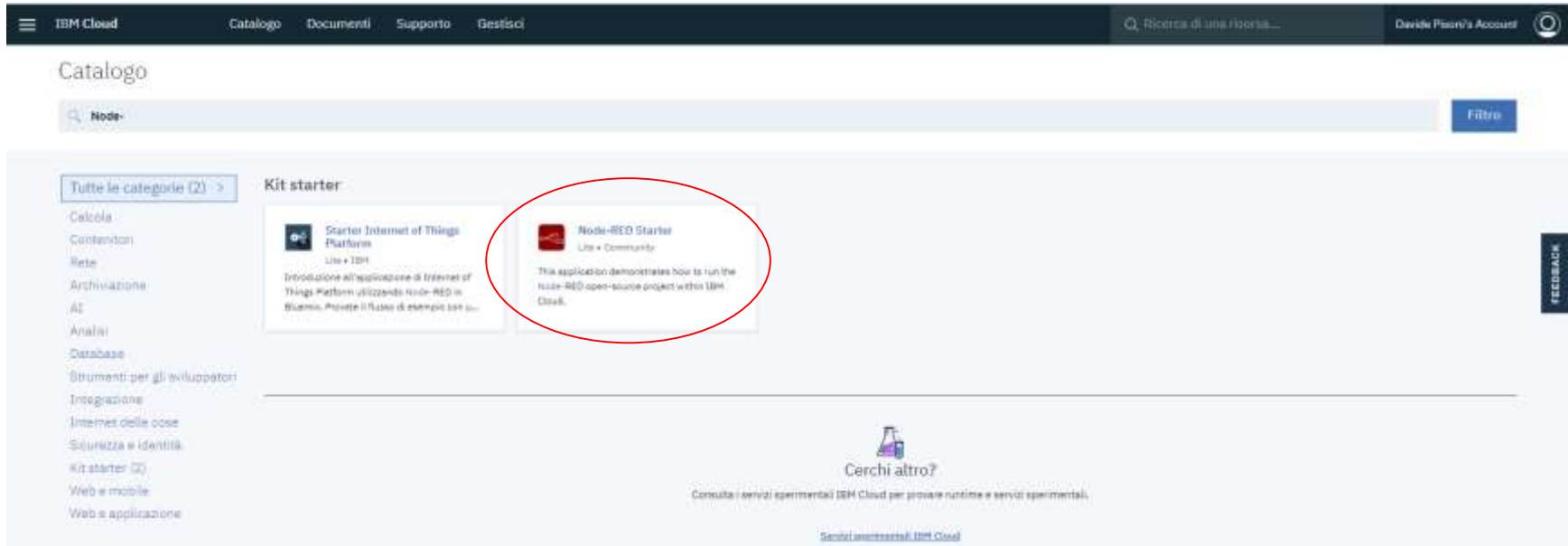
Oggi creremo un piccolo chatbot utilizzando **Node Red** e **Watson Assistant**

Istruzioni: Come creare un Bot su Telegram

1. Creare un'applicazione in Node Red su IBM Cloud
2. Creare un servizio Watson Assistant su IBM Cloud e connetterlo alla propria applicazione
3. Creare il flusso di dialogo nel workspace di Assistant
4. Creare un bot su telegram
5. Integrare il bot con il flusso di Assistant tramite Node Red

Creare l'applicazione Node Red

Dopo aver effettuato il login su IBM Cloud, andare nel **catalogo** e selezionare, tra i contenitori tipo, il **Node-RED Starter**



The screenshot displays the IBM Cloud Catalog interface. At the top, there is a navigation bar with 'IBM Cloud', 'Catalogo', 'Documenti', 'Supporto', and 'Gestisci'. A search bar contains 'Node-' and a 'Filtro' button is on the right. Below the search bar, a sidebar on the left lists various categories like 'Calcolo', 'Contenitori', 'Rete', etc. The main content area is titled 'Kit starter' and features two cards. The first card is 'Starter Internet of Things Platform' and the second, 'Node-RED Starter', is circled in red. The 'Node-RED Starter' card includes a red icon, the text 'Node-RED Starter' with 'Lib + Community' below it, and a description: 'This application demonstrates how to run the Node-RED open-source project within IBM Cloud.' At the bottom of the page, there is a 'Cerchi altro?' section with a link to 'Scopri i servizi sperimentali IBM Cloud'.

Inserire il nome, selezionare lo spazio e la regione dove crearla

Crea un'applicazione Cloud Foundry

Starter Internet of Things Platform

Introduzione all'applicazione di Internet of Things Platform utilizzando Node-RED in Bluemix. Provatelo il flusso di esempio con un simulatore e personalizzatelo per le vostre periferiche.

Libre IBM

Visualizza documenti

VERSIONE 0.7.0
TIPO Containere spo
REGIONE Regno Unito, Germania, Stati Uniti, Guai

Nome applicazione:

Inniedi un nome univoco

Nome host:

Inniedi un nome univoco

Domain:

Seleziona la regione in cui distribuire:

Regno Unito

Scegli un'organizzazione:

Caricamento in corso...

Scegli uno spazio:

Caricamento in corso...

Piano selezionato:

SDK for Node.js™

Predifinito

Cloudant NoSQL DB

Internet of Things Platform



SDK for Node.js™



Cloudant NoSQL DB



Internet of Things Platform

Dashboard di IBM Cloud

Dashboard

GRUPPO DI RISORSE
Tutte le risorse

ORGANIZZAZIONE CLOUD FOUNDRY
Tutte le organizzazioni

SPAZIO CLOUD FOUNDRY
Tutti gli spazi

REGIONE
X Frankfurt

CATEGORIA
Tutte le categorie

Filtra in base al nome risorsa...

Crea risorsa

Servizi

Nome	Ubicazione	Gruppo di risorse	Piano	Dettagli	Offerta di servizi
Cloudant-dm	Francoforte	default	Lite	Con Provisioning	Cloudant

Applicazioni Cloud Foundry

Nome	Regione	Organizzazione CF	Spazio CF	Memoria (MB)	Stato
Cooltech	Francoforte	davide.pisoni@it.ibm.com	Europe1	512	In Esecuzione (3/3)
TelegramBotDavide	Francoforte	davide.pisoni@it.ibm.com	Europe1	256	In Esecuzione (3/3)
UnIBresciaLesson	Francoforte	davide.pisoni@it.ibm.com	Europe1	512	In Esecuzione (3/3)

Servizi Cloud Foundry

Nome	Regione	Organizzazione CF	Spazio CF	Piano	Offerta di servizi
Continuous Delivery	Francoforte	davide.pisoni@it.ibm.com	Europe1		
Conversation-xt	Francoforte	davide.pisoni@it.ibm.com	Europe1	free	conversation
Cooltech-cloudantNoSQLDB	Francoforte	davide.pisoni@it.ibm.com	Europe1	Lite	cloudantNoSQLDB
Language Translator-ib	Francoforte	davide.pisoni@it.ibm.com	Europe1	standard	language_translator

Configurare Node Red

Dopo che l'applicazione si è avviata, seguire l'url associato ad essa: nome-applicazione.eu-de.mybluemix.net

Si verrà portati sulla pagina iniziale di node red, dalla quale partirà la configurazione dello spazio dove verranno chieste una username e una password per proteggere il proprio flusso

Finito questo si arriverà alla schermata dei flussi di Node red



Creare un servizio di Watson Assistant

Dopo aver effettuato il login su IBM Cloud, andare nel **catalogo** e selezionare, tra le API di Watson il servizio **Watson Assistant**

The screenshot displays the IBM Cloud AI catalog interface. The 'AI' section is visible at the top left. A red circle highlights the 'Watson Assistant' service card. The card for 'Watson Assistant' includes the following text:

- Watson Assistant** (precedentemente noto come Conversation)
- Lib + IBM
- Aggiungere un'interfaccia di linguaggio naturale all'applicazione per automatizzare le interazioni con gli utenti finali. Tra le...

Other visible services in the catalog include:

- AI OpenScale** (Lib + IBM): IBM AI OpenScale is an enterprise-grade environment for AI infused applications that provides enterprises with visibility into how...
- Compare Comply** (Lib + IBM + Beta): Process governing documents to convert, identify, classify, and compare important elements.
- Discovery** (Lib + IBM): Sblocca il valore nascosto nei dati per trovare le risposte, monitorare gli andamenti e i pattern di superficie con il motore di insight...
- Knowledge Catalog** (Lib + IBM): Discover, catalog, and securely share enterprise data.
- Knowledge Studio** (Lib + IBM): Insegna a Watson il linguaggio del dominio.
- Language Translator** (Lib + IBM): Tradurre testi, documenti e siti web da una lingua in un'altra. Creare traduzioni specifiche per settore e regione tramite la...
- Machine Learning** (Lib + IBM): IBM Watson Machine Learning - prendi decisioni più intelligenti, risolvi problemi difficili e migliora l'esperienza dell'utente.
- Natural Language Classifier** (Lib + IBM): Natural Language Classifier effettua la classificazione del linguaggio naturale nel testo delle domande. Un utente sarà in grad...
- Natural Language Understanding** (Lib + IBM): Analizza il testo per estrarre i metodi del contenuto come ad esempio concetti, entità, sentimenti, relazioni, opinioni ed altro.
- Personality Insights** (Lib + IBM): Watson Personality Insights decodifica le informazioni dai dati dei media sociali transazionali per identificare le...
- Speech to Text** (Lib + IBM): Trascrizione in streaming a bassa latenza.
- Text to Speech** (Lib + IBM): Sintetizza dal testo un parlato dal suono naturale.
- Tone Analyzer** (Lib + IBM): Tone Analyzer utilizza l'analisi linguistica per rilevare tre tipi di toni dalle comunicazioni: emozione, suavia e ligua. Queste...
- Visual Recognition** (Lib + IBM): Trova il significato di un contenuto visivo. Analizza le immagini alla ricerca di scene, oggetti, volti e altro contenuto. Segli un...
- Watson Studio** (Lib + IBM): Embed AI and machine learning into your business. Create custom models using your own data.
- PowerAI** (Terra part): The accelerated deep learning platform for enterprises. Built on the IBM PowerAI platform, powered by Nimble.

Collegare il servizio alla propria applicazione

Nella **Panoramica** della propria applicazione, cliccare su **Crea connessione** e selezionare il servizio Assistant appena creato (l'applicazione dovrà essere riavviata a seguito di questa operazione)

The screenshot displays the IBM Cloud console interface for an application named 'UniBresciaLesson'. The main dashboard includes several key metrics:

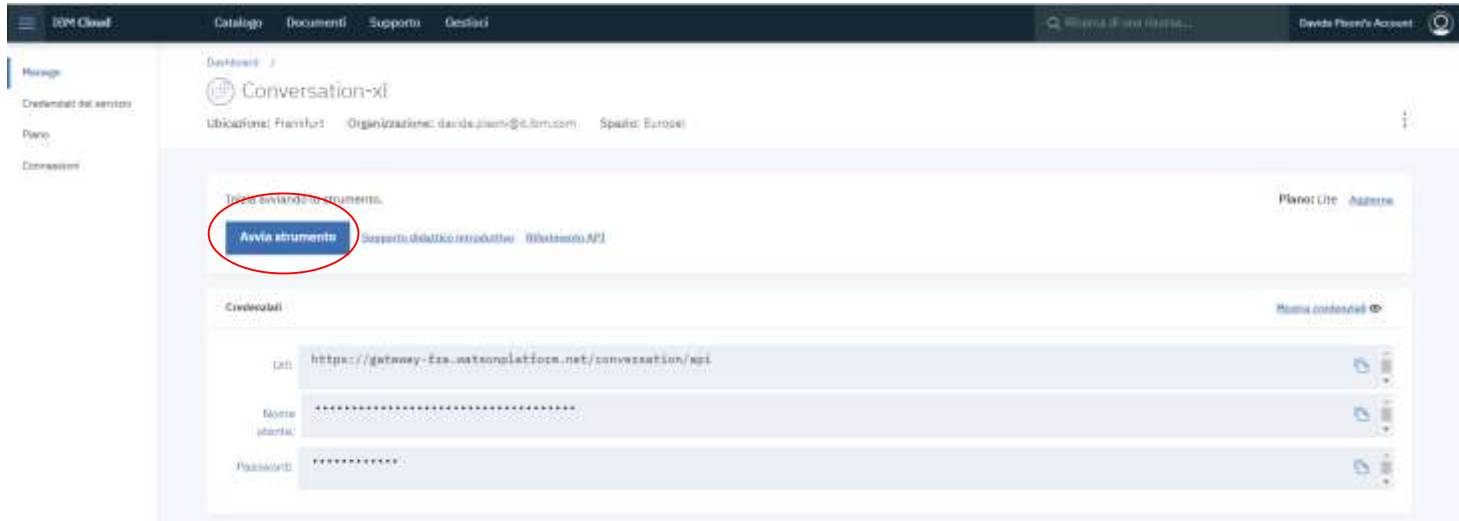
- RICCHETTA BUILD**: 0 in 10 secondi
- SERVIZI**: 1 (Tutti i servizi sono in esecuzione a 100%)
- MEMO INDIRIZZI PER STANZA**: 512
- ASSOCIAZIONE TOTALE PER**: 512 (17.185 eventi ricevuti)

The 'Connessioni (1)' section shows a single connection entry with a 'Crea connessione' button highlighted by a red circle. The 'Costo corrente' section displays a balance of 0,00 €.

Creazione del flusso di dialogo su Assistant

Per accedere al workspace di Assistant, cliccare sul servizio e poi su **Launch Tool**

Accedere utilizzando le stesse credenziali di IBM Cloud



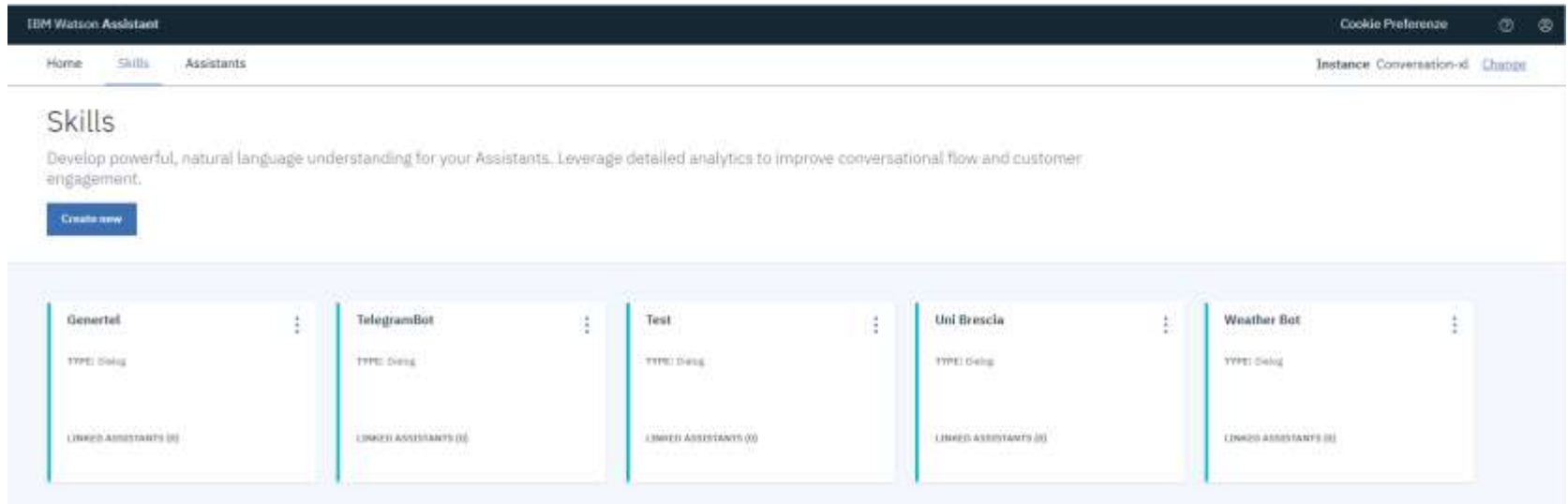
The screenshot displays the IBM Cloud console interface for the 'Conversation-xi' service. The top navigation bar includes 'IBM Cloud', 'Catalogo', 'Documenti', 'Supporto', and 'Credenziali'. A search bar and 'David's Account' are visible on the right. The left sidebar shows 'Messaggi', 'Credenziali del servizio', 'Piano', and 'Commissari'. The main content area shows the service details for 'Conversation-xi' with 'Localit : Frankfurt', 'Organizzazione: david.zileri@ibm.com', and 'Spazio: Europe'. Below this, there is a section for 'Tutti gli strumenti' with a table containing one entry: 'Avvia strumento' (circled in red), 'Crea un nuovo strumento', and 'Invia API'. Below the table is a 'Credenziali' section with a 'Mostra credenziali' link. The credentials are listed as: 'URL: https://gateway-esa-watsonplatform.net/conversation/api', 'Nome utente:,', and 'Password:

Il Workspace

Una volta entrati, cliccare su **Skills** nel menu in alto e creare un nuovo workspace, inserendo il nome e la lingua desiderata per il dialogo

Una volta creato cliccarci sopra per entrare nel workspace

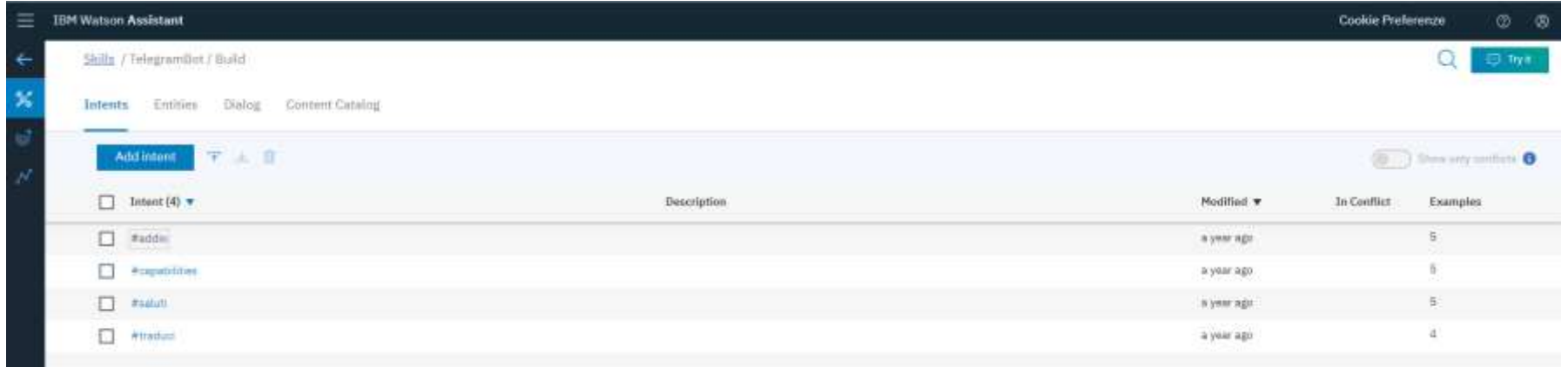
Il workspace all'interno è strutturato su tre tab: Intents, Entities e Dialog



Intenti - #nome_intento

Per creare le conversazioni si parte dagli **intenti** che servono a **categorizzare** l'input dell'utente, incanalando le diverse tipologie di richieste

Per ogni intento è necessario istruire Watson con frasi di esempio per far capire a Watson come identificare l'intento stesso.



The screenshot shows the IBM Watson Assistant interface for a skill named 'TelegramBot'. The 'Intents' tab is selected, displaying a table of existing intents. The table has columns for 'Intent (4)', 'Description', 'Modified', 'In Conflict', and 'Examples'. There are four intents listed: '#addit', '#cagatrinze', '#saluti', and '#traduc'. Each intent has a checkbox, a description, a 'Modified' date (all 'a year ago'), an 'In Conflict' status, and a number of 'Examples'.

Intent (4)	Description	Modified	In Conflict	Examples
<input type="checkbox"/> #addit		a year ago		5
<input type="checkbox"/> #cagatrinze		a year ago		5
<input type="checkbox"/> #saluti		a year ago		5
<input type="checkbox"/> #traduc		a year ago		4

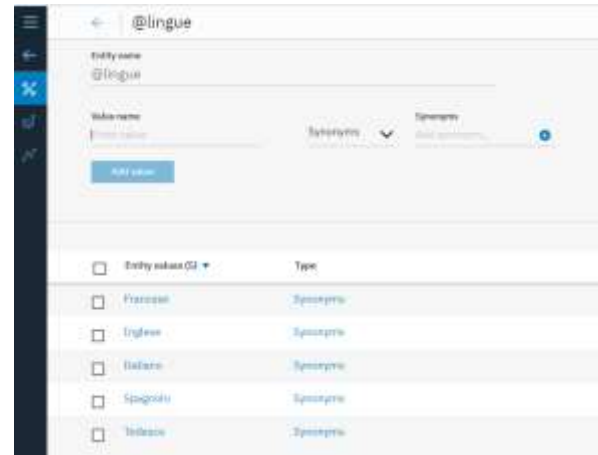
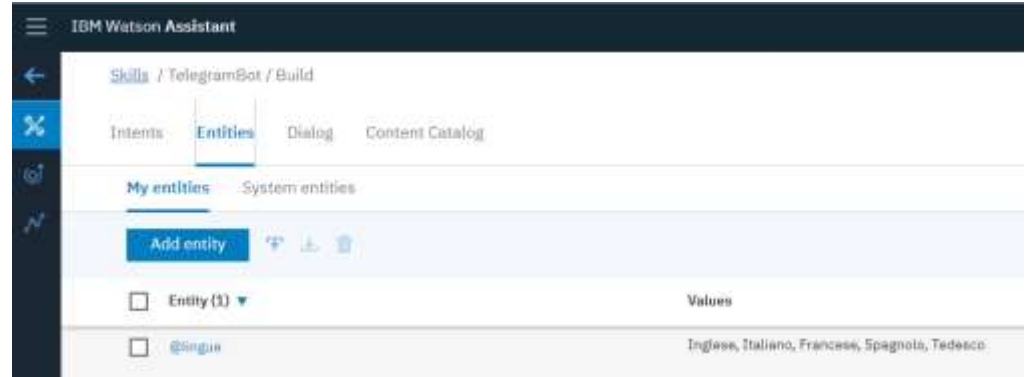
Entità - @nome_entità

Successivamente si stabiliscono le **entità**, ovvero dei **valori specifici** (*parole*) da cercare nell'input con vari sinonimi per ciascun valore

Esistono delle entità già presenti a sistema, come numeri e date

Per ogni entità è possibile attivare il **fuzzy matching** per ogni entità, che consente di riportare all'entità anche parole scritte sbagliate o che comunque distano dal valore o dal sinonimo di una o due lettere

Oltre ai sinonimi, è possibile anche inserire delle **regular expression** per catturare determinati valori (e.g. Numeri telefonici)



Il dialogo

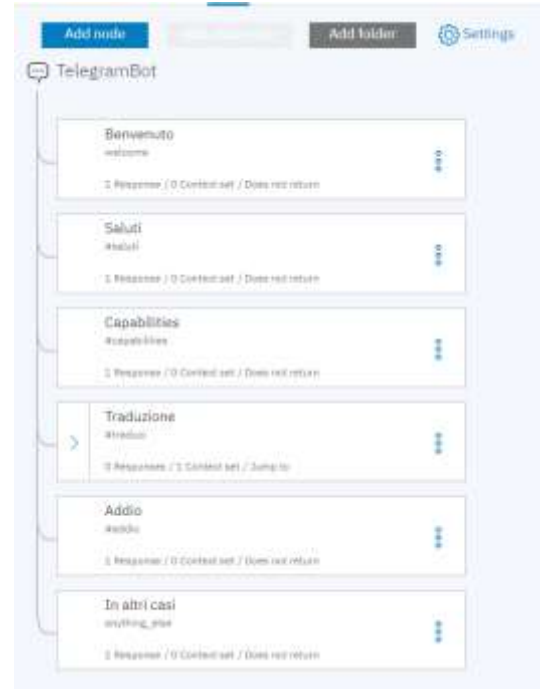
Dopo aver definito intenti ed entità si costruisce il dialogo in maniera grafica attraverso una serie di nodi collegati tra loro.

Ogni nodo conterrà una condizione che verrà valutata per accedere o meno al suo output

I nodi vengono sempre valutati dall'alto verso il basso

Solitamente il dialogo viene strutturato nel seguente modo:

- la prima fila di nodi contiene condizioni sugli intenti per intercettare la tipologia di richiesta dell'utente
- sotto ad ognuno di essi vi sono una serie di nodi figli con condizioni legate alle entità per dare la risposta più azzeccata alla domanda dell'utente



Slot, Contesto e Jump To

Altre 3 funzionalità molto importanti all'interno di Assistant sono: **gli Slot, il Contesto e i Jump To**

- **Il contesto** è una variabile globale presente all'interno del workspace nella quale si possono salvare tutte le informazioni che sono state recuperate dall'utente

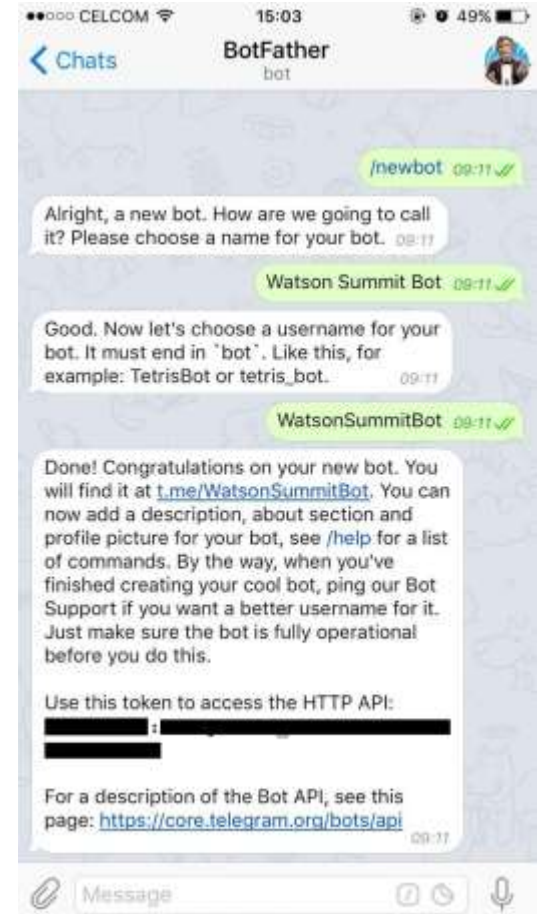
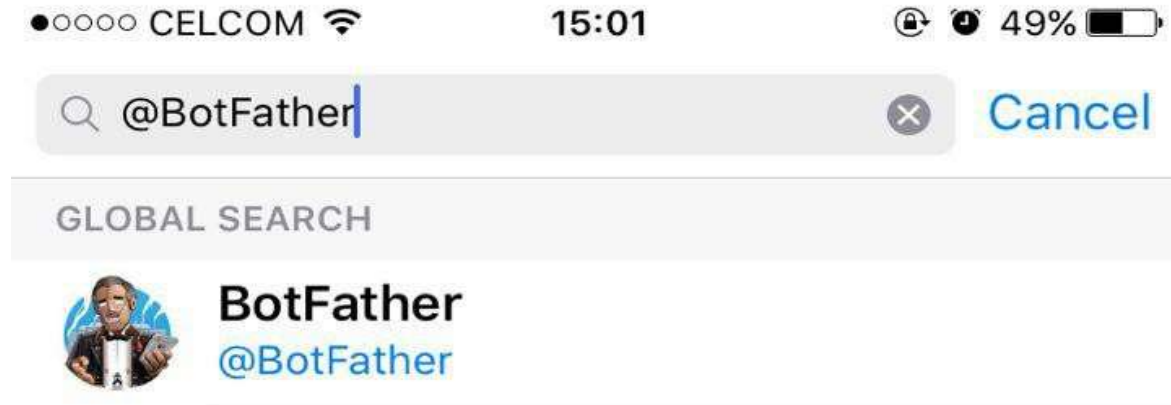
***Esempio:** vi si può salvare l'ultimo intento di cui si è parlato con l'utente, così che nel caso non venga espresso esplicitamente nella domanda si riesca comunque a dare una risposta all'utente*

- **I jump to** sono funzioni all'interno dei nodi che permettono di saltare da un nodo all'altro senza dover seguire tutto il flusso creato. Possono essere utili per riportare alcuni flussi ad altri che hanno la stessa chiusura o trattano le stesse risposte, senza dover ricreare ogni volta i nodi
- **Gli Slot** sono sempre delle funzioni all'interno dei nodi che permettono di *ciclare* su un nodo fino a quando non si sono raccolte tutte le informazioni volute

***Esempio:** se serve recuperare le informazioni per un viaggio bisognerà chiedere all'utente il luogo di partenza, quello di arrivo, l'orario, la data....*

Creare un bot su Telegram

- Sull'app di telegram cercare @BotFather sulla barra in alto
- Scrivergli /newbot ed in seguito inserire l nome e l'username del bot
- Salvare il token che verrà visualizzato per poter inviare/richiamare il bot



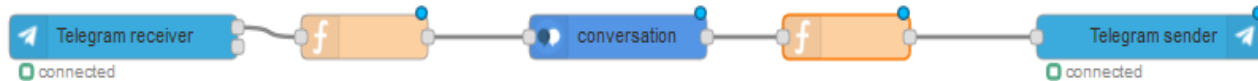
Installare su Node Red nuovi nodi

- Nel menu in alto destra selezionare **Manage Palette**
- Selezionare la tab **Install** e cercare Telegram
- Cliccare sul bottone install per i pacchetti che si vogliono installare
- I nuovi nodi compariranno nella barra a sinistra una volta finita l'installazione



Creare il flusso su Node Red (1/2)

- Per gestire la chat useremo tre nodi: uno per ricevere i messaggi inviati al bot, uno per inviarli all' API Assistant e uno per inviarli al bot
- I nodi sono i seguenti:
 - *Telegram Receiver*: riceve i messaggi inviati al bot
 - *Assistant*: invia l'input al servizio di conversazione e restituisce l'output
 - *Telegram Sender*: invia i messaggi al bot
- I due nodi di telegram vanno configurati inserendo il token creato per il bot, mentre il nodo di conversazione con le credenziali del servizio (se è connesso all'applicazione si autoconfigurerà) e il workspace id del dialogo



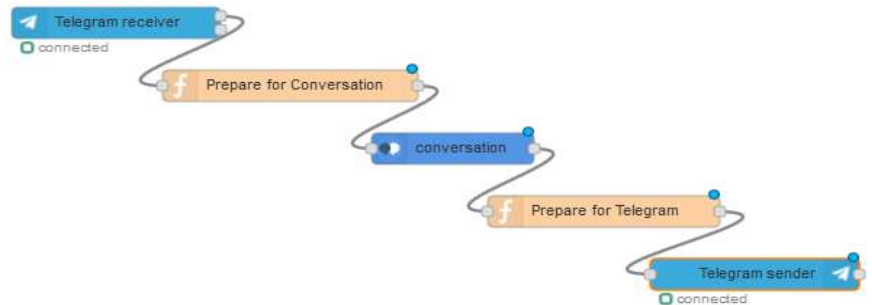
Creare il flusso su Node Red(2/2)

- Tra i vari nodi inseriremo delle funzioni che formatteranno i vari output in modo tale da renderli adatti ai nodi di conversation e telegram
- Nella prima funzione inseriamo il seguente codice (From Telegram To Assistant):

```
msg.chatId = msg.payload.chatId;  
msg.payload = msg.payload.content;  
return msg;
```

- Nella seconda il seguente codice (From Assistant To Telegram):

```
msg.payload = {  
  chatId : msg.chatId,  
  type : "message",  
  content : msg.payload.output.text[0]};  
return msg;
```



Gestire una conversazione

- Il flusso precedente gestisce interazioni singole con l'utente, ma non mantiene la chat, cioè ogni interazione è come se fosse la prima per il bot e non input successivi che fanno parte della stessa conversazione con l'utente
- Per gestire conversazioni più complesse è necessario modificare il flusso in modo che tenga traccia del *conversation_id*, ovvero il codice univoco generato dal servizio Assistant per la conversazione iniziata, e del *Context*, la variabile in cui viene tracciato il flusso seguito finora per ripartire da lì
- Per farlo aggiungiamo una funzione che salvi in una variabile di flusso il contesto:

```
flow.set('context',msg.payload.context)
```

- E modifichiamo la funzione From Telegram To Assistant aggiungendo la seguente riga:

```
msg.payload.context=flow.get('context')
```

Tips & Tricks: Come costruire un buon chatbot

- **Essere chiari sull'obiettivo e le capacità del chatbot:** cercare di essere chiari e concisi verso l'utente su cosa può fare il chatbot, in modo da non creare equivoci o aspettative che non possono essere soddisfatte. Già solo il nome del chatbot dovrebbe essere in grado di chiarire immediatamente le sue funzionalità (quindi va scelto con cura). Questo vale ovviamente anche per qualsiasi informazione venga data all'utente, il chatbot deve essere funzionale.
- **Dare supporto all'utente:** in ogni momento l'utente deve essere in grado di ricevere aiuto su come utilizzare il chatbot, per cui va sempre prevista la possibilità di richiamare un nodo con le istruzioni appropriate per ogni funzionalità
- **L'esperienza è tutto:** per quanti test e training farete, ci sarà sempre qualche frase o possibilità che non si era prevista. Avere la possibilità di farlo provare e testare dal maggior numero di persone diverse è un'ottima opportunità per testare la solidità del training e del flusso

Altre integrazioni: Language Translator (1/2)

- Proviamo ad aggiungere una nuova funzionalità al chatbot che gli permetta di tradurre una frase dell'utente in una lingua target
- Quindi creiamo un servizio di Language Translator e colleghiamolo all'applicazione (come abbiamo fatto per Assistant)
- Una volta fatto creeremo un intento ad hoc per intercettare questa richiesta dell'utente e la lingua nella quale vuole tradurre la sua frase (con un entità apposita sulle lingue). Quando avremo tutte le informazioni nella *foglia* del flusso di dialogo inseriremo una variabile di contesto chiamata action che richiamerà il servizio di traduzione nel backend (`context:{action:translate}`)

Then respond with:

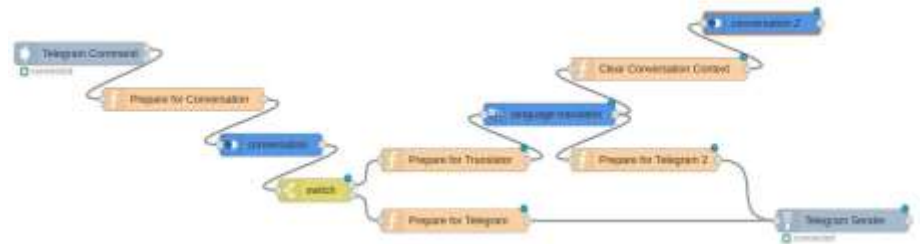
```
1 {  
2   "context": {  
3     "action": "translate"  
4   }  
5   "output": {}  
6 }
```


Altre integrazioni: Language Translator (2/2)

- Infine modifichiamo il flusso node-red come segue: un modulo di switch che catturi la variabile di contesto action quando è settata su *translate*, un modulo di language identify che identifichi la lingua della frase di partenza dell'utente e un modulo di traduzione per tradurre la frase
- Tra il modulo di identificazione e quello di traduzione, inseriremo la seguente funzione per settare tutte le variabili:

```
context=flow.get('context')  
msg.srclang = msg.lang.language;  
msg.destlang = getLanguage(context.destination)  
return msg;
```

```
function getLanguage(lang) {  
  switch (lang) {  
    case "Francese":    return "fr";  
    case "Italiano":   return "it";  
    case "Spagnolo":   return "es"; }  
  return "en";}
```



Altre integrazioni: Visual Recognition (1/2)

- Proviamo ad aggiungere una nuova funzionalità al chatbot che gli permetta di riconoscere le immagini inviate dagli utenti
- Quindi creiamo un servizio di Visual Recognition e colleghiamolo all'applicazione (come abbiamo fatto per Assistant)
- Una volta fatto, sarà tutto il back end ad occuparsi della ricezione delle immagini, della classificazione di queste e di dare una risposta all'utente su cosa ha inviato
- Per cui aggiungiamo un nodo di switch subito dopo il telegram receiver che controlla quando il *msg.payload.type*==*'photo'*, altrimenti il flusso prosegue come prima
- Nel caso sia una foto, aggiungiamo una funzione per inviare la foto a visual recognition con il seguente codice:

```
msg.chatId = msg.payload.chatId  
msg.payload = msg.payload.weblink  
return msg;
```

Altre integrazioni: Visual recognition(2/2)

- Ora aggiungiamo un nodo di visual recognition e subito dopo un'altra funzione con il seguente codice:

```
msg.payload = {
```

```
  chatId: msg.chatId,
```

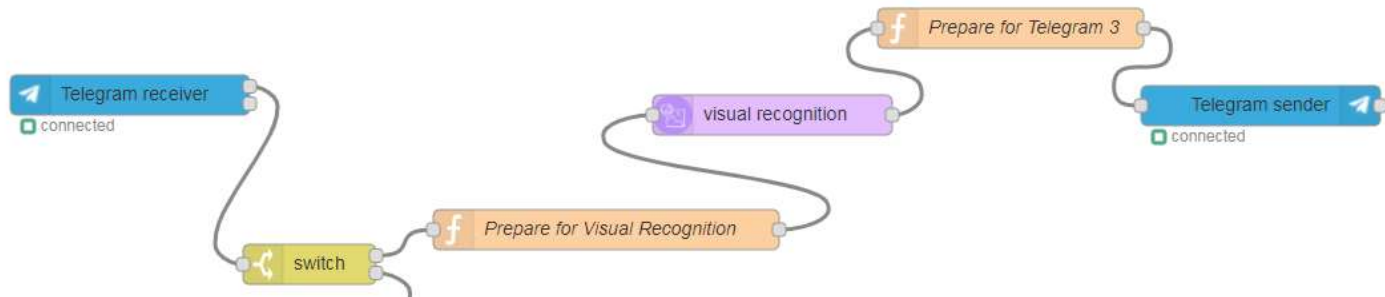
```
  type='message',
```

```
  content: 'That's a ' + msg.result.images[0].classifiers[0].classes[0].class
```

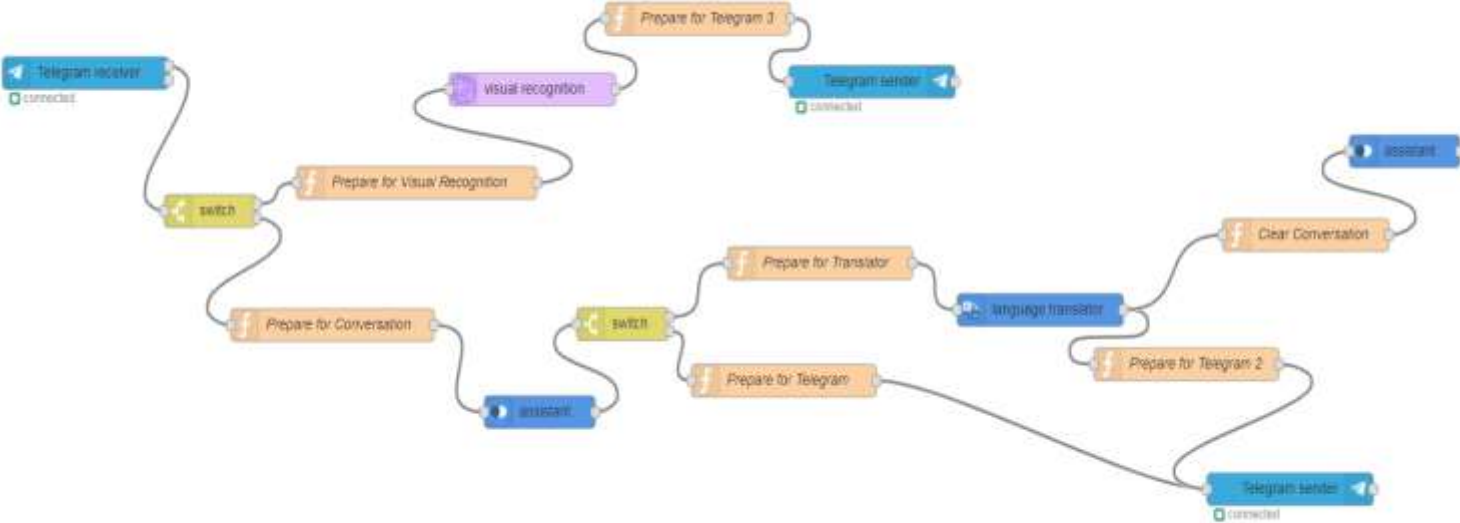
```
  }
```

```
  return msg;
```

- Infine ricollegghiamo il tutto al telegram sender



Flusso Completo



Thank You

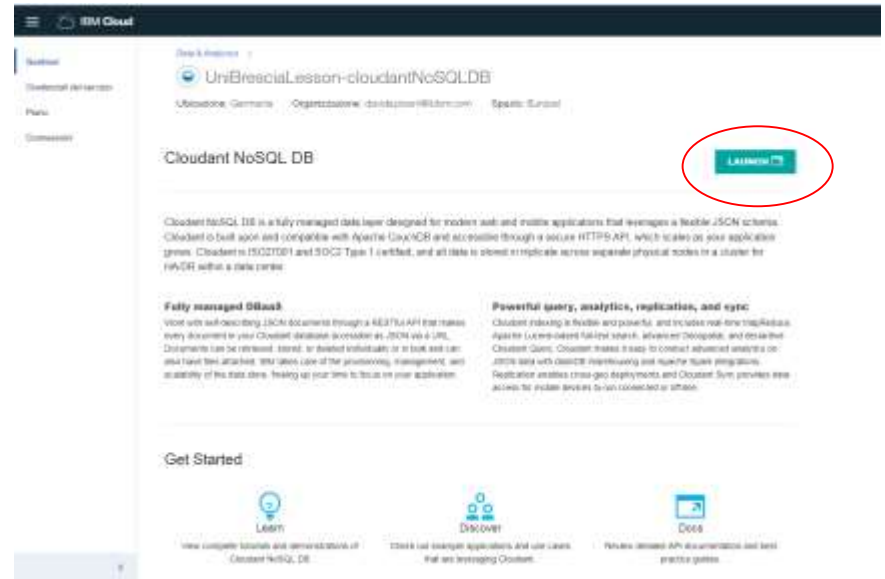
Per qualsiasi domanda o dubbio scrivetemi qui:

davide.pisoni@it.ibm.com



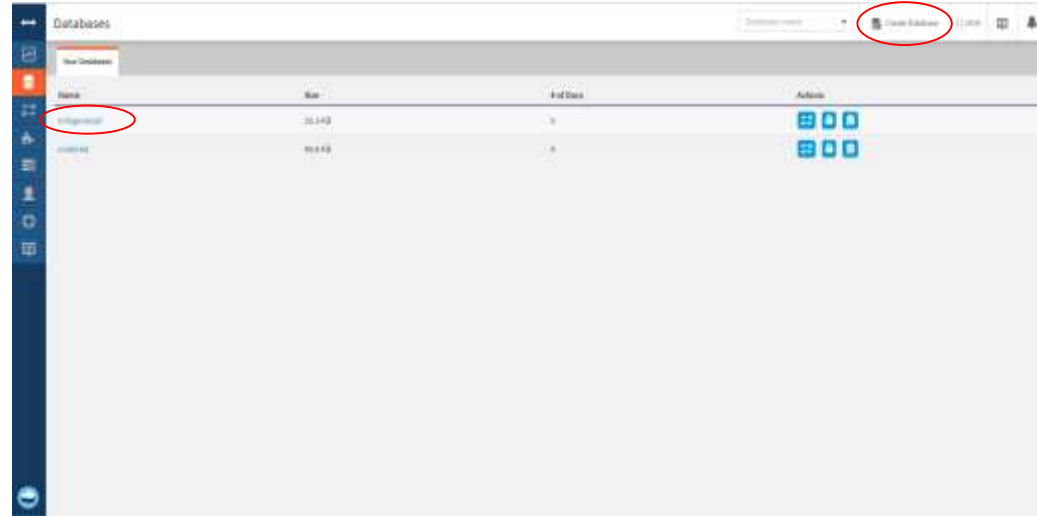
Altre integrazioni: Cloudant(1/3)

- Potrebbe essere utile al posto di impostare le risposte su Assistant, immagazzinarle in un database e recuperarle da lì
- Il container di Node-Red su IBM Cloud viene sempre creato insieme a Cloudant, un database No Sql che immagazina le informazioni su documenti in json
- In questa ipotesi conversation verrà utilizzato sole come *classificatore*, cioè identificherà intenti ed entità che ci serviranno per recuperare la risposta corretta su Cloudant
- Creiamo le risposte in Cloudant innanzitutto: per accedere al servizio basta cliccare sopra al launch nella sua schermata



Altre integrazioni: Cloudant(2/3)

- Creiamo un nuovo database con il nome dell'intento del quale creeremo le risposte
- Cliccandoci sopra vi possiamo creare all'interno i documenti con le risposte
- L'id delle risposte sarà l'entità che serve per catturarle e all'interno creeremo un campo risposta contenente la risposta



Altre integrazioni: Cloudant(3/3)

- Nel flusso Node-Red dovremo prevedere un nodo di switch che in base all'intento faccia la query nel giusto database e poi un nodo cloudant che ci restituirà il documento corretto
- Tra il nodo di switch e quello di cloudant inseriremo la seguente funzione:

```
msg.payload=msg.payload.entities[0].entity;  
return msg;
```

- In questo modo otterremo il seguente flusso

