

UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA



*Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione*

***Implementazione di un chatbot tramite framework Rasa
per curiosità e statistiche sul mondo del calcio***

Studenti:

DANIELE PALLINI 1107326

MATTEO ABBRUZZETTI 1108842

Docenti:

DOMENICO URSINO

GIANLUCA BONIFAZI

MICHELE MARCHETTI

ANNO ACCADEMICO 2022-2023

Indice

1	Introduzione	3
1.1	Rasa	3
1.2	Obiettivi chatbot	5
2	Implementazione	6
2.1	Avvio della conversazione con il chatbot	6
2.2	Ricerca dei giocatori	7
2.2.1	Informazioni personali	9
2.2.2	Statistiche stagionali	10
2.2.3	Cronologia trasferimenti	11
2.3	Ricerca delle squadre	12
3	Connessione a Telegram e testing	15
3.1	Connessione a Telegram	15
3.2	Testing	15
4	Conclusioni	20
	Elenco delle figure	21

Capitolo 1

Introduzione

In questo elaborato si presenterà la progettazione, l'implementazione e il testing di un chatbot, un software che rende possibile l'interazione tra un essere umano e un computer. I chatbot devono essere in grado di sostenere una conversazione con un essere umano ricevendo come input una voce, un testo o altre forme di comunicazione. Un buon chatbot si ottiene nel momento in cui l'essere umano non è in grado di capire se dall'altra parte ci sia un'altra persona o un computer. Nel nostro caso, il chatbot è stato realizzato in modo da fornire informazioni e statistiche sul mondo del calcio. In particolare, il bot dovrà essere in grado di comprendere e rispondere correttamente ai messaggi di testo inviati dall'utente, il quale potrà richiedere dati e statistiche riguardanti calciatori e squadre. Le informazioni per costruire le risposte saranno recuperate da *Transfermarkt*, il sito più conosciuto e affidabile per quanto riguarda il mondo del calcio e tutto ciò a esso collegato.

1.1 Rasa

Rasa è un framework open source in grado di automatizzare conversazioni testuali o vocali. Utilizza il machine learning e il Natural Language Processing (NLP) per interpretare correttamente il messaggio dell'utente e fornire ad esso una risposta adeguata.



Figura 1.1: Logo di Rasa

L'architettura di *Rasa* è divisa in due componenti:

- **Rasa NLU:** è la prima componente che viene attivata quando si riceve un messaggio. È il motore di comprensione e si occupa di interpretare e classificare il messaggio dell'utente.
- **Rasa Core:** è il cuore del chatbot; questa componente, infatti, ha il compito di scegliere la risposta più adatta alla domanda posta dall'utente.

Il framework mette a disposizione il comando *rasa init* per creare la struttura di base del progetto. In Figura 1.2 viene riportata l'organizzazione delle cartelle e dei file nel caso presentato in questo elaborato.

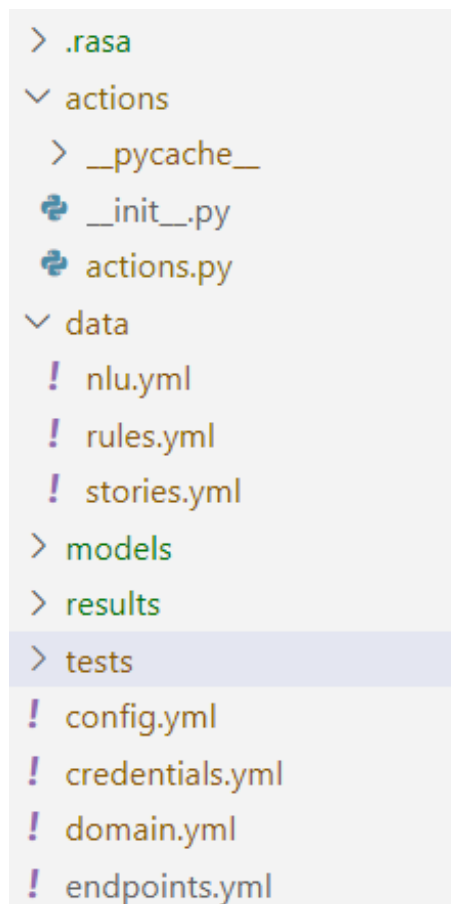


Figura 1.2: Struttura del progetto relativo al chatbot

Nel file **domain.yml** vengono definiti tutti gli elementi del chatbot che gli permettono di comprendere il linguaggio umano, di rispondere in maniera sensata e di eseguire azioni. Come si può evincere dal nome, questo file rappresenta il dominio del progetto; qui avviene la definizione dei seguenti elementi:

- *Intents*: permettono al bot di capire il contesto; estraggono l'intenzione insita nella domanda posta dall'utente.
- *Entities*: sono i soggetti principali degli intents. Grazie alle entities, il bot è in grado di comprendere molto più facilmente il contesto della domanda.
- *Actions*: sono le azioni che il bot può compiere nel momento in cui riconosce un intent.
- *Slots*: rappresentano la memoria del bot. Offrono al chatbot la capacità di memorizzare le informazioni presenti in un intent e di tenere traccia degli elementi chiave della conversazione.
- *Responses*: sono delle risposte statiche che il chatbot può utilizzare per alcuni intents.

Il chatbot deve essere addestrato a comprendere le richieste dell'utente e per farlo si forniscono ad esso degli esempi.

Nel file **nlu.yml**, per ogni intent, vengono inserite delle frasi tipiche, in modo tale che il chatbot possa riconoscere l'intent corrispondente alla frase inserita dall'utente. Questo file costituisce una parte fondamentale dell'addestramento del chatbot.

Il file **stories.yml** contiene le storie, ossia delle sequenze di dialogo che suggeriscono al programma le risposte più adatte in alcune conversazioni con l'utente.

Il file **actions.py** contiene le azioni che il chatbot dovrà eseguire per soddisfare le richieste dell'utente. Queste azioni sono chiamate *custom actions*, vengono eseguite in base al codice definito dal programmatore del chatbot e possono andare a costituire la risposta data dal sistema.

Nel file **rules.yml** vengono definite delle regole che il chatbot dovrà rispettare, come eseguire una particolare action nel caso in cui esso riconosca un intent nel messaggio ricevuto.

1.2 Obiettivi chatbot

Il chatbot che abbiamo deciso di sviluppare ha la funzione di effettuare ricerche e mostrare statistiche riguardanti calciatori e società calcistiche. L'utente può chiedere informazioni relative agli aspetti di campo, come gol e assist, ma anche riguardanti l'aspetto economico, come la valutazione di un giocatore o il prezzo del cartellino di un trasferimento. I dati sono recuperati dal sito più affidabile riguardo questo genere di informazioni, *Transfermarkt*, il quale li raccoglie dal 2004. Le diverse richieste che possono essere poste dall'utente verranno analizzate più in dettaglio nel capitolo successivo. Il chatbot, inoltre, sarà sviluppato in modo da essere il più chiaro e *user-friendly* possibile.

Capitolo 2

Implementazione

In questo capitolo verranno illustrate in dettaglio le scelte fatte in fase di implementazione del chatbot. Saranno analizzate e commentate tutte le funzionalità messe a disposizione dell'utente finale. Un fattore tenuto molto in considerazione lungo tutto il progetto è rappresentato dalla facilità di utilizzo del prodotto finale; l'interazione con il chatbot dovrà essere, infatti, il più possibile *user-friendly*. In questa direzione va anche la scelta di rendere il bot amichevole; infatti, esso utilizzerà frequentemente emoticon nelle risposte.

2.1 Avvio della conversazione con il chatbot

La prima operazione da eseguire è l'avvio della conversazione con il chatbot, attraverso il comando `/start`. A questo proposito, è stata definita una *rule* (Figura 2.1) che riconosce l'inizio di una nuova conversazione e risponde con un messaggio preimpostato di benvenuto. È in questa occasione che il bot si presenta e comunica il nome a lui assegnato, *Mark*. Il nome è stato scelto principalmente per l'assonanza con il sito da cui il bot recupera le informazioni, *Transfermarkt*. Mark, inoltre, presenta all'utente le richieste che è in grado di soddisfare. Nel caso in cui l'utente sia in difficoltà su come procedere, è stato predisposto un intent apposito, che riconosce la richiesta di chiarimenti (Figura 2.2). Un esempio di inizio di una conversazione, in cui l'utente effettua anche una richiesta di aiuto, è rappresentato in Figura 2.3.

```
- rule: start the conversation
  steps:
  - intent: start
  - action: utter_start
```

Figura 2.1: Rule per l'inizio della conversazione

```
- intent: help
  examples: |
    - Cosa ti posso chiedere?
    - Quali sono le tue funzioni?
    - Cosa fai?
    - Cosa dovrei chiederti?
    - Come mi puoi aiutare
    - Ho bisogno del tuo aiuto
    - Aiuto
```

Figura 2.2: Intent per le richieste di aiuto

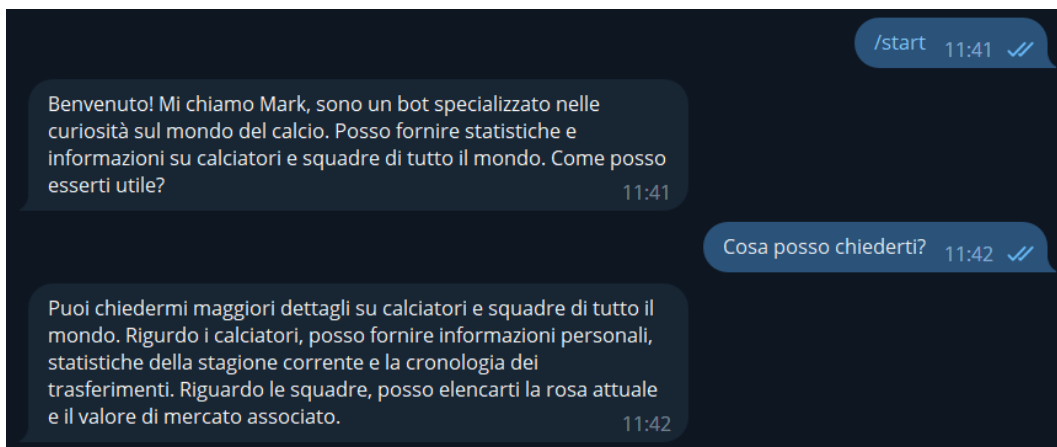


Figura 2.3: Inizio di una conversazione

2.2 Ricerca dei giocatori

La prima vera e propria funzionalità implementata è stata la ricerca di giocatori. Sono state definite delle domande standard che l'utente potrebbe porre per cercare un giocatore (Figura 2.5). Tutte prevedono che venga specificato un nome, il quale viene riconosciuto dal sistema come entity *player_name* e salvato nello slot di memoria associato (Figura 2.4). Per aiutare il bot a capire che l'utente sta facendo riferimento a un calciatore, è stata definita una tabella di *lookup* contenente circa 25.000 nomi di giocatori del mondo del calcio degli ultimi 20 anni (un estratto della lista è rappresentato in Figura 2.6).

```
slots:
  player_name:
    type: text
    mappings:
      - type: from_entity
        entity: player_name
```

Figura 2.4: Slot di memoria per i giocatori e entity associate

```
- intent: find_info
examples: |
  - puoi darmi informazioni riguardo [Donnarumma](player_name)?
  - ho bisogno che tu mi dia informazioni su [Kessie](player_name)
  - cosa sai di [Tatarusanu](player_name)?
  - vorrei informazioni riguardo [Giroud](player_name)
  - sai qualcosa di [Thiaw](player_name)?
  - chi è [Kalulu](player_name)?
  - dove gioca [Calabria](player_name)?
  - [Maignan](player_name) è un calciatore?
  - voglio informazioni su [Tonali](player_name)
  - dammi informazioni riguardo [Vlahovic](player_name)
  - info su [Bennacer](player_name)
```

Figura 2.5: Esempi per l'Intent di ricerca giocatori

```
- lookup: player_name
examples: |
  - Javier Pastore
  - Chiquinho
  - Theodoros Vasilakakis
  - Mario Pasalic
  - Abdelhamid Sabiri
  - Filip Djuricic
  - Riccardo Saponara
```

Figura 2.6: Estratto della tabella di lookup di nomi di giocatori

Il bot va quindi a cercare su *Transfermarkt* il giocatore richiesto e restituisce il nome completo, la squadra di appartenenza e una foto (Figura 2.7). A quel punto, viene chiesto all'utente se desidera ricevere maggiori informazioni, proponendogli 3 diverse opzioni:

- Informazioni personali
- Statistiche calciatore
- Cronologia trasferimenti

Per semplificare il più possibile il processo, le 3 alternative sono state associate a dei pulsanti da cliccare. Nel momento in cui l'utente preme uno dei pulsanti, viene selezionata l'opzione relativa e il bot risponde con le informazioni richieste. La definizione dei pulsanti è visualizzabile in Figura 2.8.

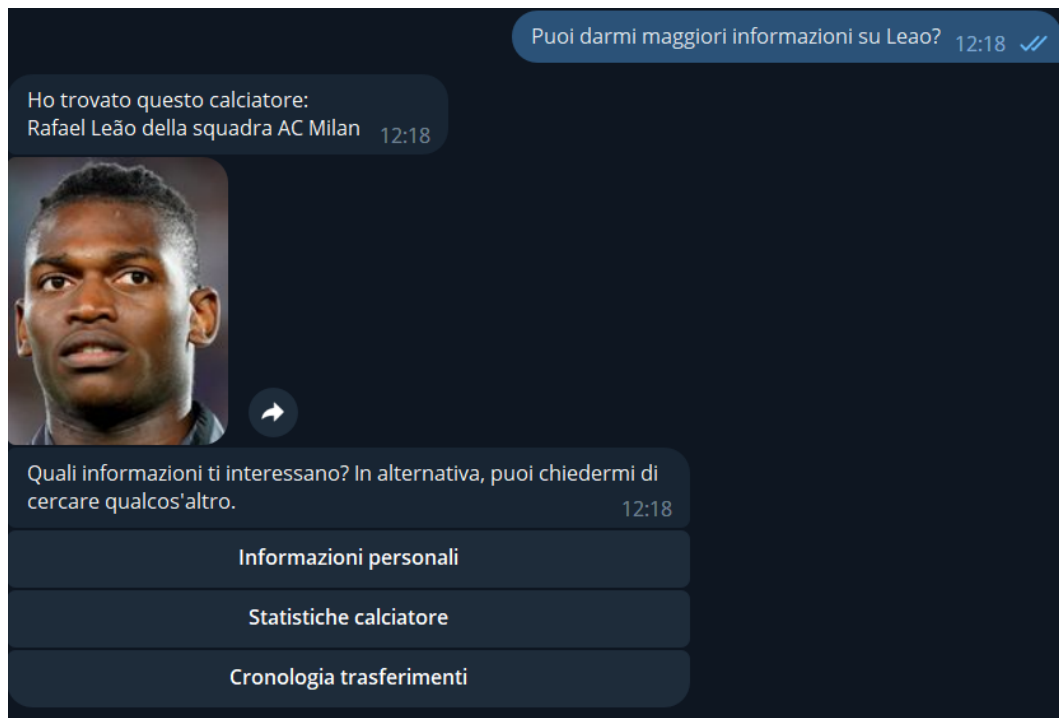


Figura 2.7: Ricerca del calciatore Leao e relativa risposta

```
utter_choose_info:
- buttons:
  - title: Informazioni personali
    payload: /info-player>{"info":"info"}
  - title: Statistiche calciatore
    payload: /info-player>{"info":"stats"}
  - title: Cronologia trasferimenti
    payload: /info-player>{"info":"transfer"}
text: Quali informazioni ti interessano? In alternativa, puoi chiedermi di cercare qualcos'altro.
button_type: vertical
```

Figura 2.8: Definizione dei tre pulsanti

2.2.1 Informazioni personali

Cliccando il primo pulsante, il chatbot fornisce maggiori informazioni riguardanti i dati anagrafici e alcuni parametri sulla valutazione di mercato del calciatore richiesto¹(Figura 2.9).

¹Dopo i messaggi di risposta alla richiesta vengono riproposti i tre pulsanti di scelta, in modo da permettere all'utente di ottenere informazioni di natura diversa in un'unica ricerca.



Figura 2.9: Risposta riguardo le informazioni personali del calciatore Leao

2.2.2 Statistiche stagionali

Cliccando il secondo pulsante, il chatbot elenca le statistiche stagionali del giocatore, competizione per competizione. Si avranno, quindi, tanti messaggi quanto è il numero di competizioni affrontate (Figura 2.10). Per ogni torneo vengono specificati: nome della competizione, numero di partite giocate, numero di goal fatti, numero di assist, numero di cartellini gialli, numero di cartellini rossi, percentuale di partite da titolare, percentuale di minuti in campo².

²Dopo i messaggi di risposta alla richiesta vengono riproposti i tre pulsanti di scelta, in modo da permettere all'utente di ottenere informazioni di natura diversa in un'unica ricerca.

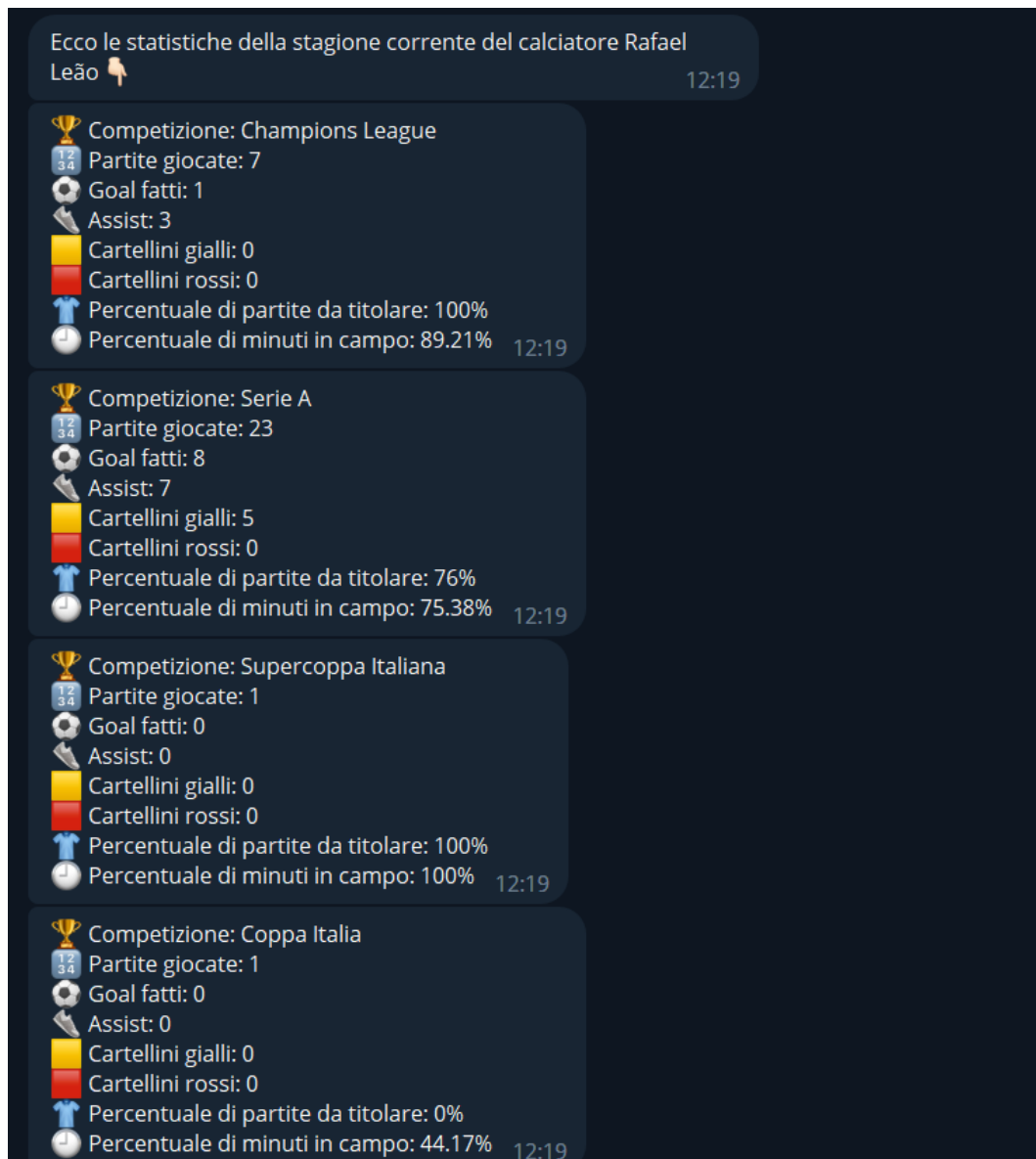


Figura 2.10: Risposta riguardo le statistiche stagionali del calciatore Leao

2.2.3 Cronologia trasferimenti

Cliccando il terzo pulsante, il chatbot elenca la cronologia dei trasferimenti del giocatore richiesto. Ogni riga è formata dalla stagione calcistica di riferimento del trasferimento, la squadra di partenza, la squadra di arrivo e il costo dell'operazione³(Figura 2.11).

³Dopo i messaggi di risposta alla richiesta vengono riproposti i tre pulsanti di scelta, in modo da permettere all'utente di ottenere informazioni di natura diversa in un'unica ricerca.

Ecco la cronologia trasferimenti del calciatore Rafael Leão

12:19

Stagione	Trasferimento	Costo
19/20	Lille → Milan	29,50 mln €
18/19	Sporting CP → Lille	FREE
17/18	Sporting B → Sporting CP	FREE
17/18	Sporting U19 → Sporting B	FREE
16/17	Sporting U17 → Sporting U19	FREE
14/15	Sporting Sub-15 → Sporting U17	FREE
12/13	Sporting Giov. → Sporting Sub-15	FREE

12:19

Figura 2.11: Risposta riguardo la cronologia trasferimenti del calciatore Leao

2.3 Ricerca delle squadre

La seconda funzionalità implementata è stata la ricerca delle società calcistiche. Sono state definite delle domande standard che l'utente potrebbe porre per cercare un club (Figura 2.12). Tutte prevedono che venga specificato il nome di una società, il quale viene riconosciuto dal sistema come entity *team_name* e salvato nello slot di memoria associato (Figura 2.13). Per aiutare il bot a capire che l'utente sta facendo riferimento a una società, è stata definita una tabella di *lookup* contenente circa 400 club europei che hanno partecipato a leghe delle massime divisioni negli ultimi 20 anni (un estratto della lista è rappresentato in Figura 2.14).

```
- intent: find_team
examples: |
- vorrei la rosa della [Lazio](team_name)?
- dimmi la rosa della [Juventus](team_name)
- cerca la squadra [Real Madrid](team_name)
- parlami del [Barcelona](team_name)
- voglio sapere la rosa del [Chelsea](team_name)
- chi gioca al [Napoli](team_name)?
- quanto vale la [Roma](team_name)?
```

Figura 2.12: Esempi per l'Intent di ricerca club

```
slots:  
  team_name:  
    type: text  
    mappings:  
      - type: from_entity  
        entity: team_name
```

Figura 2.13: Slot di memoria per i club e entity associate


```
- lookup: team_name  
examples: |  
  - Spezia  
  - Lecce  
  - Sampdoria  
  - Empoli  
  - Salernitana  
  - Cremonese  
  - Hellas Verona
```

Figura 2.14: Estratto della tabella di lookup di club

Il bot va quindi a cercare su *Transfermarkt* il club richiesto e restituisce il nome completo e il logo. Successivamente, il chatbot elenca i giocatori tesserati con quella società, dividendoli per ruolo (portieri, difensori, centrocampisti, attaccanti). Infine, viene riportato il valore complessivo dei giocatori sopracitati. Un esempio di conversazione e relativa risposta è rappresentato in Figura 2.15.

Vorrei sapere la rosa del Milan 12:23 ✓

Ecco la rosa della squadra AC Milan 12:24



Nome	Ruolo	Età	Nazionale	Valore
PORTIERI				
Mike Maignan	Por	(27)		35,00 mln €
Ciprian Tatarusanu	Por	(37)		600 mila €
Devis Vázquez	Por	(24)		350 mila €
Antonio Mirante	Por	(39)		300 mila €
DIFENSORI				
Fikayo Tomori	DC	(25)		50,00 mln €
Pierre Kalulu	DC	(22)		35,00 mln €
Matteo Gabbia	DC	(23)		7,00 mln €
Malick Thiaw	DC	(21)		6,00 mln €
Simon Kjaer	DC	(33)		5,00 mln €
Theo Hernández	TS	(25)		60,00 mln €
Fodé Ballo-Touré	TS	(26)		3,50 mln €
Davide Calabria	TD	(26)		25,00 mln €
Sergio Dest	TD	(22)		16,00 mln €
Alessandro Florenzi	TD	(31)		2,50 mln €

(a)

CENTROCAMPISTI				
Sandro Tonali	Med	(22)		50,00 mln €
Ismaël Bennacer	Med	(25)		40,00 mln €
Tiemoué Bakayoko	Med	(28)		4,00 mln €
Tommaso Pobega	CC	(23)		18,00 mln €
Aster Vranckx	CC	(20)		10,00 mln €
Rade Krunić	CC	(29)		9,00 mln €
Charles De Ketelaere	Trq	(21)		30,00 mln €
Brahim Díaz	Trq	(23)		20,00 mln €
Yacine Adli	Trq	(22)		9,00 mln €
ATTACCANTI				
Rafael Leão	AS	(23)		85,00 mln €
Ante Rebić	AS	(29)		13,00 mln €
Alexis Saelemaekers	AD	(23)		15,00 mln €
Junior Messias	AD	(31)		6,00 mln €
Divock Origi	PC	(27)		10,00 mln €
Olivier Giroud	PC	(36)		4,00 mln €
Zlatan Ibrahimović	PC	(41)		2,00 mln €
Emil Roback	PC	(19)		150 mila €
Il valore totale della squadra AC Milan è 571,40 mln €				

(b)

Figura 2.15: Ricerca della rosa del Milan e relativa risposta

Capitolo 3

Connessione a Telegram e testing

3.1 Connessione a Telegram

Si è scelto di collegare il chatbot alla piattaforma di instant messaging *Telegram*, in modo da offrire agli utenti una facile connessione e interazione con il bot stesso, direttamente attraverso il loro smartphone. Per effettuare il collegamento a *Telegram*, innanzitutto bisogna richiedere la creazione di un nuovo bot a *BotFather*, un chatbot apposito a cui vanno specificati nome e username del bot che si vuole andare a creare. Se la procedura di creazione va a buon fine, *BotFather* rilascia un token di accesso al bot appena creato. Il token e lo username vanno inseriti nel file **credentials.yml** (Figura 3.1). Il chatbot viene, quindi, eseguito su un server locale dal servizio *ngrok* e connesso a *Telegram* tramite le credenziali specificate.

```
telegram:  
  access_token: 5897509676:AAGs42x7nI850oHijKxNEvXYmIewMgSlwI  
  verify: MarktRasaBot  
  webhook_url: "https://c85c-87-26-84-145.eu.ngrok.io/webhooks/telegram/webhook"
```

Figura 3.1: Configurazione del bot relativa a Telegram

3.2 Testing

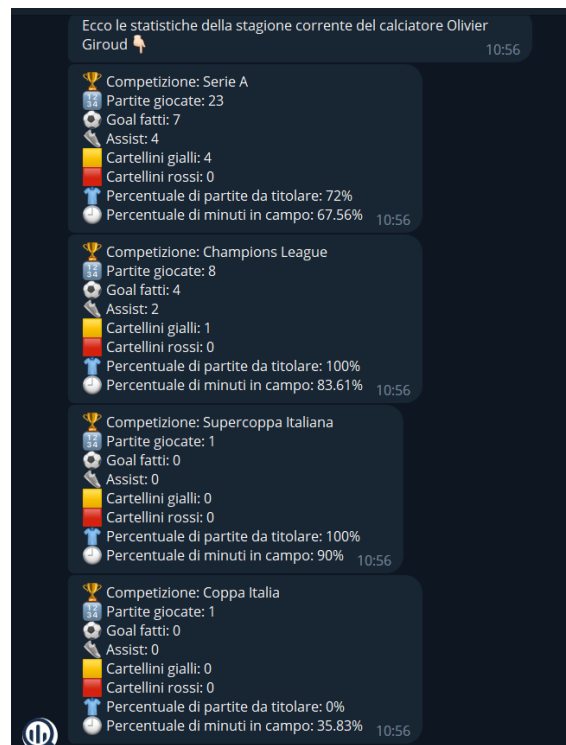
Di seguito si riportano degli esempi di conversazione con il chatbot in cui l'utente effettua diverse richieste relative a tutte le funzionalità implementate (Figure 3.2, 3.3, 3.4).



(a)



(b)

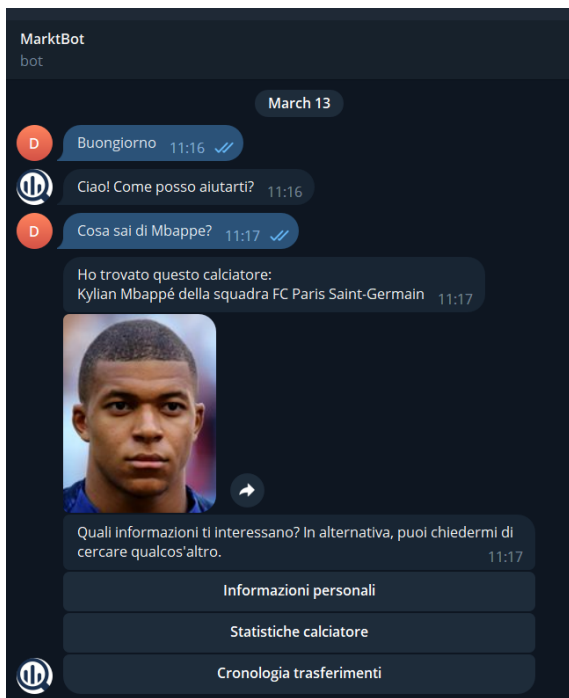


(c)

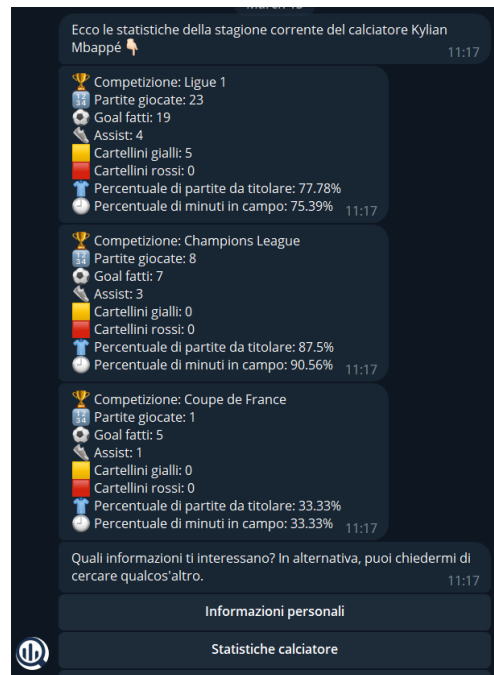


(d)

Figura 3.2: Esempio di conversazione con il chatbot



(a)



(b)

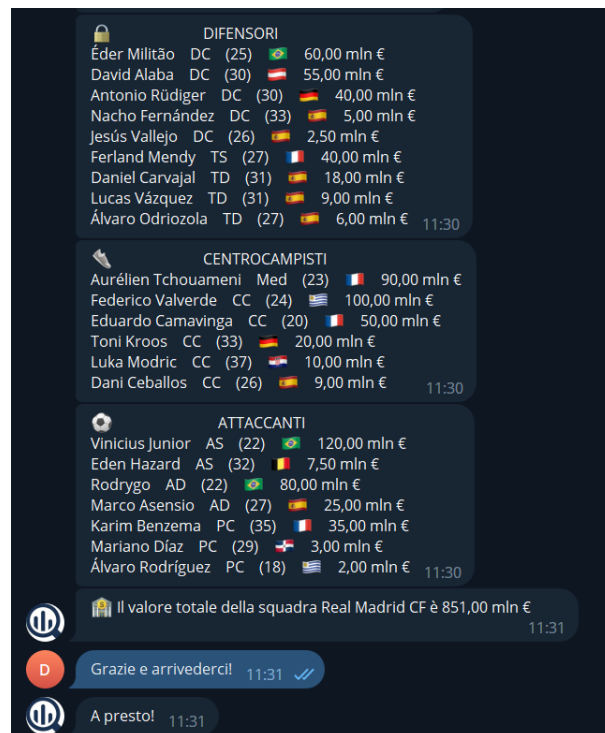


(c)

Figura 3.3: Un altro esempio di conversazione con il chatbot



(a)



(b)

Figura 3.4: Un ulteriore esempio di conversazione con il chatbot

Capitolo 4

Conclusioni

Si è utilizzato il framework Rasa per l'implementazione di un chatbot in grado di effettuare ricerche e mostrare statistiche riguardanti calciatori e società calcistiche. Si è partiti addestrando il chatbot a riconoscere il contesto delle domande, attraverso la definizione di richieste tipiche che l'utente potrebbe porre, e i nomi di calciatori e squadre, mediante tabelle di riferimento (*lookup tables*). Si è proseguito, poi, con lo sviluppo delle funzionalità di ricerca e risposta del chatbot. Sono stati, quindi, valutati i risultati ottenuti, effettuato il collegamento del bot a *Telegram* e svolti test basati su conversazioni realistiche.

In generale, Rasa si è dimostrato un ottimo strumento di sviluppo di chatbot. La facilità di utilizzo, unita alla possibilità di implementare funzionalità di ogni tipo, rende il mondo dei chatbot molto interessante e aperto ai più svariati sviluppi futuri.

Elenco delle figure

1.1	Logo di Rasa	3
1.2	Struttura del progetto relativo al chatbot	4
2.1	Rule per l'inizio della conversazione	6
2.2	Intent per le richieste di aiuto	7
2.3	Inizio di una conversazione	7
2.4	Slot di memoria per i giocatori e entity associate	7
2.5	Esempi per l'Intent di ricerca giocatori	8
2.6	Estratto della tabella di lookup di nomi di giocatori	8
2.7	Ricerca del calciatore Leao e relativa risposta	9
2.8	Definizione dei tre pulsanti	9
2.9	Risposta riguardo le informazioni personali del calciatore Leao	10
2.10	Risposta riguardo le statistiche stagionali del calciatore Leao	11
2.11	Risposta riguardo la cronologia trasferimenti del calciatore Leao . .	12
2.12	Esempi per l'Intent di ricerca club	12
2.13	Slot di memoria per i club e entity associate	13
2.14	Estratto della tabella di lookup di club	13
2.15	Ricerca della rosa del Milan e relativa risposta	14
3.1	Configurazione del bot relativa a Telegram	15
3.2	Esempio di conversazione con il chatbot	17
3.3	Un altro esempio di conversazione con il chatbot	18
3.4	Un ulteriore esempio di conversazione con il chatbot	19