Bot Telegram per la consultazione delle lezioni UniPG

UNIVERSITÀ DEGLI STUDI DI PERUGIA

A. A. 2019-2020

Diego Arcelli
Matricola 311045

Progetto di Ingegneria del Software
Docente Alfredo Milani

Sommario:

1. Obiettivo del progetto
2. Analisi dei requisiti
   - 2.1. Glossario
   - 2.2. Descrizione informale
   - 2.3. Individuazione dei requisiti
   - 2.4. Aspetti tecnici
3. Architettura del sistema
   - 3.1. Descrizione
   - 3.2. Diagramma delle classi
   - 3.3. Diagramma dei casi d’uso
   - 3.4. Diagramma di sequenza
   - 3.5. Diagramma di stato
   - 3.6. Diagramma di attività
4. Codice sorgente
5. Casi di test
6. Design pattern
7. Sitografia e fonti

1. Obiettivo del progetto
Realizzazione di un bot Telegram che permette, tramite l'interrogazione di un database, la ricerca di informazioni relative alle lezioni degli insegnamenti erogati dalle varie facoltà dell'Università degli Studi di Perugia.

2. Analisi dei requisiti

2.1 Glossario
Definizioni relative all'organizzazione dell'università (non sono le definizioni istituzionali ufficiali, ma servono ad evitare ambiguità e fraintendimenti nella lettura del documento):
- **Corso di laurea**: corso che prevede una serie di insegnamenti relativi ad una particolare materia e permette il conseguimento di una laurea.
- **Insegnamento**: insieme di lezioni che prevedono la trattazione di un certo argomento. È associato ad un particolare corso di laurea, ed ha un docente titolare.
- **Lezione**: momento nel quale docente e studenti si riuniscono per trattare gli argomenti relativi all'insegnamento. Viene svolta in un'aula, con data e orario prestabiliti.
- **Docente**: colui che può essere titolare di un particolare insegnamento e svolgere lezioni (non deve essere necessariamente il docente titolare dell'insegnamento a svolgere le lezioni).

Definizioni relative agli aspetti tecnici:
- **Bot**: programma che ha accesso alle chat di Telegram, come se fosse un normale utente, il quale può eseguire, in automatico o in risposta all'utente, determinate azioni.
- **Comando**: messaggio speciale che viene mandato nella chat dove è presente il bot, a seguito del quale il bot esegue una particolare funzione. Per quanto riguarda Telegram ogni comando deve avere come prefisso il carattere '/'.
- **Parametro**: informazione aggiuntiva che viene inserta insieme ad un comando per fornire informazioni aggiuntive al bot su come eseguire il comando.

2.2 Descrizione informale
Dal punto di vista dell'utente il bot deve permettere di ricercare informazioni riguardanti le lezioni erogate dai vari corsi di laurea. Le informazioni che possono essere di interesse per l'utente sono l'elenco delle lezioni previste per un determinato insegnamento e le informazioni ad esse annesse, come la data e l'orario di svolgimento. Può anche essere utile poter ricevere delle informazioni generali sui singoli insegnamenti, come il docente titolare ed il corso di laurea dell'insegnamento. Inoltre vanno previste diverse modalità di ricerca, in particolare una modalità che permette di ricevere informazioni sull'insegnamento in maniera immediata, ed una che permette di navigare tra gli insegnamenti in maniera non confusionaria. Inoltre l'utente che utilizza il bot per la prima volta deve poter essere messo nelle condizioni di capirne il funzionamento.

2.3 Individuazione dei requisiti
Dalla descrizione informale si può evincere che le informazioni di interesse all'utente sono relative a due principali entità: gli insegnamenti e le lezioni. Le lezioni a loro volta devono essere comunque associate ad un insegnamento.
Per quanto riguarda le lezioni, le informazioni da restituire all'utente a seguito di una ricerca sono:
- Il docente che ha tenuto la lezione
- A quale insegnamento è relativa la lezione
- La data della lezione
- L'ora di inizio della lezione

- L’ora della fine della lezione
- L’aula nella quale si tiene la lezione

Per quanto riguarda gli insegnamenti le informazioni di interesse sono:

- Il nome dell’insegnamento
- Il docente titolare dell’insegnamento
- Il corso di laurea che eroga l’insegnamento
- Il link all’aula virtuale di Microsoft Teams dell’insegnamento

Altre due entità importanti che emergono sono i docenti ed i corsi di laurea. Per i docenti le informazioni di interesse all’utente sono il nome ed il cognome del docente, mentre per il corso di laurea interessa solo il nome del corso.
Per quanto riguarda le modalità di ricerca, ognuna di esse deve essere implementata da un differente comando. In particolare andranno realizzati comandi che permettono i seguenti tipi di ricerca:

- Una ricerca che permette di ricevere informazioni immediate di un particolare insegnamento specificato dall’utente. Va dunque previsto un modo per l’utente di specificare in maniera immediata e univoca un particolare insegnamento.
- Una ricerca che funziona in maniera analoga alla precedente, ma che restituisce l’elenco delle lezioni previste dall'insegnamento.
- Una ricerca che permette di navigare tra gli insegnamenti per poi restituire la informazioni dell’insegnamento che verrà selezionato dall’utente. Per non intasare eccessivamente la chat la navigazione può avvenire nella seguente maniera: prima si sceglie da un elenco dei corsi di laurea il corso al quale l’insegnamento appartiene, poi viene visualizzata la lista degli insegnamenti del corso di laurea, dalla quale si sceglie l’insegnamento desiderato.
- Una ricerca che funziona in maniera analoga alla precedente, ma che restituisce l’elenco delle lezioni previste dall'insegnamento.
- Una ricerca che permette all’utente di ricercare gli insegnamenti inserendo il loro nome. Visto che differenti corsi di laurea possono avere insegnamenti con lo stesso nome dovrà anche essere possibile filtrare la ricerca ad un solo corso di laurea (ad esempio l’esame di analisi matematica è comune a più corsi di laurea).
- Una ricerca che funziona in maniera analoga alla precedente ma filtra in base a nome e cognome del docente titolare dell’insegnamento.

Dovrà inoltre essere possibile per l’utente avere una breve guida che esplicita l’utilizzo del bot e le varie modalità di ricerca delle informazioni.

2.4 Aspetti tecnici

Dal punto di vista tecnico l’applicazione si suddivide in due moduli principali:

- Il primo è quello che riguarda la memorizzazione e l’organizzazione delle informazioni di interesse all’utente.
- Il secondo è quello riguardante la programmazione del comportamento del bot.

Per quanto riguarda il primo modulo, la memorizzazione delle informazioni avverrà tramite la gestione di un database, che conterrà le tabelle necessaire alla risoluzione del problema. Per la realizzazione di questo modulo si è scelto di utilizzare XAMPP, software che permette di realizzare in maniera semplice un web server con un database da poter interrogare. La scelta di XAMPP, impone MySQL come DBMS e Apache come web server. La scelta è ricaduta su XAMPP per esperienza pregressa dello sviluppatore con tale piattaforma.
La programmazione del bot avverrà tramite linguaggio Python, per il quale Telegram ha rilasciato le API per i programmatori. La scelta è ricaduta su Python per le seguenti motivazioni:

- Esperienza pregressa del programmatore con tale linguaggio.
- Ampia documentazione offerta da Telegram per l’utilizzo delle API.
- Possibilità di interrogare database MySQL tramite il modulo Python chiamato mysql.

3. Architettura del sistema

3.1 Descrizione

Per la gestione del database sono state innanzitutto realizzate due tabelle principali.
La tabella Insegnamento che contiene i seguenti campi:

- L'id dell'insegnamento, di tipo intero. È un codice numerico che serve ad identificare in maniera univoca l'insegnamento all'interno del sistema.
- Il nome dell'insegnamento, di tipo stringa.
- L'id del docente titolare dell'insegnamento, di tipo intero.
- Il nome del corso di laurea dell'insegnamento, di tipo stringa.
- Il link all'aula virtuale di Microsoft Teams dell'insegnamento, di tipo stringa.

#	Nome	Tipo	Codifica caratteri
1	**nome_insegnamento**	varchar(255)	utf16le_bin
2	**id_insegnamento** 🔑	int(255)	
3	**id_docente**	int(255)	
4	**corso_di_laurea**	varchar(255)	utf16le_bin
5	**link_aula_virtuale**	varchar(255)	utf16le_bin

La tabella Lezione che contiene i seguenti campi:

- L'id della lezione, di tipo intero. È un codice numerico che permette di identificare in maniera univoca la lezione (non ha una utilità pratica nelle query ma è necessario come chiave primaria della la tabella).
- L'id del docente che tiene la lezione, di tipo intero.
- L'id dell'insegnamento associato alla lezione, di tipo intero.
- La data nella quale si svolge la lezione, di tipo data.
- L'orario di inizio della lezione, di tipo time.
- L'orario della conclusione della lezione, di tipo time.
- L'aula nella quale si svolge la lezione, di tipo stringa.

#	Nome	Tipo	Codifica caratteri
1	**id_docente**	int(255)	
2	**id_insegnamento**	int(255)	
3	**id_lezione** 🔑	int(255)	
4	**data_lezione**	date	
5	**ora_inizio**	time(6)	
6	**ora_fine**	time(6)	
7	**aula**	varchar(255)	utf16le_bin

Per quanto riguarda il corso di laurea, siccome il suo utilizzo è solo quello di indicare il nome del corso quando si vogliono visualizzare le informazioni di un insegnamento, è stato scelto di memorizzarlo come campo di tipo stringa nella tabella Insegnamento. Per il docente siccome compare sia nella tabella Lezione che in quella Insegnamento è stato deciso di realizzare una terza tabella Docente che contiene come campi:

- L'id del docente, di tipo intero. È un codice numerico che serve per indicare in maniera univoca il docente all'interno del sistema.
- Il nome del docente, di tipo stringa.
- Il cognome del docente, di tipo stringa.

#	Nome	Tipo	Codifica caratteri
1	**nome**	varchar(255)	utf16le_bin
2	**cognome**	varchar(255)	utf16le_bin
3	**id_docente** 🔑	int(255)	

Per quanto riguarda il modulo dell'applicazione relativo alla programmazione del bot, esso è stato suddiviso a sua volta in due parti:

- La parte di programmazione effettiva del comportamento del bot e di come esso si interfaccia con l'utente, tramite l'utilizzo delle API di Telegram.
- La parte di gestione della connessione e delle interrogazioni al database.

Per interrogare il database sarà realizzata una classe chiamata DataBase, che avrà come solo attributo un oggetto della classe MySQLConnection del modulo Python mysql, che permette di instaurare una connessione con un database, che potrà in seguito essere interrogato tramite linguaggio SQL. Ogni metodo della classe interrogherà il database con la query necessaria all'estrazione delle informazioni che andranno inviate nella chat all'utente, a seguito dell'inserimento di un comando (quindi saranno solo query di tipo SELECT).
La parte di gestione del bot invece farà largo uso dei metodi e delle classi dei moduli Python telegram e telegram.ext. Inoltre manterrà un'istanza della classe DataBase che verrà utilizzata per ottenere i risultati delle interrogazioni al database da inviare in chat.
Le modalità di ricerca individuate nell'analisi dei requisiti saranno implementate dei seguenti comandi:

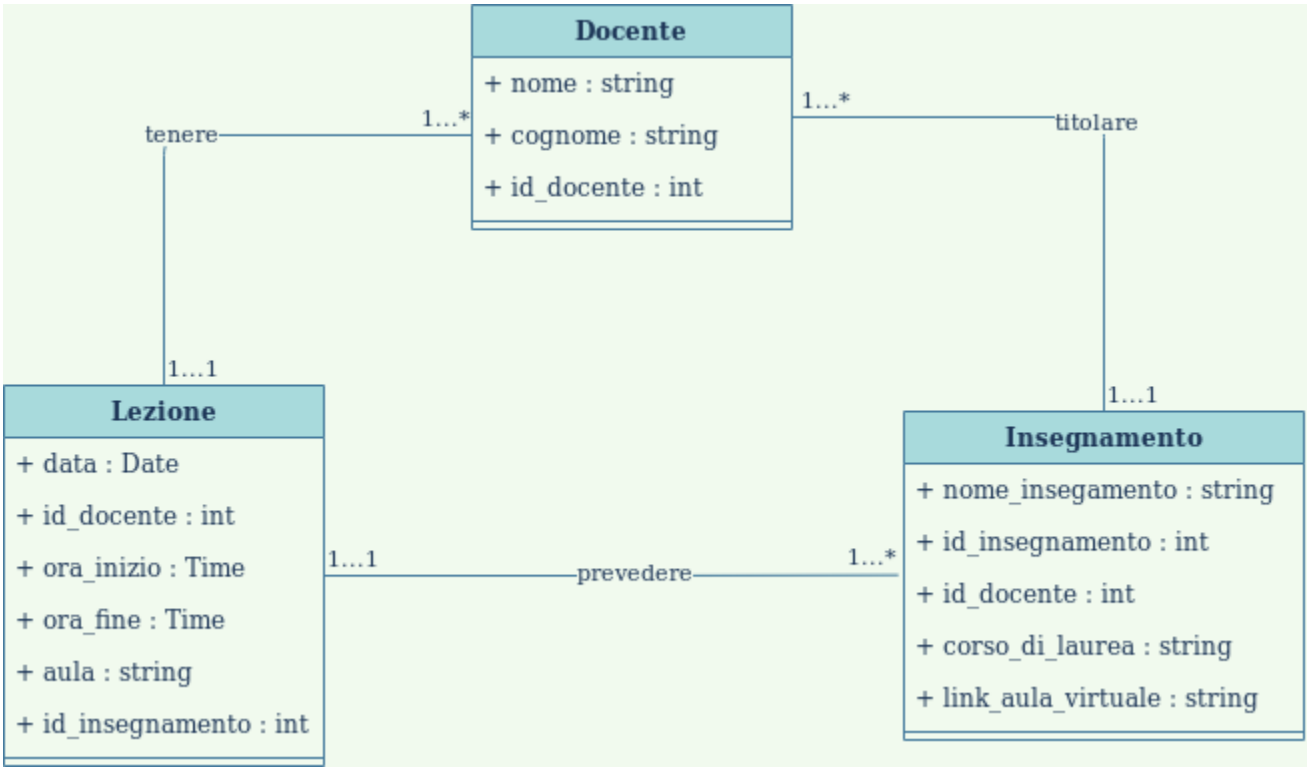
- **/list**: comando che non prende parametri. Ritorna l'elenco degli insegnamenti previsti per un determinato corso di laurea che viene selezionato tramite un menu grafico.
- **/info**: comando che ha lo scopo di fornire le informazioni di un particolare insegnamento. Ha due modalità di esecuzione. La prima si ha quando il comando viene invocato senza parametri: tramite menu grafico si seleziona il corso di laurea del quale si vogliono avere le informazioni, poi con un secondo menu grafico si seleziona l'insegnamento desiderato. La seconda modalità prevede il passaggio di un parametro al comando, che corrispondente all'id dell'insegnamento desiderato.
- **/lesson**: funziona come il comando /info, ma anziché fornire informazioni sull'insegnamento fornisce l'elenco delle lezioni.
- **/search**: comando che prevede il passaggio di una stringa come parametro. Prima di tutto richiede la selezione tramite menu grafico di un corso di laurea, poi ritorna gli insegnamenti il cui nome contiene come sotto-stringa il parametro passato.
- **/prof**: funziona come il comando /search, ma ritorna gli insegnamenti dove la stringa passata come parametro è contenuta nella concatenazione del nome e del cognome del docente.

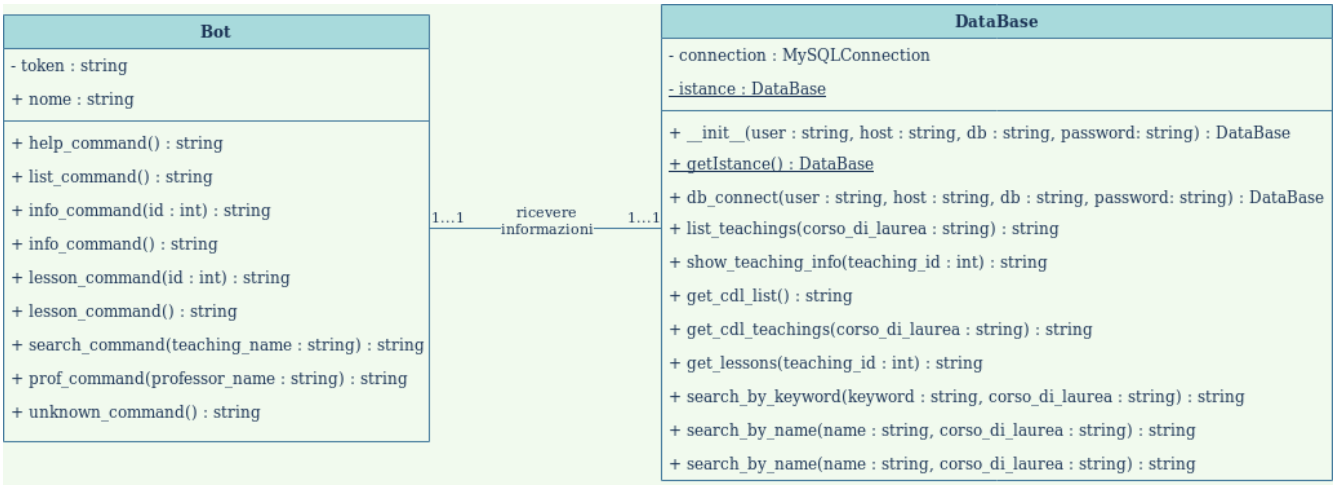
Sarà anche implementato il comando **/help** che serve ad inviare un messaggio in chat che spiega in maniera dettagliata tutti i comandi e come vanno utilizzati.

Il menu grafico citato nella descrizione dei comandi sarà implementato tramite la classe InlineKeyboardButton del modulo Python telegram, che permette di inviare in chat pulsanti che possono essere premuti dall'utente, che a seguito della pressione possono far eseguire al bot determinate azioni.

3.2 Diagramma delle classi

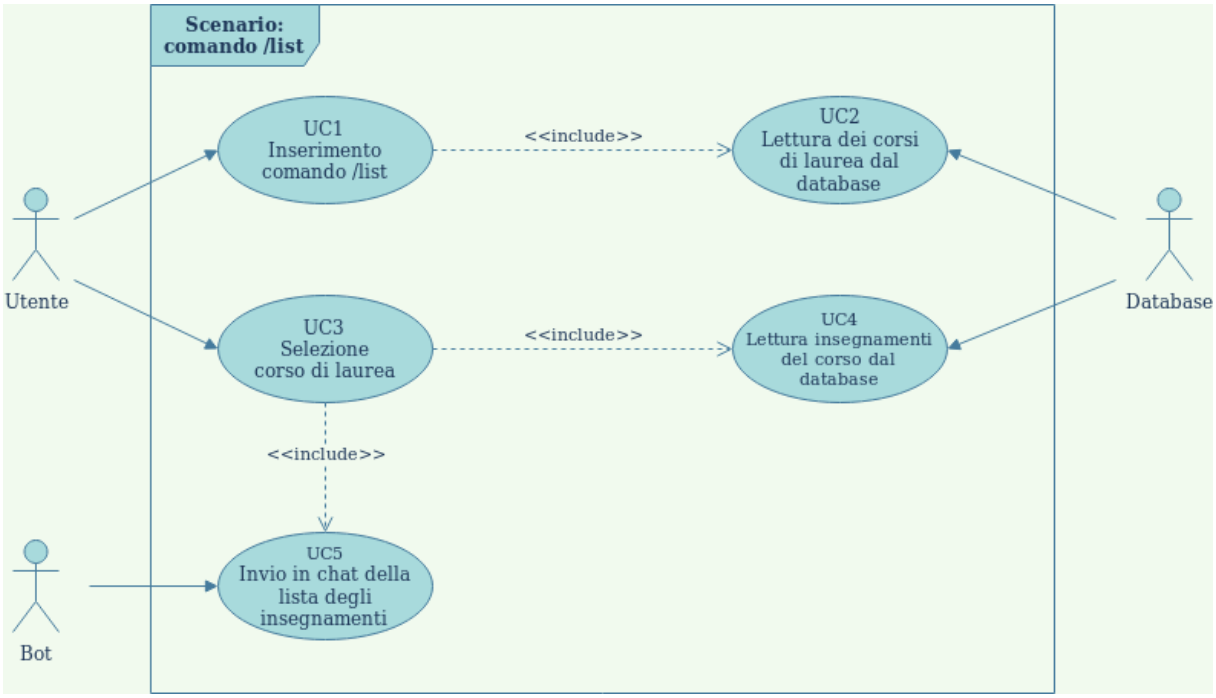
Sono stati realizzati due diagrammi delle classi. Il primo per descrivere le tabelle del database, il secondo per descrivere le classi coinvolte nel modulo relativo alla programmazione del bot.





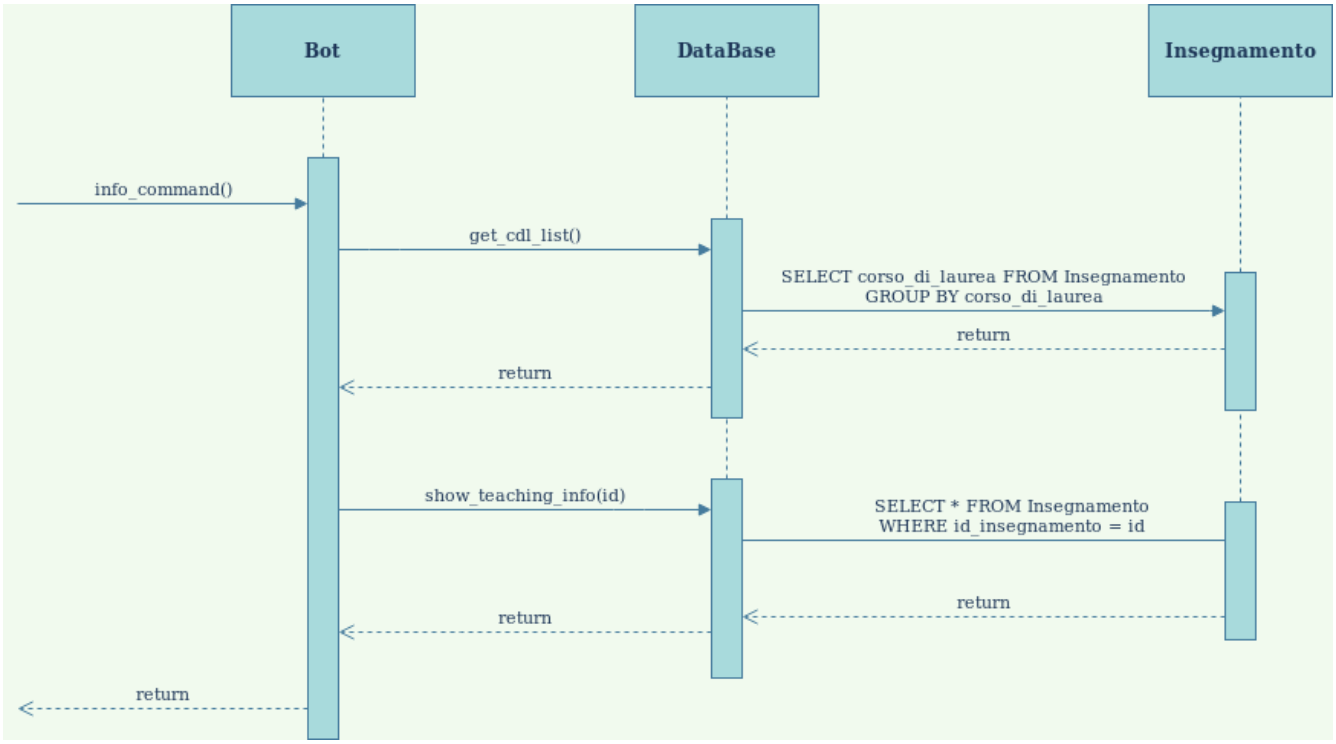
3.3 Diagramma dei casi d’uso

Diagramma relativo allo scenario: esecuzione del comando /list.



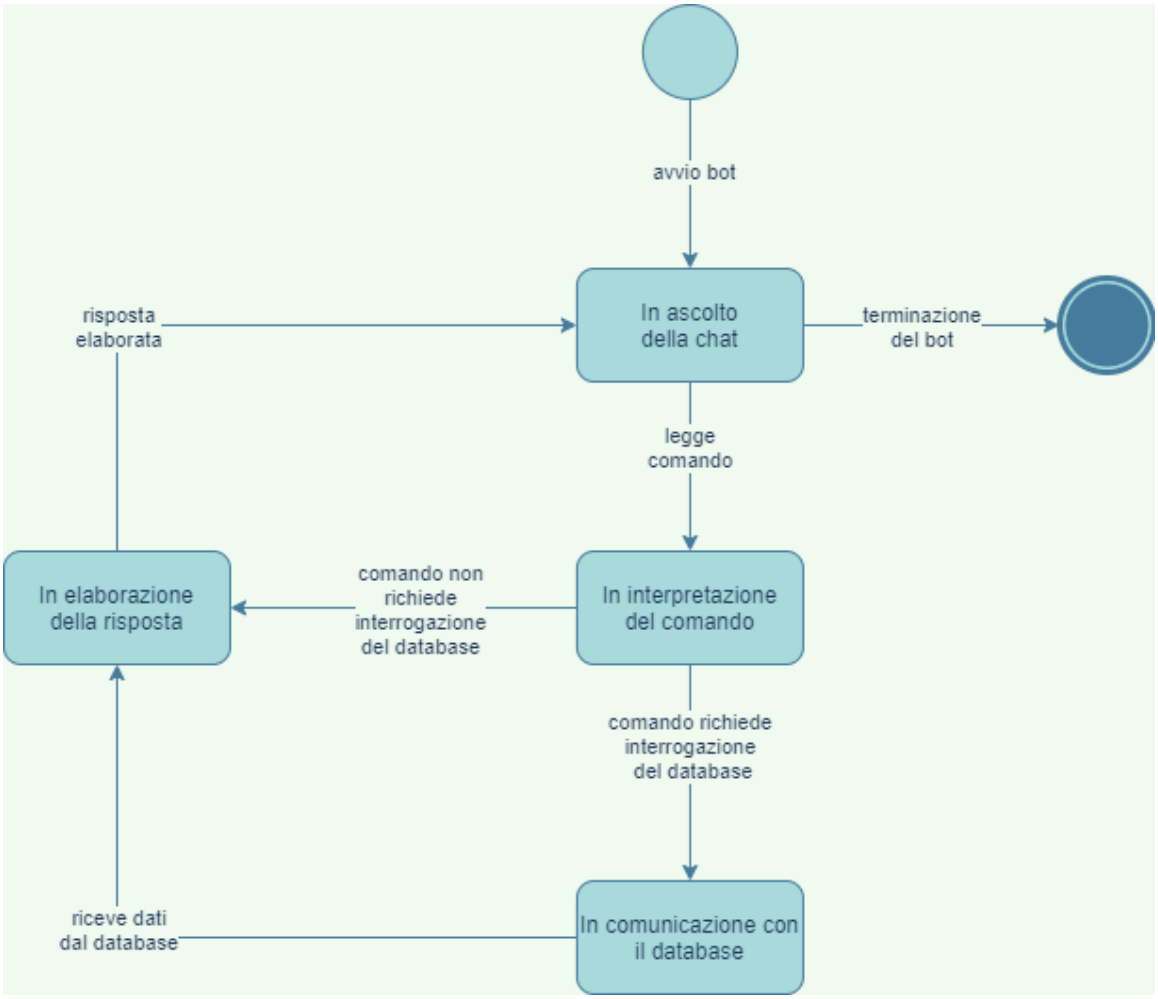
3.4 Diagramma di sequenza

Diagramma relativo all’esecuzione del comando /info invocato senza parametro.



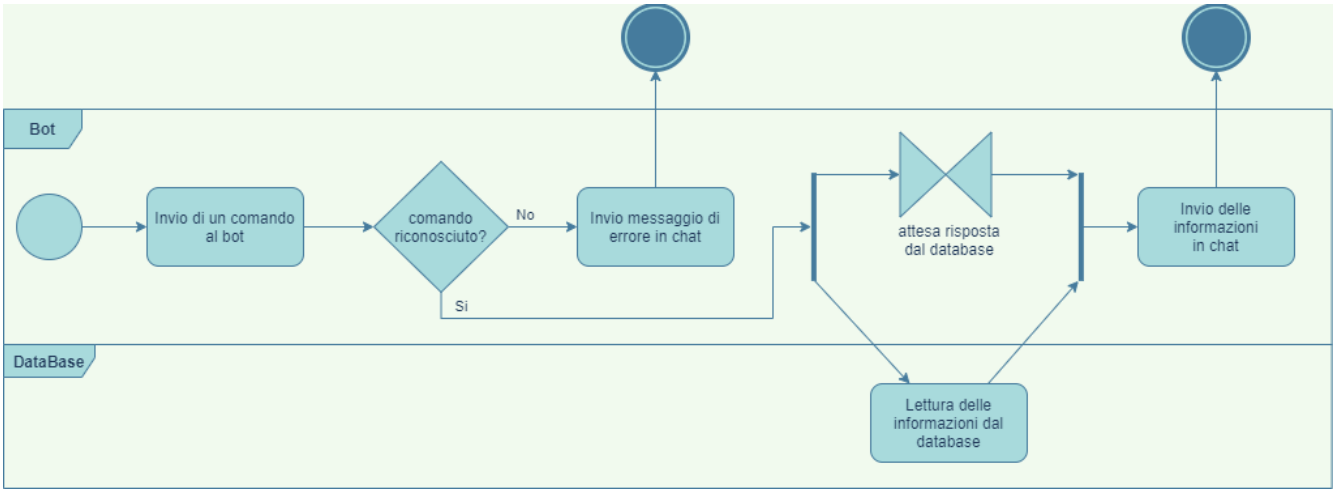
3.5 Diagramma di stato

Diagramma che descrive lo stato del bot durante l'esecuzione dell'applicazione.



3.6 Diagramma di attività

Diagramma che descrive l'esecuzione di un generico comando del bot con interrogazione del database.



4. Codice sorgente

Per il codice vedere la cartella BotLezioniUniPG o la repository su GitHub. Qui verranno riportate le query che sono utilizzate per interrogare il database durante l'esecuzione del bot:

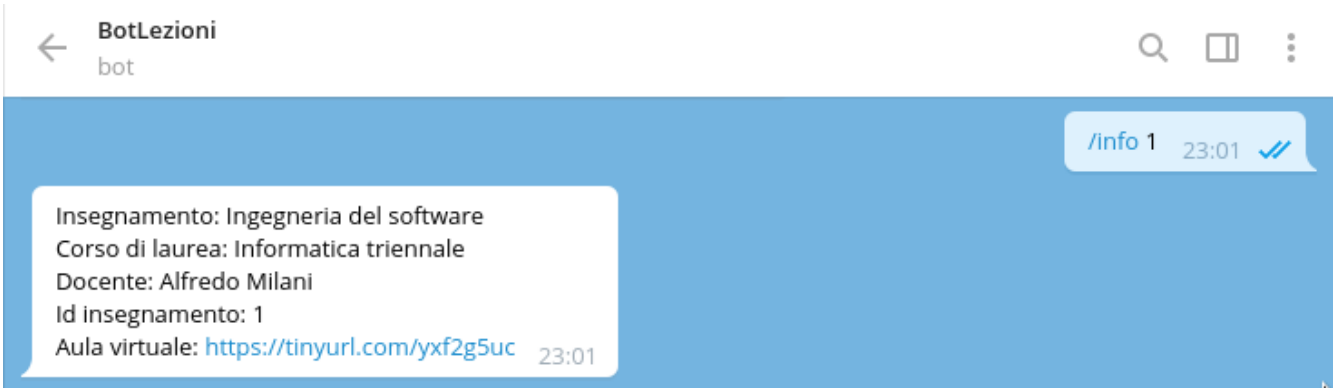
- **SELECT corso_di_laurea FROM Insegnamento GROUP BY corso_di_laurea**, restituisce un elenco di tutti i corsi di laurea.
- **SELECT nome_insegnamento, id_insegnamento FROM Insegnamento WHERE corso_di_laurea = "corso_x"**, restituisce id e nome dell'insegnamento corrispondenti ad un certo corso di laurea specificato.
- **SELECT * FROM Insegnamento INNER JOIN Docente ON Insegnamento.id_docente = Docente.id_docente WHERE Insegnamento.corso_di_laurea = "corso_x"**, restituisce informazioni sugli insegnamenti, e dei corrispondenti docenti titolari, di un particolare corso di laurea specificato.
- **SELECT * FROM Insegnamento INNER JOIN Docente ON Insegnamento.id_docente = Docente.id_docente WHERE Insegnamento.id_insegnamento = id_x**, restituisce informazioni sugli insegnamenti, e dei corrispondenti docenti titolari, di un insegnamento specificato.
- **SELECT * FROM Lezione INNER JOIN Insegnamento ON Insegnamento.id_insegnamento = Lezione.id_insegnamento INNER JOIN Docente ON Docente.id_docente = Lezione.id_docente WHERE Lezione.id_insegnamento = id_x**, restituisce informazioni relative alle lezioni di un particolare insegnamento specificato.
- **SELECT Insegnamento.id_insegnamento FROM Insegnamento INNER JOIN Docente ON Insegnamento.id_docente = Docente.id_docente WHERE LOWER(nome_insegnamento) LIKE '%parolachiave%' AND corso_di_laurea = "cdl_x"**, ritorna gli id degli insegnamenti di uno specificato corso di laurea il cui nome dell'insegnamento contiene una certa stringa specificata.
- **SELECT Insegnamento.id_insegnamento FROM Insegnamento INNER JOIN Docente ON Insegnamento.id_docente = Docente.id_docente WHERE LOWER(CONCAT(Docente.nome, ' ' ,Docente.cognome)) LIKE '%name%' AND corso_di_laurea = "cdl_x",** ritorna gli id degli insegnamenti di uno specificato corso di laurea, dove la stringa ottenuta dalla concatenazione del nome e del cognome del docente titolare contiene la stringa specificata.

5. Casi di test

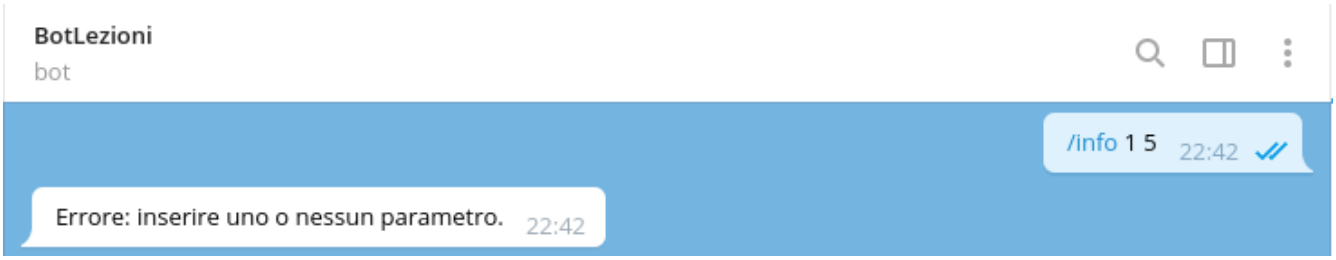
Sono stati svolti dei casi di test funzionali sui parametri passati al comando /info e sul parametro passato al comando /search.
La descrizione del comando /info specifica un range del numero di parametri che possono essere passati al comando. In particolare ha due diverse modalità di esecuzione, una quando non viene passato nessun parametro e l'altra quando viene passato un solo parametro. Vengono dunque individuate due classi di equivalenza:

- Una classe di test validi per un numero di parametri passati compresi tra 0 e 1.
- Una classe non validi per un numero di parametri passati compresi tra 2 e il massimo numero di parametri che è possibile inviare ad un bot in Telegram.

Seguendo la tecnica della copertura delle classi di equivalenza, sono sufficienti un test valido per la prima classe e uno non valido per la seconda.
Il test per la classe dei test validi:



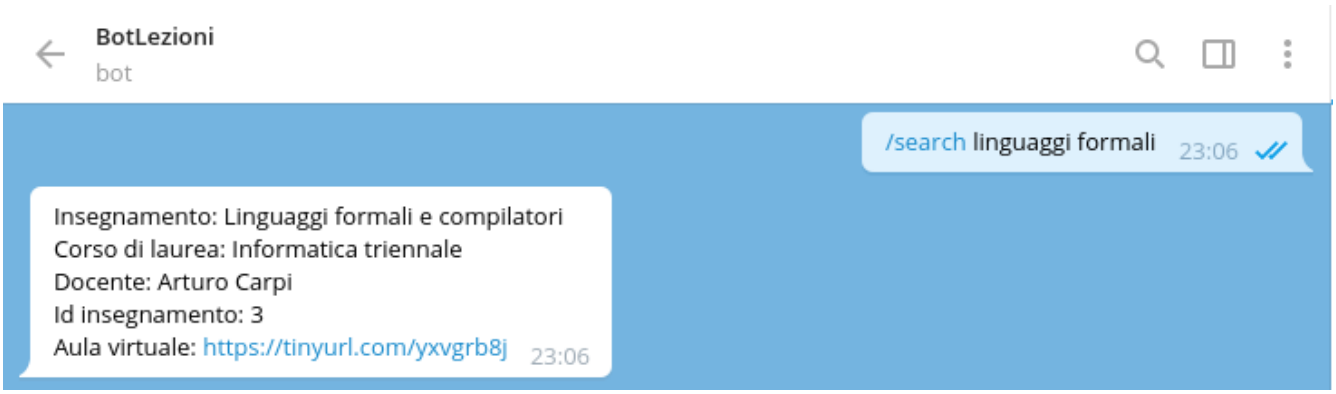
Il bot fornisce correttamente le informazioni dell'insegnamento corrispondente all'id passato.
Il test per la classe dei test non validi:



Il bot segnala correttamente l'eccessivo numero di parametri, specificando quello corretto.
Per il comando /search, onde evitare inserimenti di stringhe eccessivamente lunghe, si è imposto un limite sul numero dei caratteri del parametro di 500. Questo permette di individuare tre classi di equivalenza:

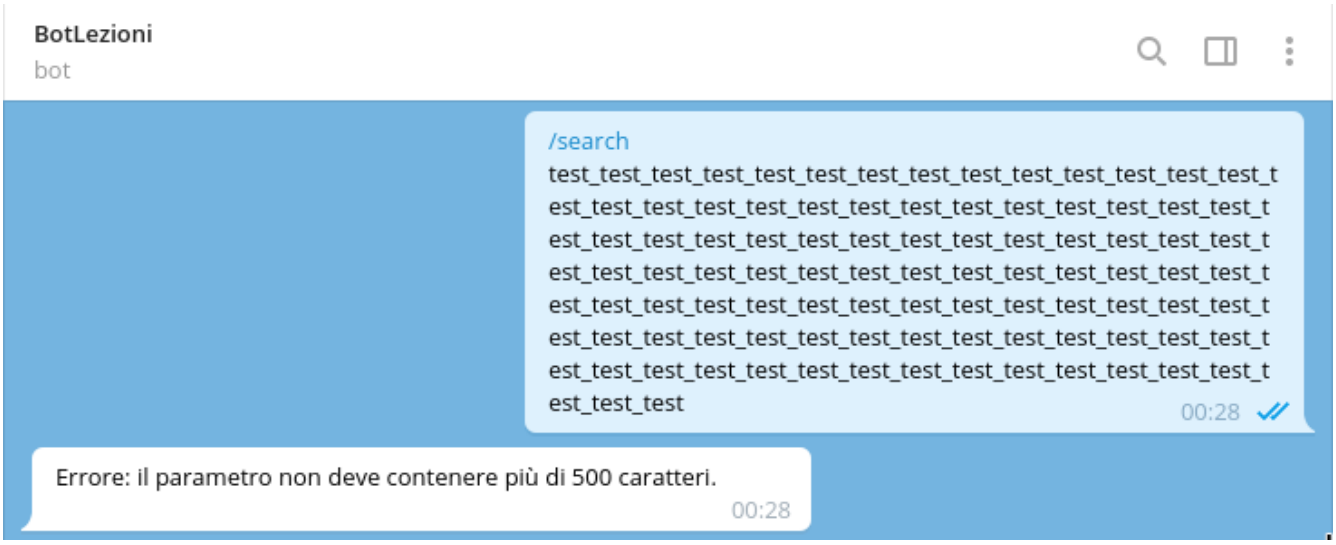
- Una classe di test validi per stringhe composte da un numero di caratteri compreso tra 1 e 500.
- Una classe di test non validi per stringhe composte da 0 caratteri (che equivale a non passare nessun parametro al comando).
- Una classe di test non validi per le stringhe composte da un numero di caratteri compreso tra 501 e il massimo numero di caratteri permessi per una stringa in Python.

Seguendo la tecnica della copertura di classi di equivalenza, è necessario un test per la classe dei test validi, e un test per ognuna delle due classi di test non validi.
Il test per la classe di test validi:



Il parametro passato contiene 17 caratteri, ed il bot risponde correttamente.
Il test per la prima classe di test non validi:

Il bot segnala correttamente l'assenza di parametri (quindi l'inserimento di una stringa di lunghezza 0).
Il test per la seconda classe di test non validi:



Il parametro inserito contiene 504 caratteri. Il bot segnala correttamente l'eccessiva lunghezza del parametro e specifica il limite massimo.

6. Design pattern

Il design pattern implementato nel progetto è il design pattern singleton, un design pattern creazionale che ha lo scopo di garantire l'esistenza di una sola istanza di una determinata classe all'interno di tutta l'applicazione, nel nostro caso tale classe sarà la classe DataBase. L'implementazione del design pattern avviene come segue:

```
5
6   class DataBase:
7
8
9       def __init__(self, user, host, db, password):
10          self.connection = self.db_connect(user, host, db, password)
11
12
13      @staticmethod
14      def get_instance():
15          global __instance
16          if not '_DataBase__instance' in globals():
17              __instance = DataBase('root','localhost','lezioni_db','')
18          return __instance
19
20
```

- Si crea un attributo statico e privato del tipo della classe presa in esame che conterrà l'istanza che si vuole garantire unica. Nel caso specifico l'attributo è stato chiamato __instance (Nota: in Python il doppio underscore come prefisso del nome di un attributo lo rende privato).
- Si crea un metodo statico pubblico che ha lo scopo di sostituire il metodo costruttore nella creazione di un'istanza della classe. Il metodo controlla se all'attributo che contiene l'istanza della classe è stata effettivamente assegnata un'istanza. Se non è stata assegnata la si assegna chiamando il metodo costruttore, altrimenti viene ritornata direttamente l'istanza della classe memorizzata dall'attributo. Nel caso specifico il metodo è stato chiamato get_instance(). Questo garantisce che l'unica istanza della classe all'interno dell'applicazione sarà quella che viene assegnata ad __instance la prima volta che il metodo get_instance() viene chiamato.
- Per una corretta implementazione del design pattern si dovrebbe anche rendere privato il metodo costruttore in maniera tale da rendere possibile la creazione di un oggetto della classe solo tramite il metodo get_instance(). Python però non permette di rendere privato il metodo costruttore di una classe, quindi questo aspetto del design pattern non è stato implementato.

Quando poi al di fuori della classe è necessaria un'istanza della classe DataBase si richiama il metodo get_instance() come nelle esempio che segue:

```
48  def lesson_command(update, context):
49      if len(context.args) == 1:
50          if check_if_int(context.args[0]):
51              db = DataBase.get_instance()
52              text = db.get_lessons(context.args[0])
53              update.message.reply_text(text)
```

7. Sitografia e fonti

https://github.com/python-telegram-bot/python-telegram-bot
https://python-telegram-bot.readthedocs.io/en/stable/
https://core.telegram.org/bots
https://www.w3schools.com/python/python_mysql_getstarted.asp