SNA4Slack gruppo Dijkstra

Indice

1.Introduzione

SNA4Slack del **gruppo Dijkstra** è un Software analitico a *command-line interface*. Come suggerisce il nome, SNA4Slack, presentiamo una *Social Network Analysis* adattata al Software Californiano Slack.

Lo **scopo principale** di una SNA è studiare intere strutture sociali (reti complete) o reti locali (reti egocentrate) individuando e analizzando tra individui o gruppi, i nodi della rete, le relazioni che sussistono tra loro,gli archi.

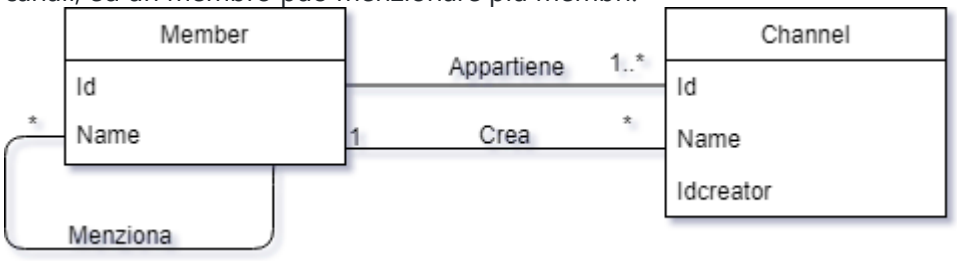
Nello specifico, il Software qui presentato permette di analizzare un workspace Slack, che può essere esportato dall'amministratore in formato .zip, ed avere informazioni sui canali pubblici. Il nostro **obiettivo** è stato quello di analizzare l'intero file .zip, mostrare all'utente una lista di canali membri e mentions, e dare la possibilità di potare il grafo con dei parametri. In particolare i filtri utilizzati sono i nomi di un utente, e i nomi di un canale, utilizzabili in base ai comandi che vengono descritti più nel particolare nel punto **7.Manuale Utente**

Riferimenti del progetto: sna4slack

2.Modello Concettuale

Ci sono due Classi: **Member** e **Channel**, con relazione *Appartenere* di molteplicità uno o più di uno, e con relazione *Creare* di molteplicità uno a molti. Member è in relazione ricorsiva con se stesso per mezzo di *Menzionare* di molteplicità zero o più di uno.

Ciò si traduce con: un membro appartiene ad uno o più canali, almeno un membro crea uno o più canali, ed un membro può menzionare più membri.



3.Requisiti Specifici

- Help
    - L'help può essere richiesto digitando il nome del programma senza parametri aggiuntivi
    - L'help è suggerito se un comando digitato non è valido
    - L'help è mostrato su standard output
    - L'help mostra i comandi per tutte le funzionalità
- Lista Members
    - Si può fare richiesta standard da input
    - Si può specificare il workspace
    - Vi è una cartella esportata associata al workspace
    - I membri sono visualizzati uno per riga
    - I membri del workspace sono tutti presenti
    - Non vi sono membri estranei al workspace
- Lista Channels
    - Si può fare richiesta standard da input
    - Si può specificare il workspace
    - Vi è una cartella esportata associata al workspace
    - I canali sono visualizzati uno per riga
    - I canali del workspace sono tutti presenti
    - Non vi sono canali estranei al workspace
- Lista Members di un Channel
    - Si può fare richiesta standard da input
    - Si può specificare il workspace
    - Vi è una cartella esportata associata al workspace
    - I membri e i canali sono visualizzati uno per riga, con i membri visualizzati subito dopo il canale a cui appartengono
    - È possibile distinguere quale nome è un "Member" e quale è un "Channel"
    - I canali del workspace sono tutti presenti
    - I membri del canale sono tutti presenti
    - Non vi sono canali o membri estranei al workspace
- Lista Members per Channel
    - Si può fare richiesta standard da input
    - Si può specificare il workspace
    - Vi è una cartella esportata associata al workspace
    - I membri e i canali sono visualizzati uno per riga, con i membri visualizzati subito dopo il canale a cui appartengono
    - È possibile distinguere quale nome è un "Member" e quale è un "Channel"
    - I canali del workspace sono tutti presenti
    - I membri del canale sono tutti presenti
    - Non vi sono canali o membri estranei al workspace
- Lista Mentions

    - Per ogni mention è visualizzata una riga con la coppia (From, To) dove From è lo User che scrive il messaggio con il mention e To è lo User menzionato.
    - Le coppie (From, To) non sono ripetute
    - Le coppie (From, To) corrispondenti a un mention sono tutte presenti
    - Sono visualizzate solo coppie (From, To) corrispondenti a un mention
    - È possibile specificare il Channel e, nel caso sia specificato, la lista è ristretta ai soli mention del Channel
- Lista Mentions che partono da uno User
    - È possibile specificare lo User da cui partono i mention
    - Per ogni mention è visualizzata una riga con la coppia (From, To) dove From è lo User che scrive il messaggio con il mention e To è lo User menzionato.
    - Le coppie (From, To) non sono ripetute
    - Le coppie (From, To) corrispondenti a un mention sono tutte presenti
    - Sono visualizzate solo coppie (From, To) corrispondenti a un mention
    - È possibile specificare il Channel e, nel caso sia specificato, la lista è ristretta ai soli mention del Channel
- Lista Mentions che arrivano ad uno User
    - È possibile specificare lo User a cui arrivano i mention
    - Per ogni mention è visualizzata una riga con la coppia (From, To) dove From è lo User che scrive il messaggio con il mention e To è lo User menzionato e specificato nel comando
    - Le coppie (From, To) non sono ripetute
    - Le coppie (From, To) corrispondenti a un mention sono tutte presenti
    - Sono visualizzate solo coppie (From, To) corrispondenti a un mention
    - È possibile specificare il Channel e, nel caso sia specificato, la lista è ristretta ai soli mention del Channel
- Lista Mentions Pesate
    - Per ogni mention è visualizzata una riga con la tripla (From, To, Weight) dove From è lo User che scrive il messaggio con il mention e To è lo User menzionato, e Weight il numero di mention
    - Le triple (From, To, Weight) non sono ripetute
    - Le triple (From, To, Weight) corrispondenti a un mention sono tutte presenti
    - Sono visualizzate solo triple (From, To, Weight) corrispondenti a un mention
    - È possibile specificare il Channel e, nel caso sia specificato, la lista è ristretta ai soli mention del Channel
- Lista Mentions Pesate che partono da uno User
    - È possibile specificare lo User da cui partono i mention
    - Per ogni mention è visualizzata una riga con la tripla (From, To, Weight) dove From è lo User che scrive il messaggio con il mention e To è lo User menzionato, e Weight il numero di mention
    - Le triple (From, To, Weight) non sono ripetute
    - Le triple (From, To, Weight) corrispondenti a un mention sono tutte presenti
    - Sono visualizzate solo triple (From, To, Weight) corrispondenti a un mention
    - È possibile specificare il Channel e, nel caso sia specificato, la lista è ristretta ai soli mention del Channel

- Lista Mentions Pesate che arrivano ad uno User
    - È possibile specificare lo User a cui arrivano i mention
    - Per ogni mention è visualizzata una riga con la tripla (From, To, Weight) dove From è lo User che scrive il messaggio con il mention e To è lo User menzionato, e Weight il numero di mention
    - Le triple (From, To, Weight) non sono ripetute
    - Le triple (From, To, Weight) corrispondenti a un mention sono tutte presenti
    - Sono visualizzate solo triple (From, To, Weight) corrispondenti a un mention
    - È possibile specificare il Channel e, nel caso sia specificato, la lista è ristretta ai soli mention del Channel

4.Architettura

Lo stile architetturale adottato è stato a strati (***Layered***), in quanto solo le interfacce delle classi principali vengono utilizzate. Il primo strato infatti è di presentazione, ed è posto nei package con dicitura `interface`. Le principali interfacce richiamano le classi individuate nel modello concettuale:

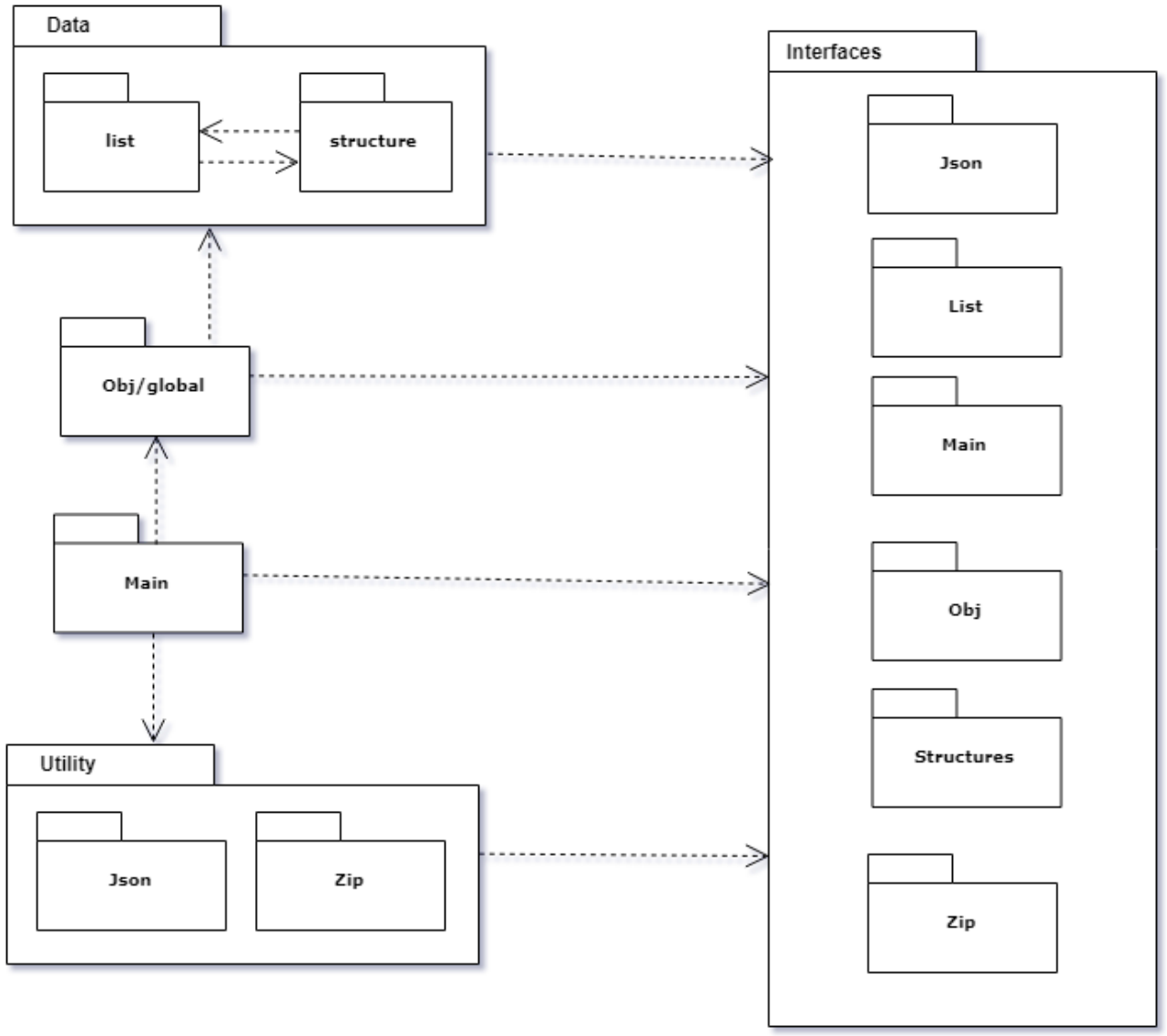
- **Users**: i membri del workspace sono identificati da un ID univoco e vari campi opzionali che corrispondono al nome. Noi abbiamo deciso di inserire durante il parsing del file .JSON il nome che l'utente mostra nello Slack UI, questo ha comportato una nostra decisione di controllo nel caso in cui non sia presente il parametro display_name. Tutti questi controlli sono stati nascosti attraverso l'uso dell'interfaccia, in modo da avere sia una demarcazione tra strato logico e strato esterno, sia una modificabilità estesa degli strati superiori che non influisca a cascata quelli inferiori.

- **Channels**: i canali del workspace sono identificati da un ID univoco, un nome ed una lista di Users. Ancora una volta mostriamo solo le interfacce di classi predisposte alla gestione di singoli canali e liste di utenti.

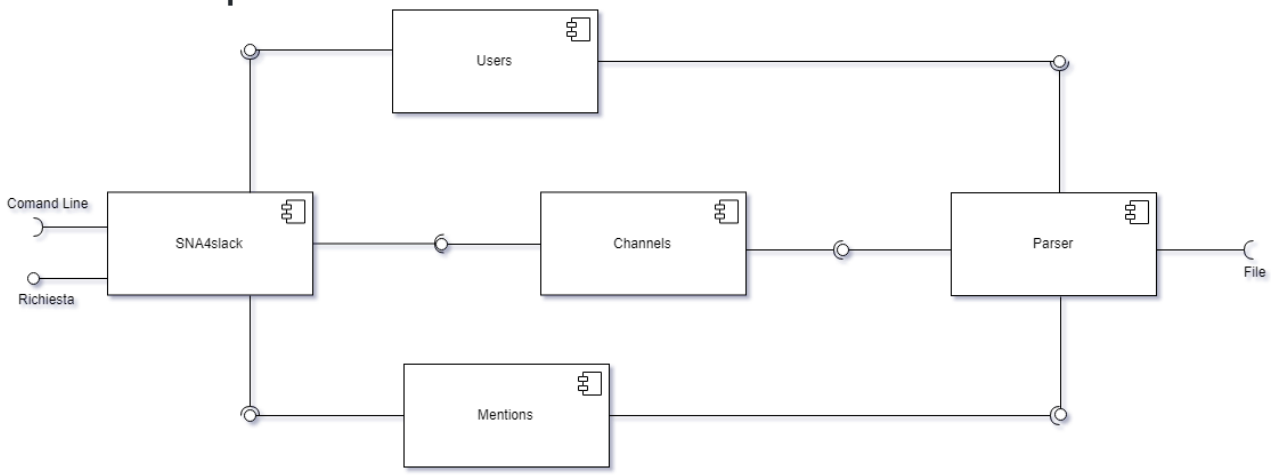
- **Mentions**: le menzioni, che dal modello concettuale hanno luogo dalla relazione ricorsiva di un utente, sono costituite da interfacce per la gestione dei mittenti e il/i proprio/i destinatatio/i. Per l'aggiunta della pesatura delle mentions, il sistema a strati ci ha permesso una piccola modifica del codice, utilizzando un HashMap formato da un intero che rappresenta il peso (il numero di menzioni scambiate da un utente ad un altro) e la propria chiave (una stringa che rappresenta il destinatario).

L'architettura gode di una statificazione lasca in modo tale da poter invocare operazioni di qualsiasi strato inferiore.

Diagramma dei Package:



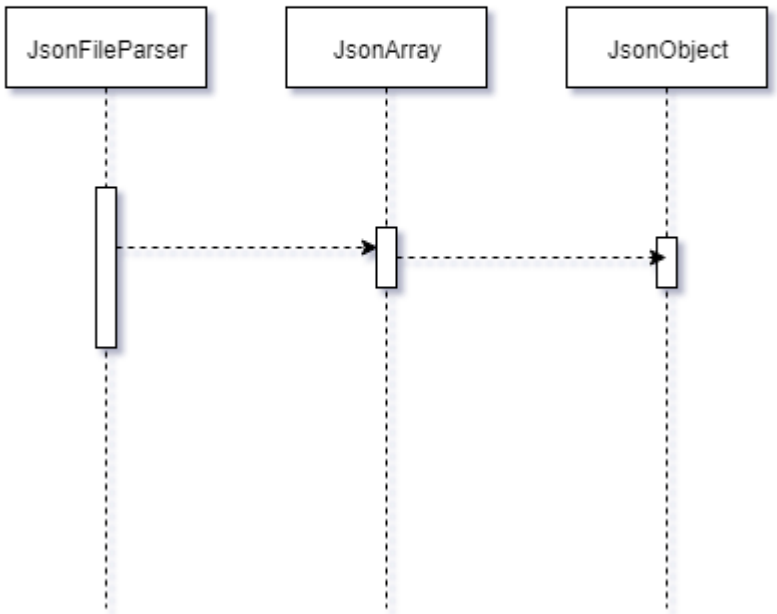
Modello dei Componenti:



L'utente fa una richiesta che arriva in input a SNA4Slack tramite command line, SNA4Slack riceve in input i dati dalle 3 classi principali che vengono a loro volta formate attraverso il parsing di file.

5.System Design

Parser



Sprint1

1. In qualita' di utente voglio visualizzare la lista dei Member
2. In qualita' di utente voglio visualizzare la lista dei Channel
3. In qualita' di utente voglio visualizzare la lista dei Member di un Channel
4. In qualita' di utente voglio visualizzare la lista dei Member raggruppati per Channel

Diagramma delle classi:

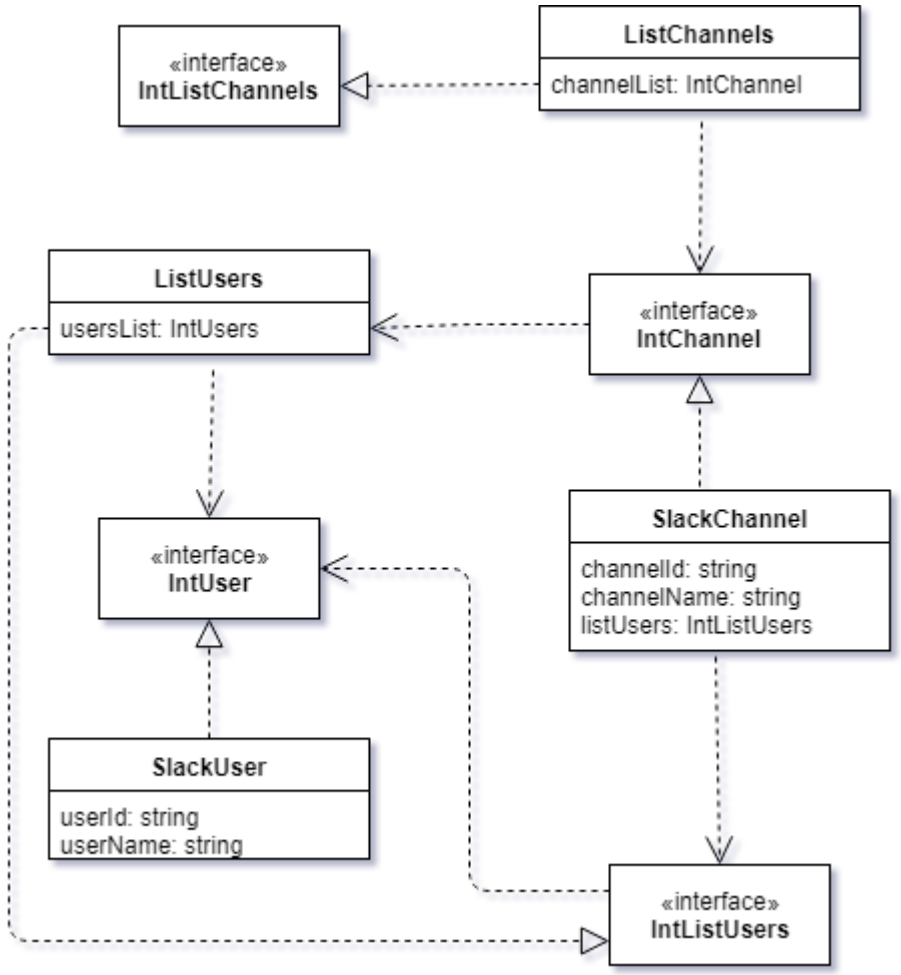
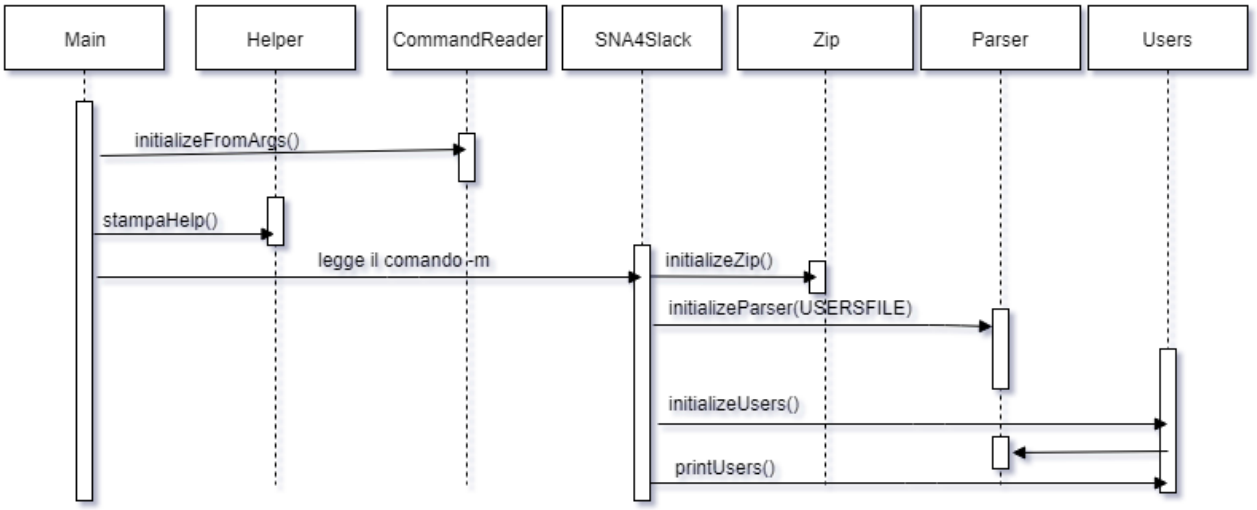
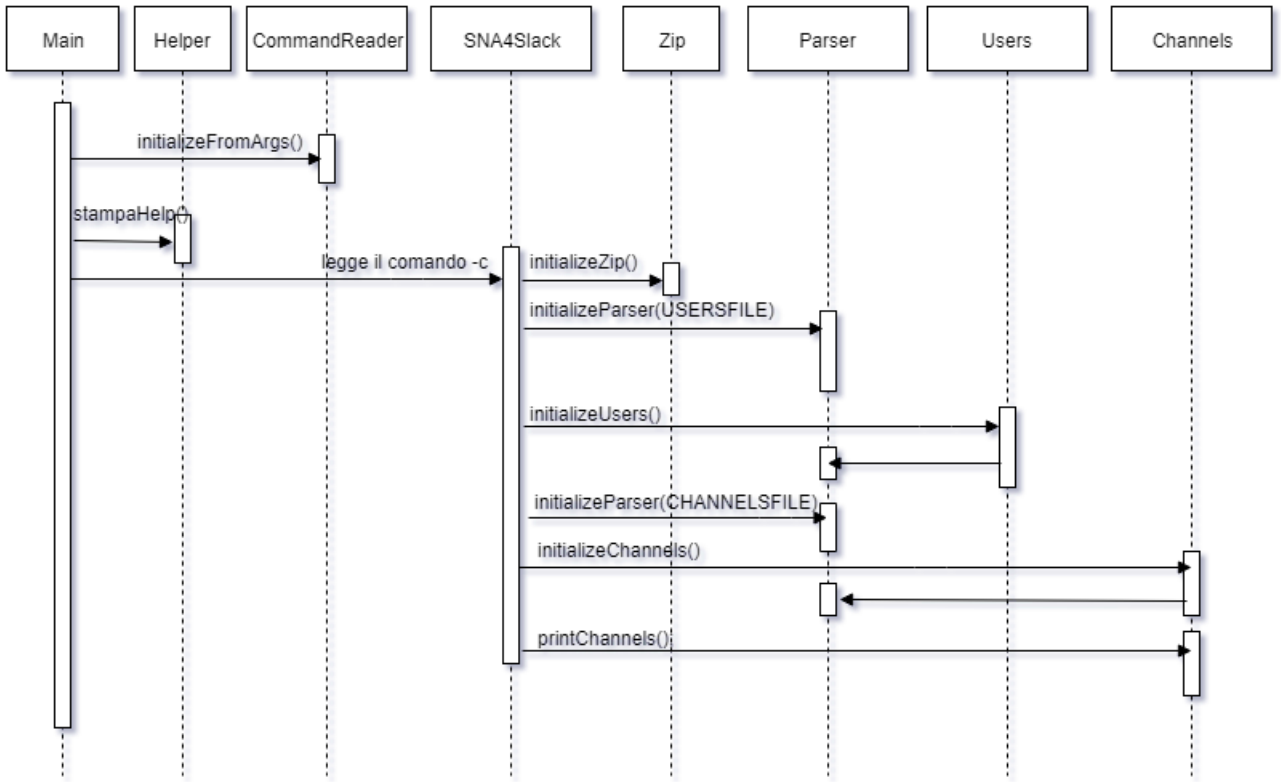
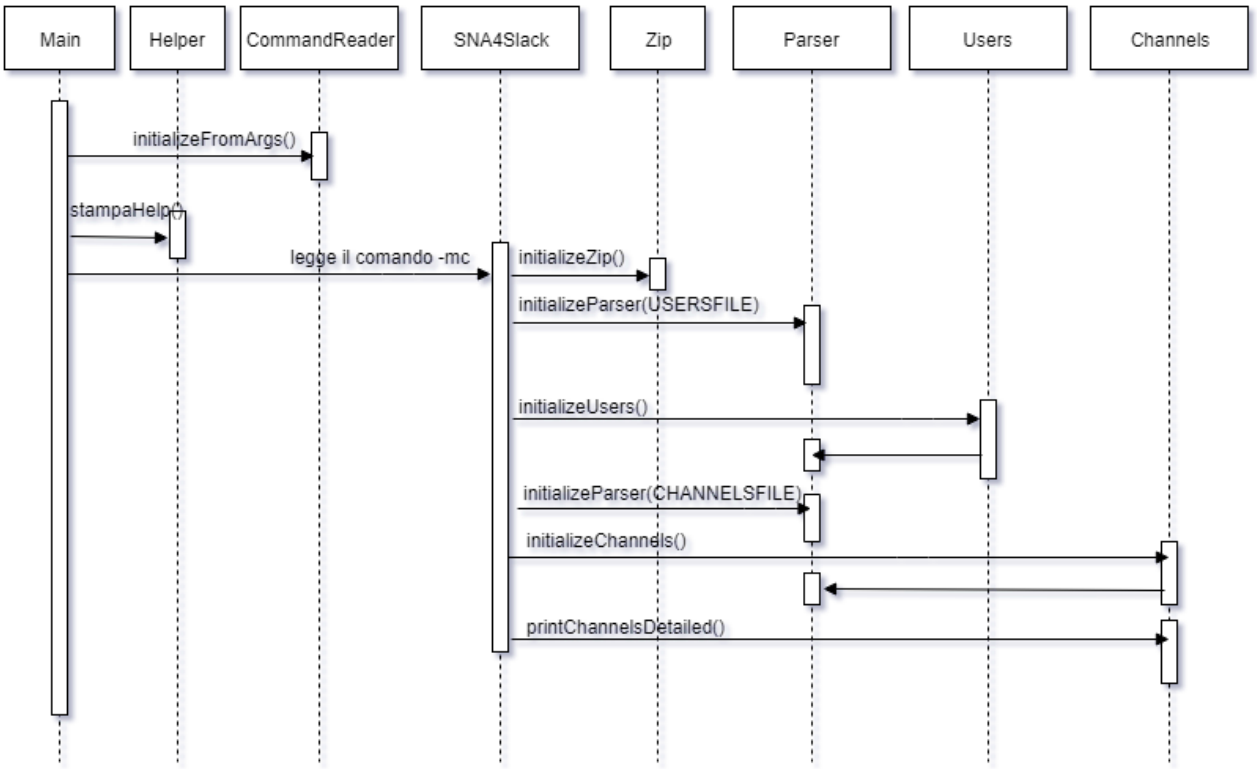
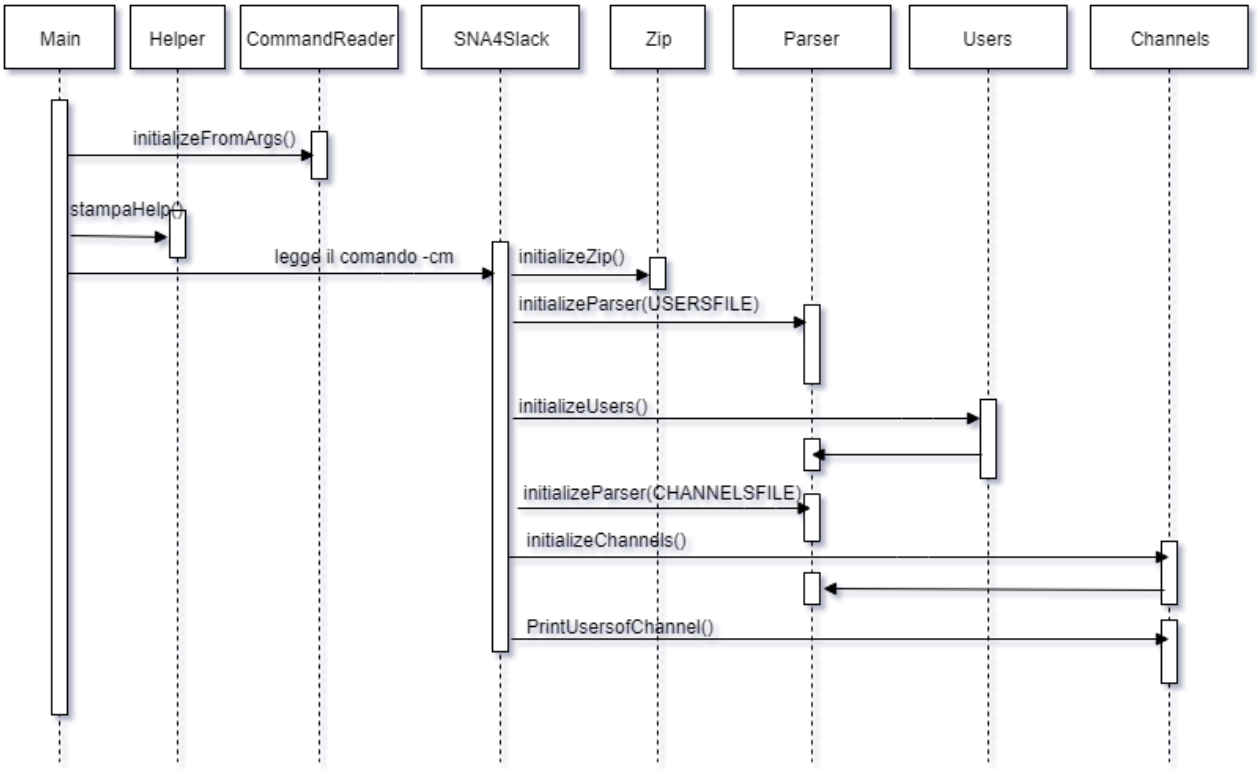


Diagramma di sequenza:




Main
Helper
CommandReader
SNA4Slack
Zip
Parser
Users
Channels
initializeFromArgs()
stampaHelp()
legge il comando -c
initializeZip()
initializeParser(USERSFILE)
initializeUsers()
initializeParser(CHANNELSFILE)
initializeChannels()
printChannels()





Sprint2

1. In qualita' di utente voglio visualizzare la lista dei @mention
2. In qualita' di utente voglio visualizzare la lista dei @mention che partono da uno user
3. In qualita' di utente voglio visualizzare la lista dei @mention che arrivano ad uno user

Diagramma delle classi:

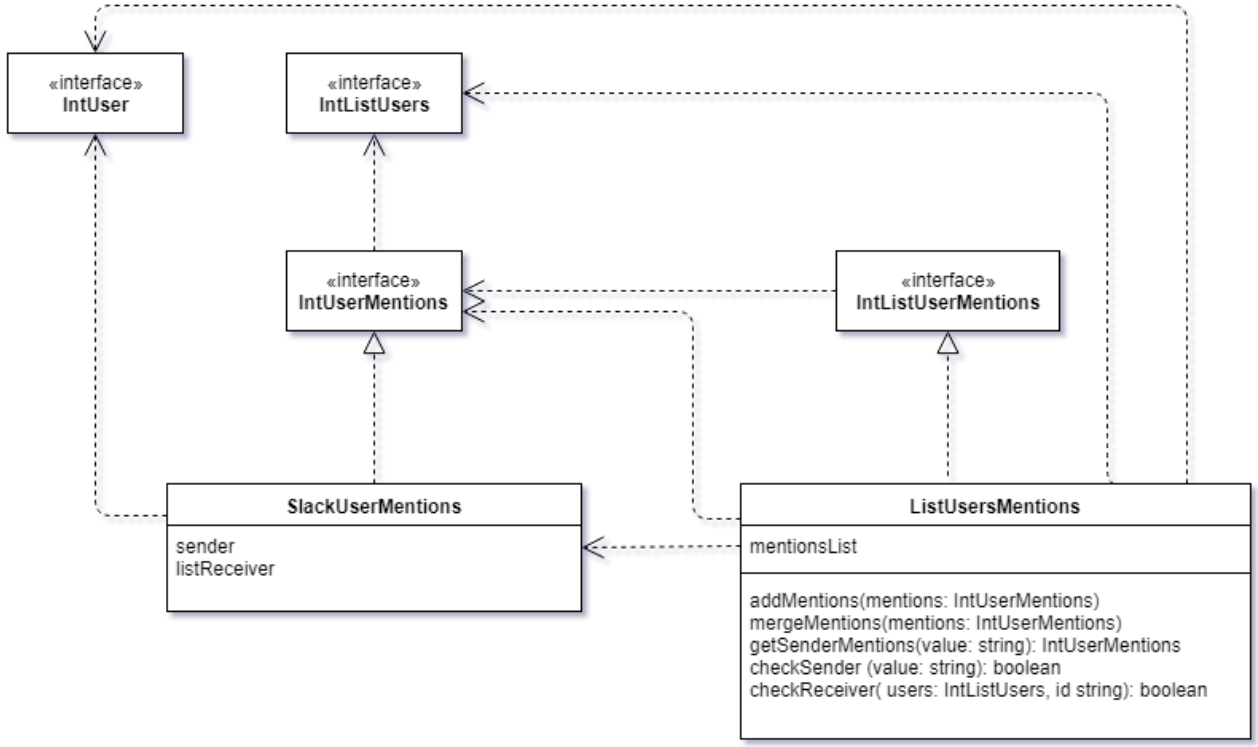
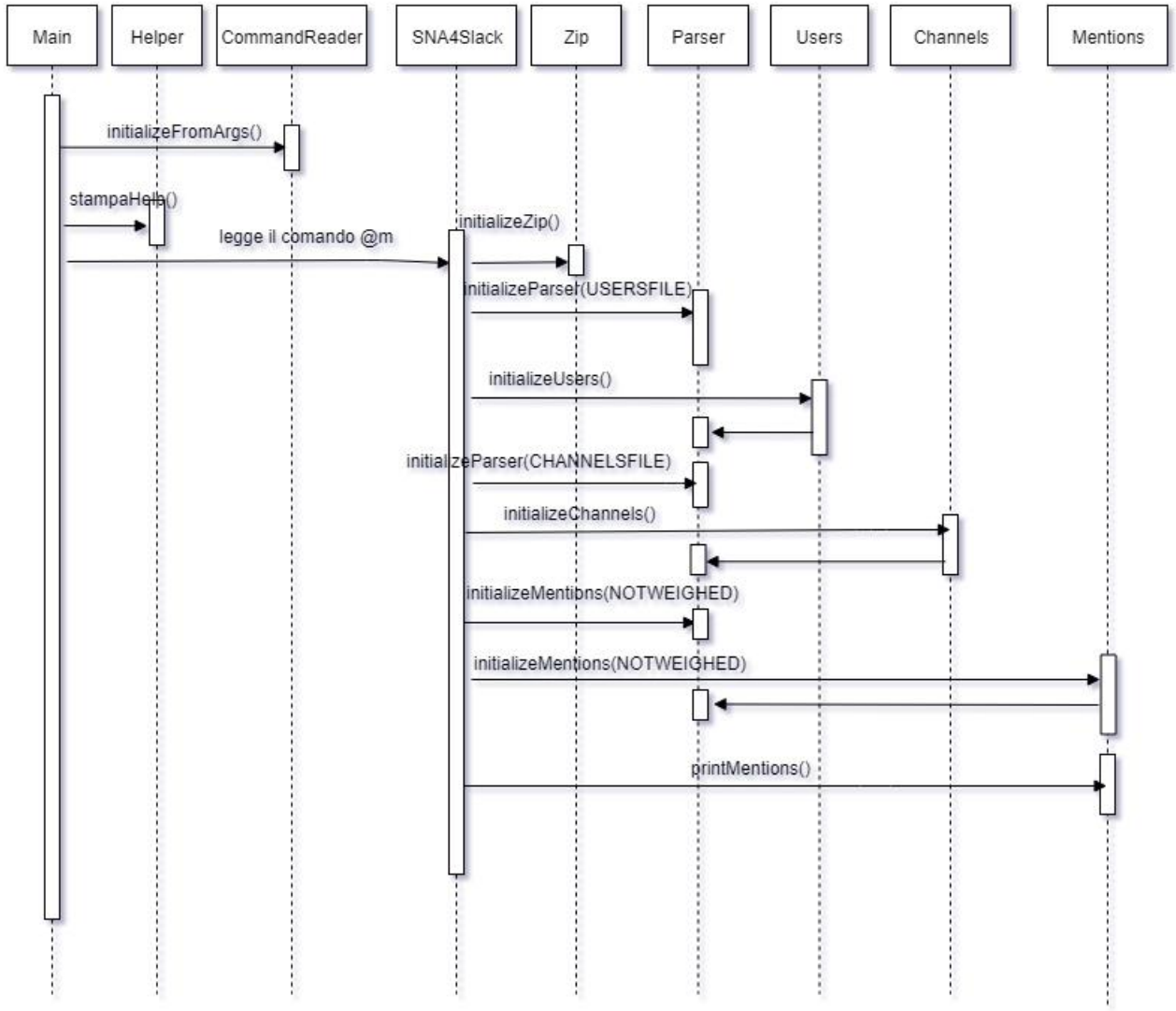
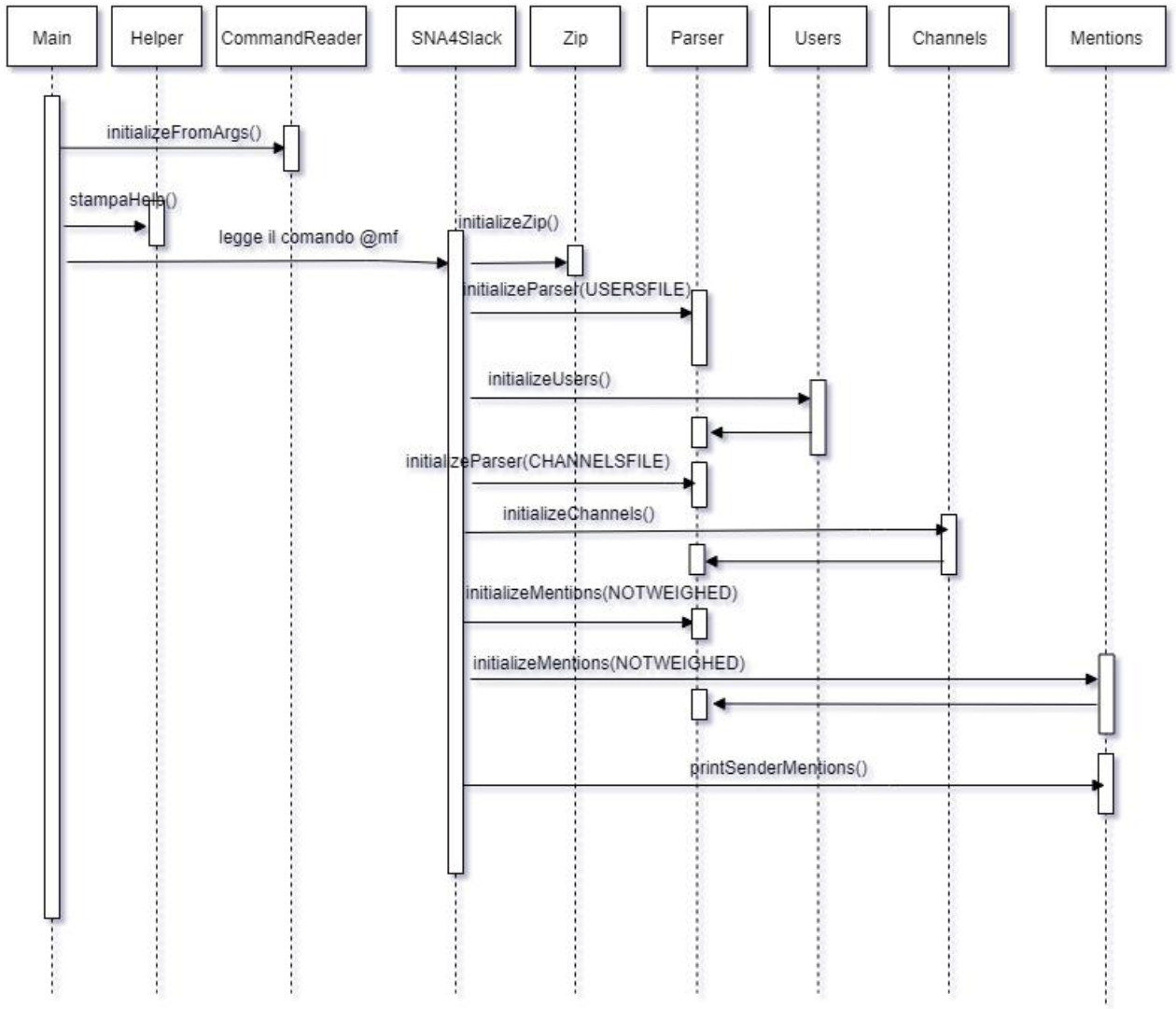
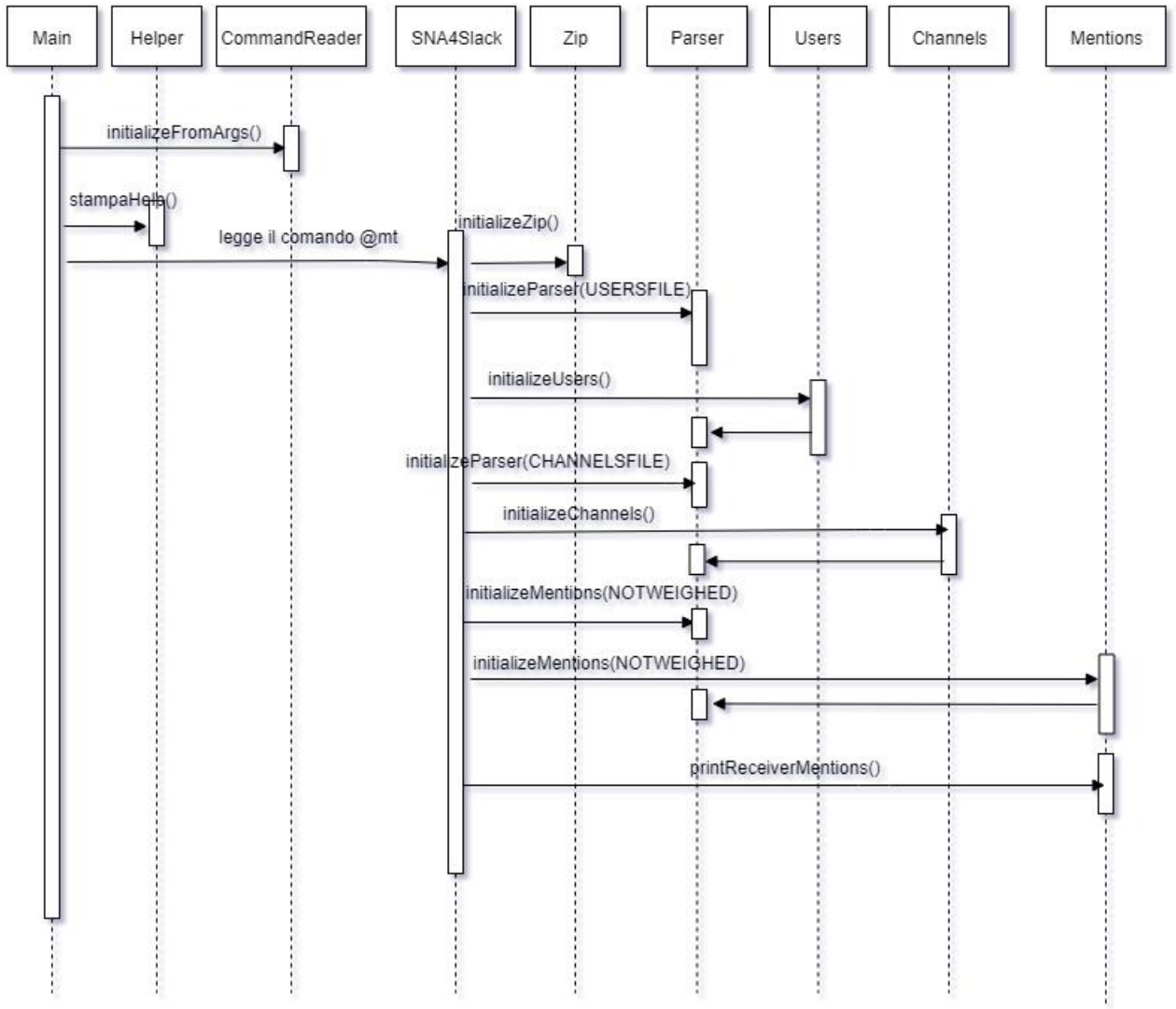


Diagramma di sequenza:




Main
Helper
CommandReader
SNA4Slack
Zip
Parser
Users
Channels
Mentions
initializeFromArgs()
stampaHelp()
legge il comando @mf
initializeZip()
initializeParser(USERSFILE)
initializeUsers()
initializeParser(CHANNELSFILE)
initializeChannels()
initializeMentions(NOTWEIGHED)
initializeMentions(NOTWEIGHED)
printSenderMentions()



Sprint3

1. In qualita' di utente voglio visualizzare la lista pesata dei @mention
2. In qualita' di utente voglio visualizzare la lista pesata dei @mention che partono da uno user
3. In qualita' di utente voglio visualizzare la lista pesata dei @mention che arrivano ad uno user

Diagramma delle classi:

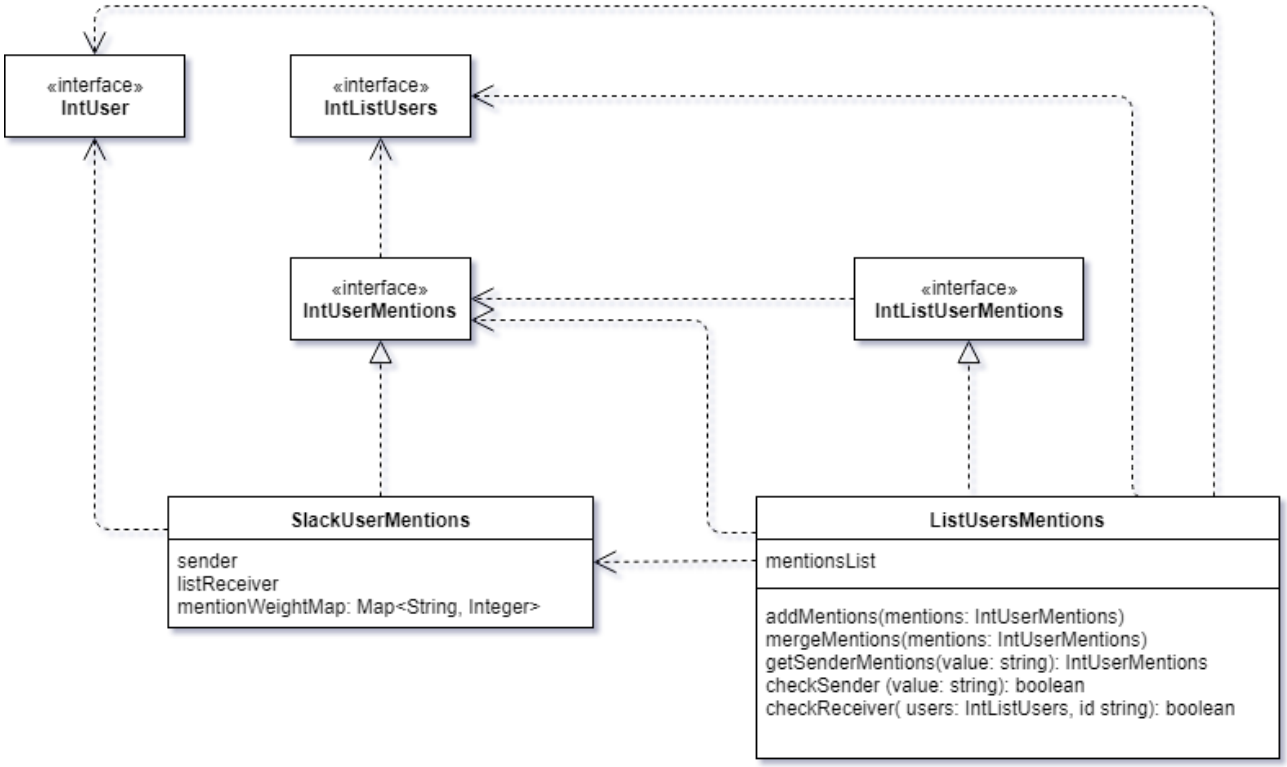
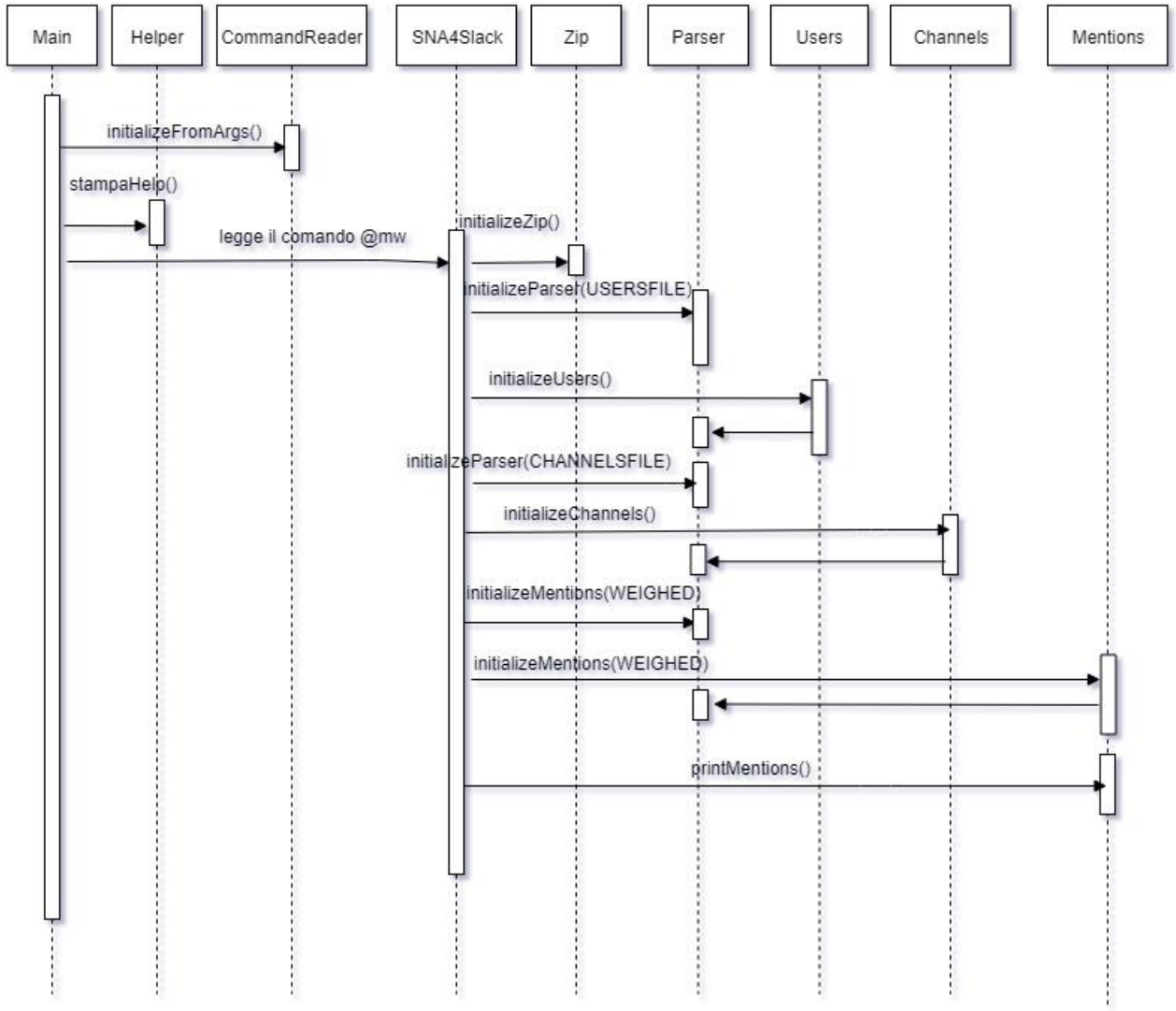
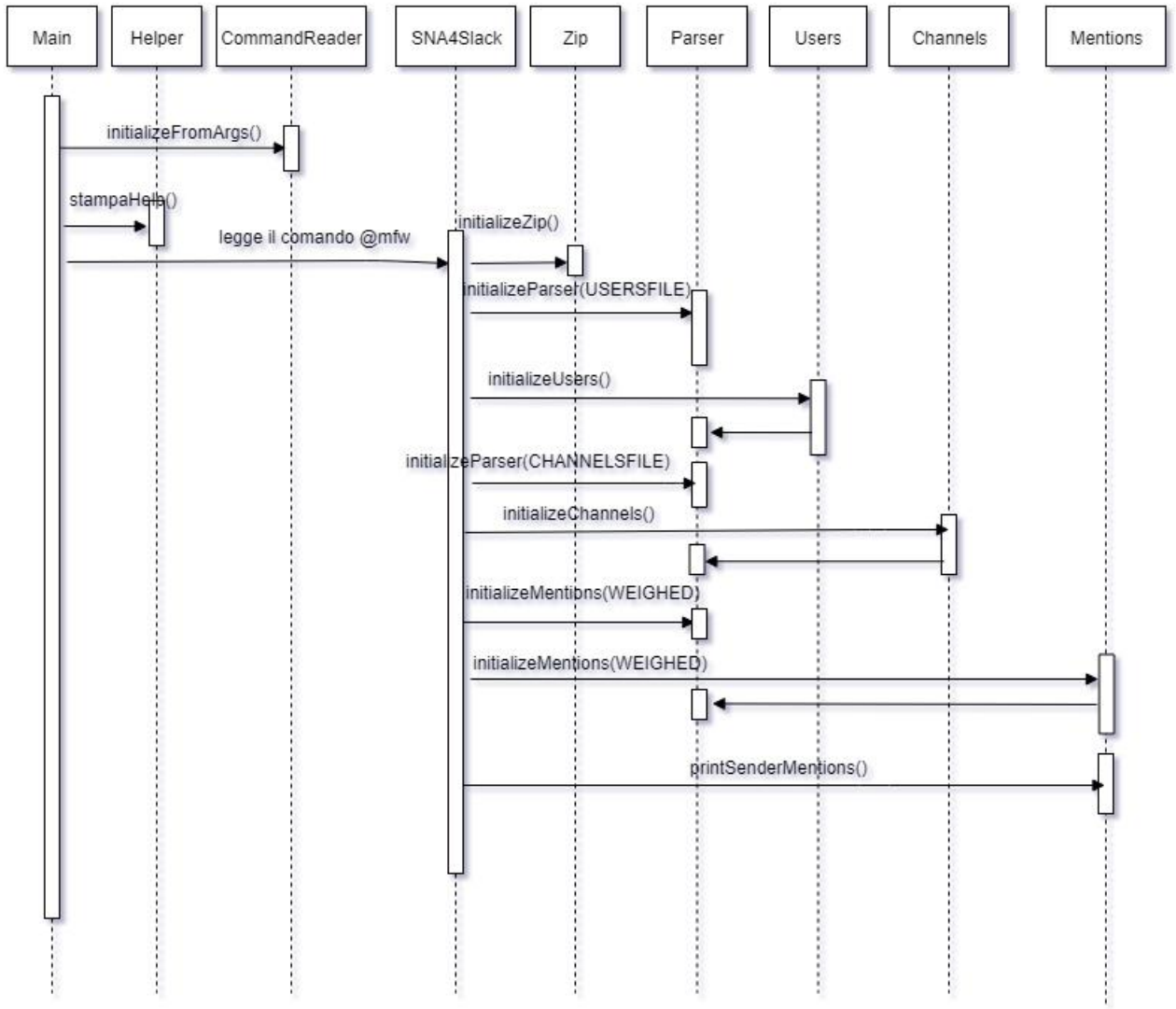
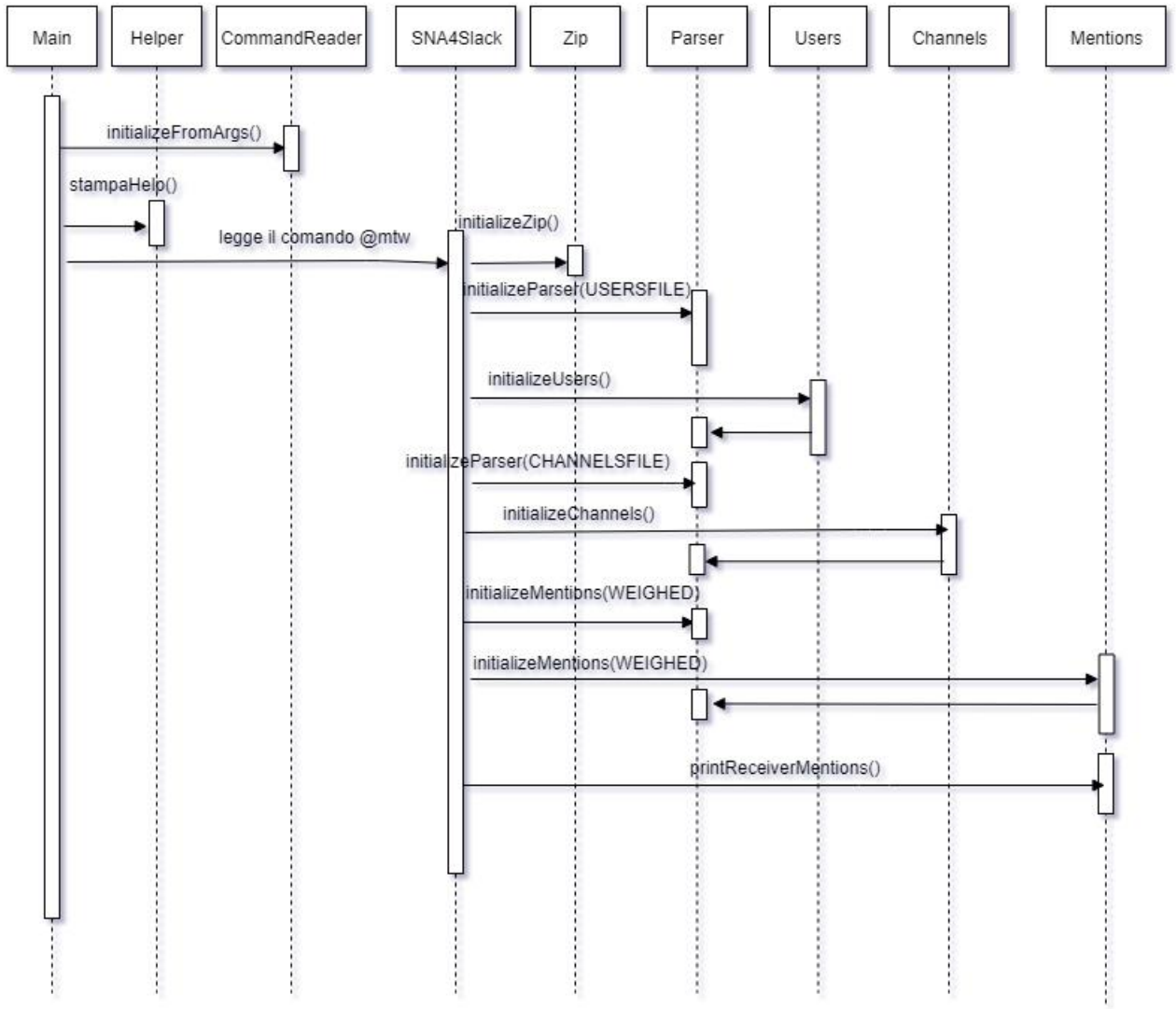


Diagramma di sequenza:




Main
Helper
CommandReader
SNA4Slack
Zip
Parser
Users
Channels
Mentions
initializeFromArgs()
stampaHelp()
legge il comando @mfw
initializeZip()
initializeParser(USERSFILE)
initializeUsers()
initializeParser(CHANNELSFILE)
initializeChannels()
initializeMentions(WEIGHED)
initializeMentions(WEIGHED)
printSenderMentions()



Design Pattern

È stato utilizzato Behavioral Design Pattern ***Strategy***, la classe SNA4Slack.java che si occupa dell'esecuzione del programma permette di gestire la scelta dell'algoritmo da utilizzare in base al contesto definito dall'utente, in questo caso il contesto è dato dai parametri.

Abbiamo raggiunto questo risultato con un `@Override runCommand()`, che modifica la sequenza algoritmica in modo tale da dare un output adeguato al "contesto" (Un oggetto che usa una Strategy è spesso identificato come oggetto "contesto"). Il nostro design Strategy prevede la memorizzazione dei casi in una struttura dati, e l'estrazione di quest'ultimi. La programmazione orientata ad oggetti permette l'uso di funzioni di prima classe che tratta le funzioni come cittadini di prima classe, questo significa che viene supportato il passaggio di funzioni come parametro ad altre funzioni. Nello specifico utilizziamo un `Map commandsTable <String, Command>` dove il primo parametro è il comando che inserisce l'Utente e il secondo, l'interfaccia Command, stabilisce l'esecuzione del comando.

PMD

Le uniche segnalazioni aperte lasciate riguardano la gestione del matcher, 4 `Avoid instantiating new objects inside loops` con bassa priorità, e la complessità ciclomatica di SNA4Slack dovuta al numero dei metodi e dei @Override.

6.Riepilogo Dei Test

Test Summary

92	0	0	9.063s
tests	failures	ignored	duration

100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
snaforslack.data.list.test	25	0	0	0.061s	100%
snaforslack.data.structures.test	23	0	0	0.028s	100%
snaforslack.main.test	25	0	0	8.450s	100%
snaforslack.obj.global.test	14	0	0	0.055s	100%
snaforslack.utility.json.test	1	0	0	0.001s	100%
snaforslack.utility.zip.test	4	0	0	0.468s	100%

SNA4Slack

SNA4Slack

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
it.uniba.main		100%		50%	1	2	0	6	0	1	0	1
snaforslack.data.list		99%		94%	2	43	1	72	0	25	0	3
snaforslack.data.structures		87%		73%	7	41	6	69	2	28	0	3
snaforslack.interfaces.main		100%		n/a	0	2	0	32	0	2	0	1
snaforslack.main		96%		80%	8	64	11	161	0	44	0	8
snaforslack.obj.global		100%		82%	9	46	0	106	0	20	0	3
snaforslack.utility.json		93%		76%	10	61	12	130	0	38	0	4
snaforslack.utility.zip		96%		50%	1	7	1	25	0	6	0	1
Total	90 of 2.310	96%	39 of 204	80%	38	266	31	601	2	164	0	24

7.Manuale Utente

L'utente può scaricare l'ultima immagine disponibile del programma attraverso Docker con il comando

```
docker pull softenginfuniba/dijkstra
```

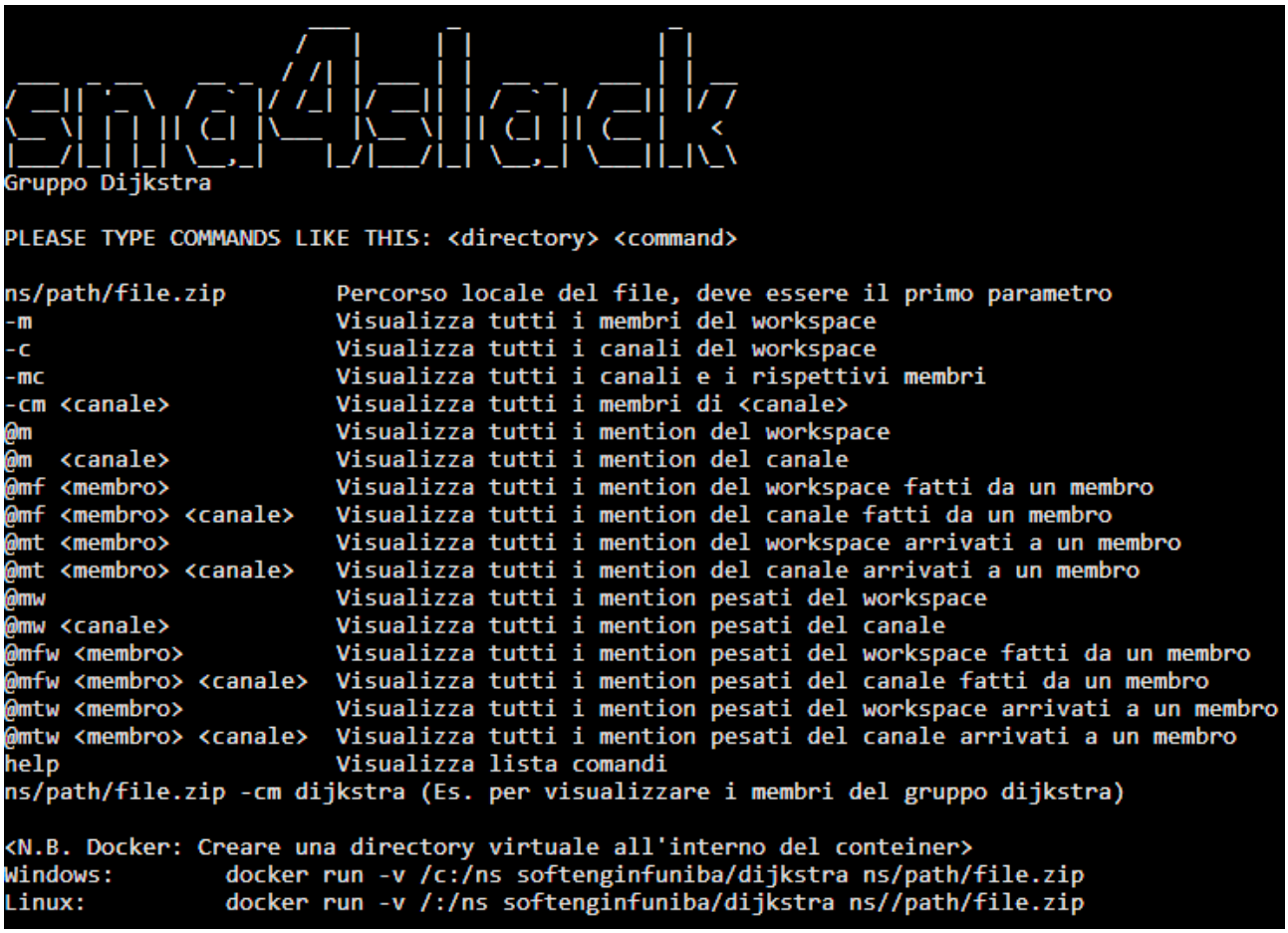
Per creare una directory virtuale all'interno del container, aggiungere "*-v /c:/ns -it*"

```
docker run -v /c:/ns -it softenginfuniba/dijkstra ns/
```

Un comando prevede il percorso assoluto o realtivo del workspace in formato .zip come primo parametro.

```
docker run -v /c:/ns -it softenginfuniba/dijkstra ns/path/file.zip <comandi>
```

A questo si possono giustapporre i comandi mostrati in figura.



8.Processo Di Sviluppo e Organizzazione Del Lavoro

Dopo un'analisi iniziale assieme al nostro *product-owner* si è stabilito di proseguire con uno **sviluppo agile** il quale ha permesso una divisione naturale del carico di lavoro in 5 iterazioni fondamentali, e con un gradiente di difficoltà crescente che ha permesso di avere un ritmo di lavoro costante. Le metodologie agili usate sono state Scrum e Kanban tramite una tabella virtuale offerta da GitHub.

Per quanto riguarda **Scrum** si è seguito il modello: vi è stato un *Product Backlog* mostratoci dal nostro *product-owner* ad inizio progetto, successivamente il lavoro è stato diviso in sprint di circa 10 giorni con uno *Sprint Backlog* ad inizio di ognuno, in fine il gruppo Dijkstra ha effettuato quando possibile dei *Daily Scrum Meeting* sia face-to-face che con comunicazioni online.

Ad ogni sprint abbiamo provveduto all'assegnazione delle *User-Story* dividendole in base alla loro difficoltà e l'esperienza dei membri del team. Lavorando lontani l'uno dagli altri, siamo riusciti a controllare lo svolgimento effettivo del lavoro, anche prima del meeting giornaliero, attraverso il **Kanban** virtuale ed automatico offertoci da Github.

Il gruppo Dijkstra è formato da quattro membri : Amendola Piera, Angelini Giuseppe, Chekalin Nazar, Lapenta Francesco Davide. Inizialmente i membri erano cinque, ma dopo una decisione di gruppo abbiamo deciso di espellerne uno, per il suo scarso contributo e poca partecipazione. Quest'ultima decisione è stata presa la settimana precedente alla consegna finale, in modo da offrire un possibilità anche con pair-programming, che però non è stata colta.

9.Analisi Retrospettiva

- *Cosa da tenere*: La comunicazione tramite Slack con tutti i gruppi è stata un'esperienza positiva. Le soluzioni offerte da alcuni membri hanno permesso che non rallentassimo il lavoro, sia per problemi che inizialmente non ci eravamo posti sia per problemi a cui cercavamo risposta.

- *Problemi*: Scarsa comunicazione inter gruppo. Ci siamo resi conto che il daily scrum meeting è fondamentale, il nostro gruppo non è riuscito nella riunione giornaliera per via delle lezioni mattutine e pomeridiane. A questa mancanza abbiamo ovviato con riunioni meno sporadiche e più corpose.

- *Cosa da provare*: Aiutare e invogliare i gruppi al daily scrum meeting durante le pause delle lezioni.