



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Corso di  
Ingegneria, Gestione ed Evoluzione del Software



## Pre-Maintenance Report

DOCENTE

**Prof. Andrea De Lucia**

LINK REPOSITORY - BRANCH "DEV"

**CADOCS:** <https://github.com/alfcan/CADOCS/tree/dev>

**CADOCS\_NLU:** [https://github.com/alfcan/CADOCS\\_NLU\\_Model/tree/dev](https://github.com/alfcan/CADOCS_NLU_Model/tree/dev)

**csDetector:** <https://github.com/alfcan/csDetector/tree/dev>

AUTORI

**Alfonso Cannavale**

Matricola: 0522501597

**Davide La Gamba**

Matricola: 0522501464

**Kevin Pacifico**

Matricola: 0522501501

# Indice

<b>1</b>	<b>Architettura software</b>	<b>1</b>
<b>2</b>	<b>Reverse Engineering</b>	<b>5</b>
2.1	Package Diagram . . . . .	5
2.2	Class Diagram . . . . .	5
2.3	Use Case . . . . .	9
2.3.1	Get Smells . . . . .	10
2.3.2	Get Smells Date . . . . .	11
2.3.3	Info . . . . .	12
2.3.4	Report . . . . .	13
2.4	Sequence Diagram . . . . .	14
2.4.1	Get Smells e Get Smells Date . . . . .	14
2.4.2	Info . . . . .	14
2.4.3	Report . . . . .	15
2.4.4	Modello NLU . . . . .	16

# Capitolo 1

## Architettura software

CADOCS è un agente conversazionale disponibile sulla piattaforma Slack grazie al quale gli utenti possono inserire delle richieste ed usufruire delle funzionalità.

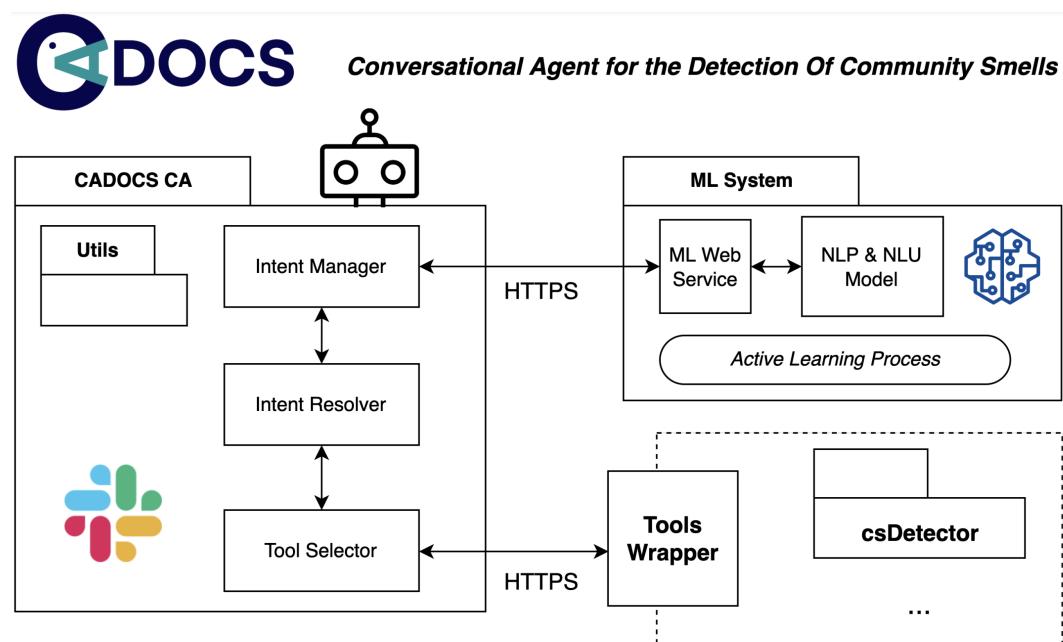
CADOCS si divide in tre moduli:

- *CADOCS\_NLU*: questo modulo è composto da un modello di Machine Learning per il Natural Language Understanding. Il modello permette di riconoscere e classificare le richieste degli utenti.
- *csDetector*: questo modulo è un tool esterno che permette di analizzare delle repository restituendone i community smells identificati.
- *CADOCS*: questa è la parte centrale del bot che interagisce con gli utenti nelle workspace di Slack tramite l'API Slack. Gestisce i messaggi degli utenti e tramite il modulo CADOCS\_NLU riesce ad assegnare un intent alla richiesta. Successivamente si occupa di inoltrare la richiesta al modulo csDetector e processare la risposta all'utente.

Nella Figura 1.1<sup>1</sup> è illustrata una panoramica del sistema, estratta dalla documentazione presente sulla repository originale.

---

<sup>1</sup>G. Voria et al., "Community Smell Detection and Refactoring in SLACK: The CADOCS Project," 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), Limassol, Cyprus, 2022, pp. 469-473, doi: 10.1109/ICSME55016.2022.00061

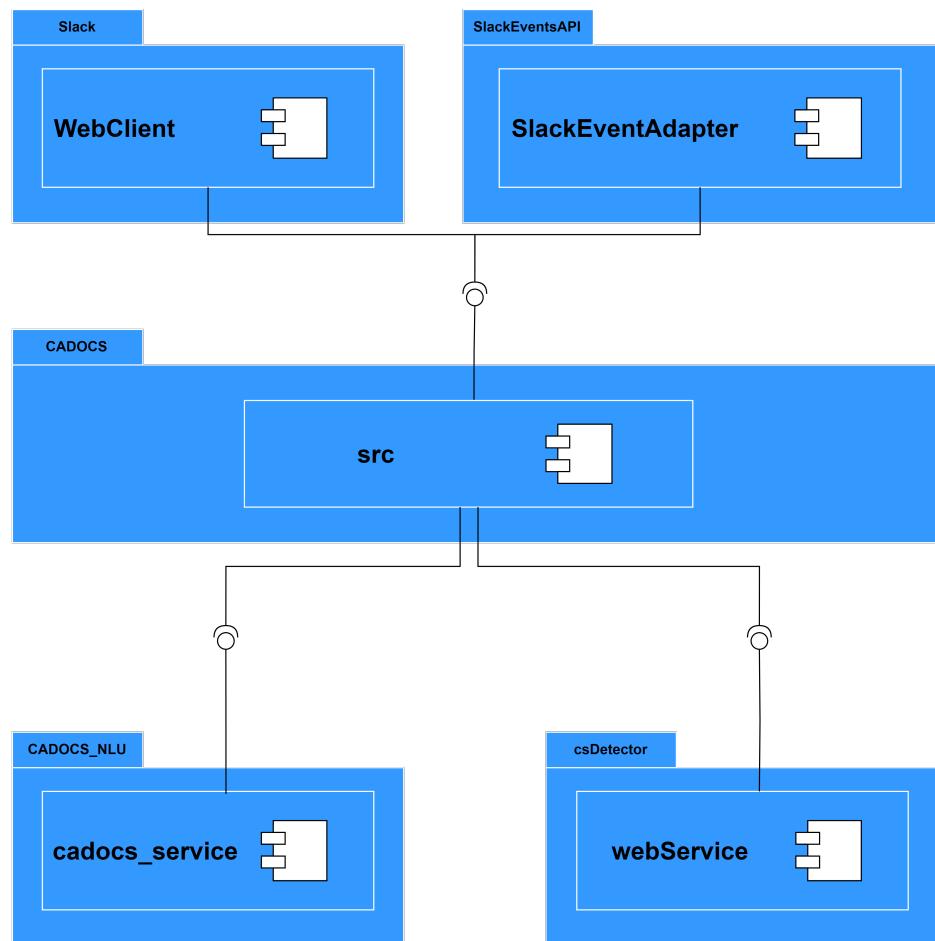


**Figura 1.1:** Panoramica del sistema

Esistono quattro tipi di intent:

- **get\_smells**: funzionalità che permette di analizzare una repository fornita in input, restituendo i relativi community smells identificati dal tool *csDetector*. Questa funzione consente agli utenti di ottenere un resoconto dettagliato sui community smells presenti nel progetto analizzato.
- **get\_smells\_date**: funzionalità che permette di analizzare una repository fornita in input da una data specificata in poi, restituendo i relativi community smells identificati dal tool *csDetector*. In questo modo, gli utenti possono ottenere informazioni dettagliate riguardanti i community smells emersi dopo una data fornita.
- **report**: funzionalità che permette all’utente di visualizzare l’ultima analisi effettuata durante la conversazione con il chatbot CADOCS.
- **info**: funzionalità che permette all’utente di visualizzare le informazioni relative ai community smells che il chatbot può identificare.

L’architettura di partenza estratta durante la fase di Reverse Engineering è riportata in Figura 1.2 e mostra le relazioni tra i vari moduli dell’applicazione, comprendendo anche le piattaforme esterne come quelle relative a Slack.



**Figura 1.2:** Architettura del sistema attuale

# Capitolo 2

## Reverse Engineering

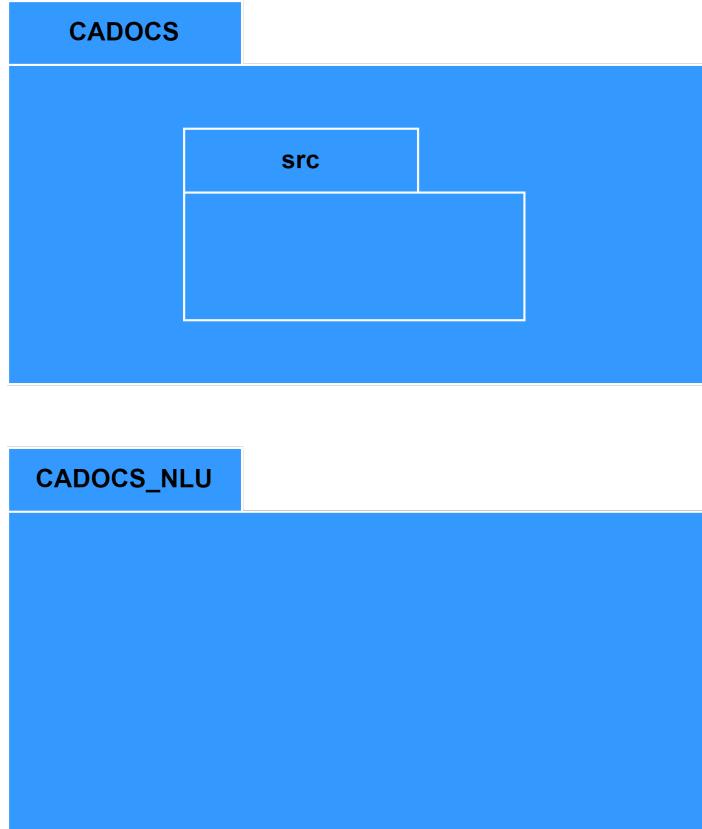
### 2.1 Package Diagram

Il progetto CADOCS contiene tutta la logica in un unico package *src*, che gestisce tramite Slack le richieste degli utenti ed interagisce con CADOCS\_NLU e con csDetector. Il progetto CADOCS\_NLU contiene tutta la logica per la classificazione delle richieste tramite un modello di Natural Language Understanding, allo stato attuale tutti i file sono nella root del progetto. I package diagram ottenuti dalla fase di Reverse Engineering sono riportati in Figura 2.1.

### 2.2 Class Diagram

Il class diagramma estratto, rappresentato in figura 2.2, fornisce una panoramica delle relazioni tra diverse classi e moduli nel sistema CADOCS, con particolare attenzione all’interfacciamento con Slack e all’utilizzo del tool csDetector per l’analisi delle repository GitHub. Il class diagram è stato analizzato nel dettaglio e di seguito vengono riportati i punti chiave per la comprensione del modello:

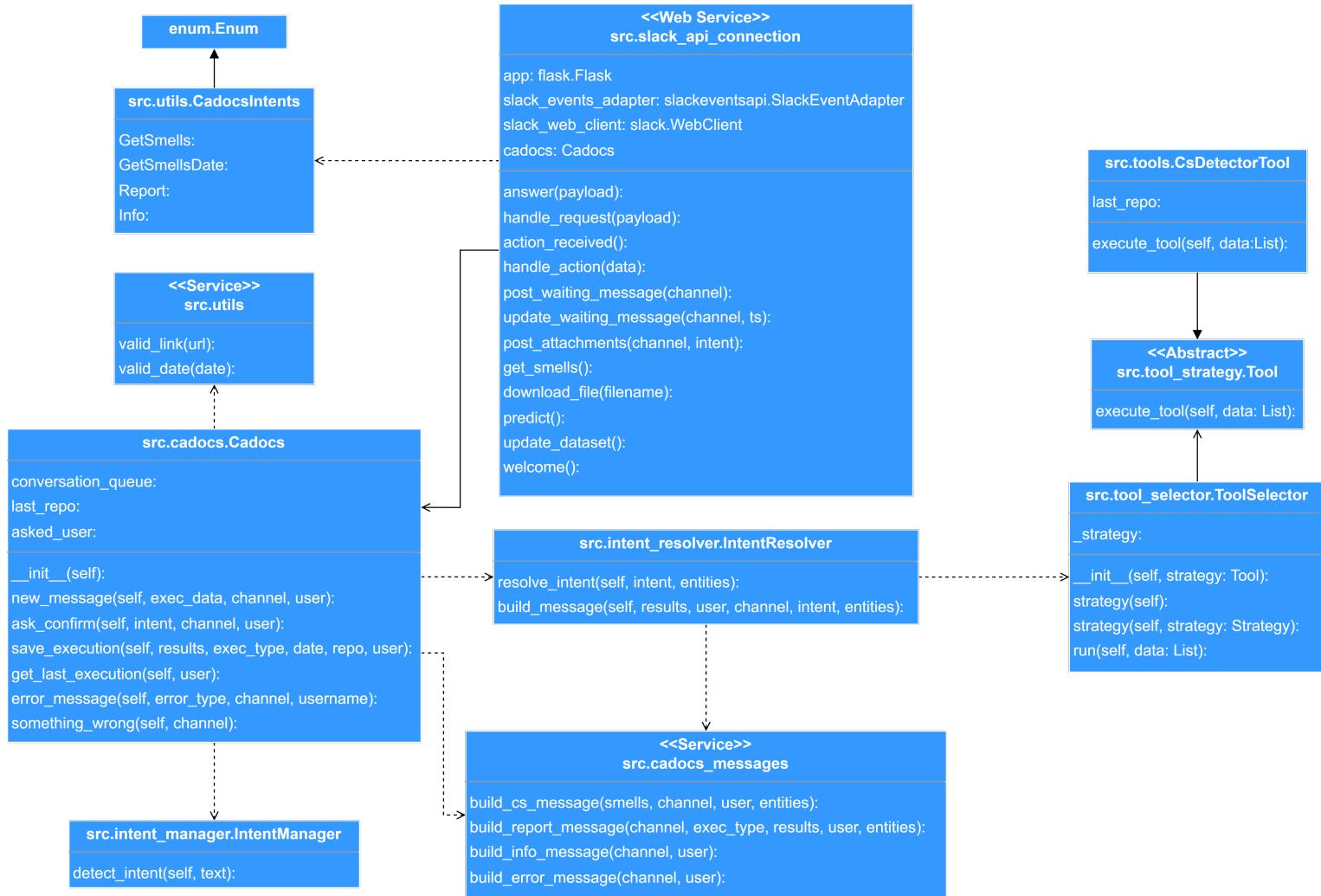
- *src.slack\_api\_connection* con stereotipo *Web Service*: questo modulo rappresenta l’interfacciamento con Slack, e la rappresentazione come un componente con stereotipo *Web Service* indica che esso non è una classe, ma fornisce un’interfaccia di comunicazione web con il servizio di messaggistica Slack.



**Figura 2.1:** Package Diagrams

- *src.slack\_api\_connection* ha una dipendenza con *src.utils.CadocsIntent*: questa dipendenza indica che nel web service viene utilizzato *CadocsIntent*, una sottoclasse della classe *enum.Enum* di Python, che viene utilizzata per rappresentare diversi intent come un'enumerazione.
- *src.slack\_api\_connection* ha un'associazione con *src.cadocs.Cadocs*: questa associazione indica che il modulo *src.slack\_api\_connection* usa la classe *src.cadocs.Cadocs*. La classe *Cadocs* rappresenta il core del chatbot e gestisce il flusso di informazioni tra Slack e le classi per la risoluzione degli intent degli utenti.
- *src.cadocs.Cadocs* ha una dipendenza diretta con *src.intent\_manager.IntentManager* e *src.intent\_resolver.IntentResolver*: queste dipendenze indicano che *Cadocs* utilizza le funzionalità fornite da queste due classi per gestire gli intent degli utenti che interagiscono con il chatbot. La classe *IntentManager* si occupa di interpretare le intenzioni degli utenti, mentre la classe *IntentResolver* determina come rispondere o agire in base a tali intenzioni.

- *src.cadocs.Cadocs* ha dipendenze con altri due moduli con stereotipo *Service*: *src.cadocs\_messages* e *src.utils*. Queste dipendenze indicano che *Cadocs* utilizza funzioni utili fornite dai due moduli. Questi moduli contengono funzioni di utilità per la validazione dei link e delle date, nonché per la costruzione dei messaggi da inviare a Slack per la visualizzazione da parte dell’utente.
- La classe *src.intent\_resolver.IntentResolver* ha una dipendenza con il *Service* *src.cadocs\_messages* e con la classe *src.tool\_selector.ToolSelector*: questa dipendenza indica che *IntentResolver* utilizza il modulo *cadocs\_messages* e si interfaccia con la classe *ToolSelector*. *ToolSelector* svolge un ruolo nella scelta degli strumenti da utilizzare per analizzare i repository.
- *src.tool\_selector.ToolSelector* ha un’associazione con la classe astratta *src.tool\_strategy.Tool*: questa associazione indica che *ToolSelector* interagisce con l’astrazione *Tool*, che rappresenta uno strumento generico all’interno del sistema. *ToolSelector* è responsabile di selezionare uno strumento specifico, come *src.tools.CsDetectorTool*, in base alle esigenze dell’analisi delle repository. Questa è un’implementazione di uno Strategy pattern.
- *src.tools.CsDetectorTool* è una classe che si interfaccia con *csDetector* per avviare l’analisi della repository: questa classe rappresenta lo strumento specifico che si collega al tool *csDetector* per effettuare l’analisi delle repository GitHub alla ricerca di community smells.

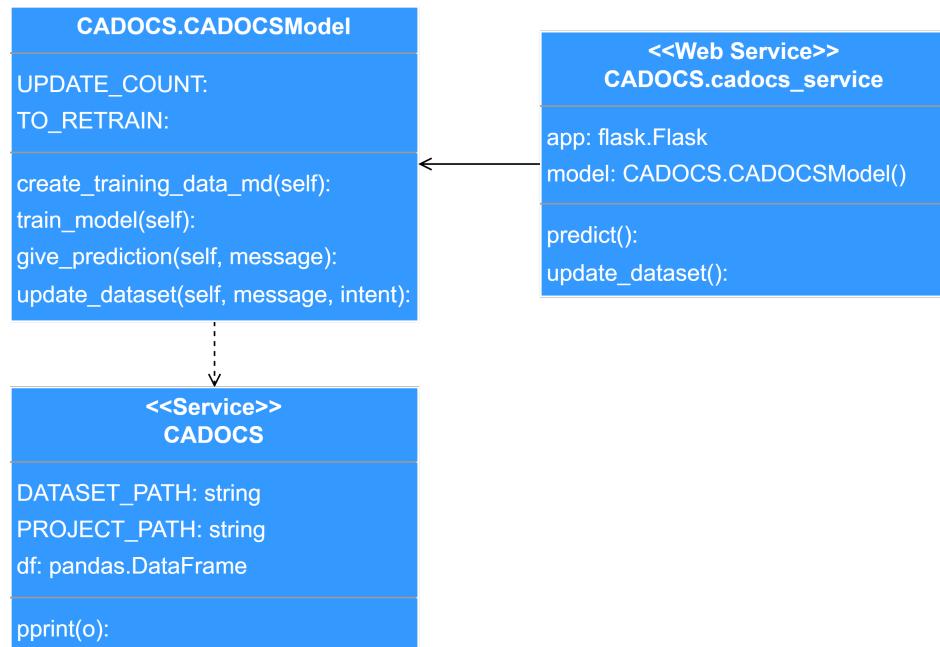


**Figura 2.2:** Class Diagrams di CADOCs

Il class diagramm estratto, e rappresentato in figura 2.3, fornisce una panoramica delle relazioni tra diverse classi e moduli nel progetto CADOCS\_NLU. Nel dettaglio:

- *CADOCS.cadocs\_service* con stereotipo *Web Service*: questo modulo permette di ricevere le richieste HTTP ed inoltrarle tramite il metodo *predict()* alla classe *CADOCSModel*, con cui ha un’associazione.
  - *CADOCS.CADOCSModel*: questa classe gestisce il modello per la classificazione delle richieste, per il re-training e per la gestione delle richieste da aggiungere al dataset, che viene così aggiornato per i futuri re-training. Questa classe ha una dipendenza con il modulo Service *CADOCS*.

- CADOCS con stereotipo *Service*: questo modulo contiene i path per localizzare il dataset ed il progetto; inoltre contiene l’oggetto DataFrame utilizzato da CADOCSModel per gli aggiornamenti del dataset.



**Figura 2.3:** Class Diagrams di CADOCS NLU

## 2.3 Use Case

Gli use case di seguito riportati sono riferiti alle funzionalità principali offerte dall’applicazione. In particolare, ci si concentra sul comportamento del modulo CADOCS, che interagisce con gli altri moduli.

### 2.3.1 Get Smells

Identificativo	UC_GS
Nome	GetSmells
Descrizione	<p>Lo UC fornisce la funzionalità di effettuare una richiesta di tipo "get smells" per avere informazioni sui community smells presenti in una repository Github</p>
Attore Principale	Utente
Attori Secondari	Modello NLU,
Entry Condition	<p>tool che identifica i community smells</p> <p>L'utente deve aver installato il tool su una piattaforma che lo supporta</p>
Exit Condition	<p>L'utente visualizza i community smells presenti nella repository fornita tramite link</p>
On Success	L'utente non visualizza i community smells presenti nella repository fornita tramite link
Exit Condition	
On Failure	
Rilevanza	Elevata
Frequenza Stimata	20/settimana
Flusso di eventi principale	
1	<p>Utente: L'utente formula una richiesta inserendo un link alla repository da analizzare</p>
2	<p>Sistema: Il sistema inoltra la richiesta al modello di NLU</p>
3	<p>Sistema: Il sistema riceve la risposta dal modello NLU circa l'intent della richiesta</p>
4	<p>Sistema: Il sistema estrae ed inoltra il link della repository al tool che analizza i community smells</p>
5	<p>Sistema: Il sistema mostra all'utente i risultati ottenuti</p>
Flusso di eventi di errore: Il modello NLU non riesce ad interpretare la richiesta dell'utente	
3.1	<p>Sistema: Il sistema mostra all'utente un messaggio di errore</p>
Flusso di eventi di errore: Il tool per la rilevazione di community smells non riesce a produrre risultati	
5.1	<p>Sistema: Il sistema mostra all'utente un messaggio di errore</p>

### 2.3.2 Get Smells Date

Identificativo	UC_GSD
Nome	GetSmellsDate
Descrizione	Lo UC fornisce la funzionalità di effettuare una richiesta di tipo "get smells date" per avere informazioni sui community smells presenti in una repository Github da una certa data in poi
Attore Principale	Utente
Attori Secondari	Modello NLU, tool che identifica i community smells
Entry Condition	L'utente deve aver installato il tool su una piattaforma che lo supporta
Exit Condition	L'utente visualizza i community smells presenti nella repository fornita tramite link
On Success	da una data in poi
Exit Condition	L'utente non visualizza i community smells presenti nella repository fornita tramite link
On Failure	da una data in poi
Rilevanza	Elevata
Frequenza Stimata	20/settimana
Flusso di eventi principale	
1	Utente: L'utente formula una richiesta inserendo un link alla repository da analizzare e la data da cui iniziare l'analisi
2	Sistema: Il sistema inoltra la richiesta al modello di NLU
3	Sistema: Il sistema riceve la risposta dal modello NLU circa l'intent della richiesta
4	Sistema: Il sistema estrae ed inoltra il link della repository al tool che analizza i community smells dalla data fornita
5	Sistema: Il sistema mostra all'utente i risultati ottenuti
Flusso di eventi di errore: Il modello NLU non riesce ad interpretare la richiesta dell'utente	
3.1	Sistema: Il sistema mostra all'utente un messaggio di errore
Flusso di eventi di errore: Il tool per la rilevazione di community smells non riesce a produrre risultati	
5.1	Sistema: Il sistema mostra all'utente un messaggio di errore

### 2.3.3 Info

Identificativo	UC_IN
Nome	Info
Descrizione	Lo UC fornisce la funzionalità di effettuare una richiesta di tipo "info" per avere informazioni sui community smells
Attore Principale	Utente
Attori Secondari	Modello NLU
Entry Condition	L'utente deve aver installato il tool su una piattaforma che lo supporta
Exit Condition	L'utente visualizza info community smells
On Success	L'utente non visualizza info community smells
Exit Condition	Elevata
On Failure	
Rilevanza	
Frequenza Stimata	10/settimana
Flusso di eventi principale	
1	Utente: L'utente formula una richiesta chiedendo informazioni sui community smells
2	Sistema: Il sistema inoltra la richiesta al modello di NLU
3	Sistema: Il sistema riceve la risposta dal modello NLU circa l'intent della richiesta
4	Sistema: Il sistema mostra all'utente le informazioni sui community smells
Flusso di eventi di errore: Il modello NLU non riesce ad interpretare la richiesta dell'utente	
3.1	Sistema: Il sistema mostra all'utente un messaggio di errore

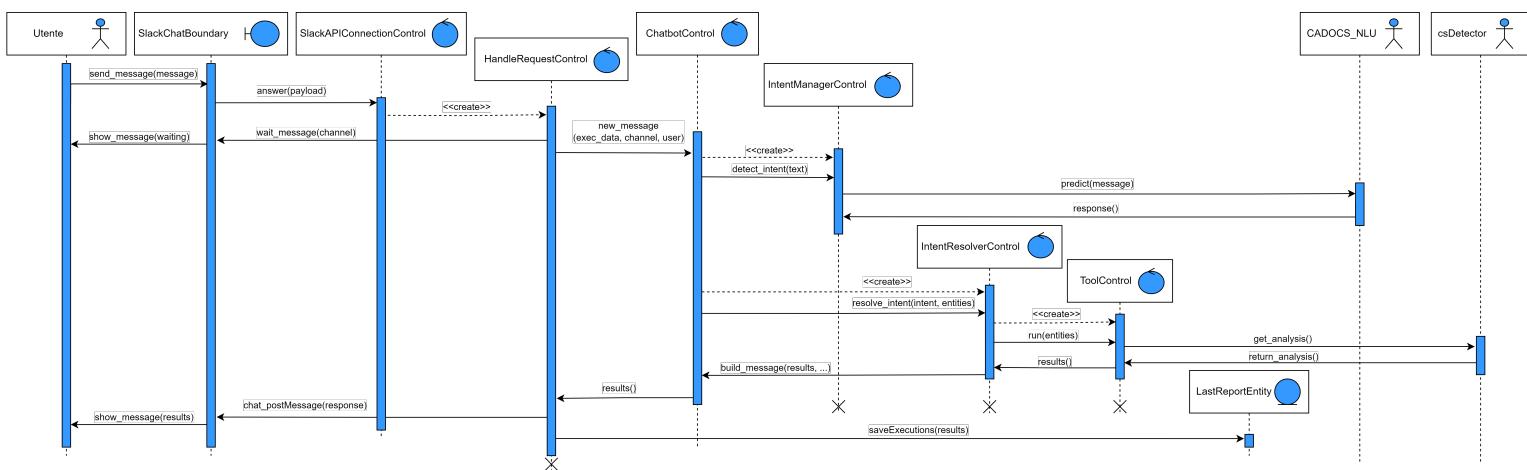
### 2.3.4 Report

Identificativo	UC_RE
Nome	Report
Descrizione	Lo UC fornisce la funzionalità di effettuare una richiesta di tipo "report" per avere l'ultimo report prodotto dal sistema
Attore Principale	Utente
Attori Secondari	Modello NLU
Entry Condition	L'utente deve aver installato il tool su una piattaforma che lo supporta
Exit Condition	L'utente visualizza l'ultimo report
On Success	
Exit Condition	L'utente non visualizza l'ultimo report
On Failure	
Rilevanza	Elevata
Frequenza Stimata	10/settimana
Flusso di eventi principale	
1	Utente: L'utente formula una richiesta chiedendo l'ultimo report prodotto
2	Sistema: Il sistema inoltra la richiesta al modello di NLU
3	Sistema: Il sistema riceve la risposta dal modello NLU circa l'intent della richiesta
4	Sistema: Il sistema mostra all'utente l'ultimo report prodotto
Flusso di eventi di errore: Il modello NLU non riesce ad interpretare la richiesta dell'utente	
3.1	Sistema: Il sistema mostra all'utente un messaggio di errore

## 2.4 Sequence Diagram

### 2.4.1 Get Smells e Get Smells Date

Il Sequence Diagram in Figura 2.4 descrive il comportamento dinamico del software in caso di una richiesta da parte dell’utente di tipo *get\_smells* oppure di tipo *get\_smells\_date*. In particolare sono stati riportati dettagli di più basso livello per poter avere una panoramica più dettagliata dei flussi di informazione all’interno dell’applicazione. Il Sequence Diagram si concentra quindi sul modulo CADOCS, che si interfaccia con il modulo CADOCS\_NLU e il tool csDetector come attori esterni. L’Entity *LastReportEntity* viene aggiornata per permettere in seguito di poter restituire l’ultimo report eseguito all’utente.



**Figura 2.4: SD\_GS\_GSD:** Sequence Diagram della richiesta Get Smells o Get Smells Date  
(Riferimento: UC\_GS e UC\_GSD)

### 2.4.2 Info

Il Sequence Diagram in Figura 2.5 si diversifica da quello di Figura 2.4 poiché in questo caso l’utente formula una richiesta di tipo *info*, per cui non è presente l’interfacciamento con CADOCS\_NLU e csDetector, e non viene salvata la risposta in *LastReportEntity*.

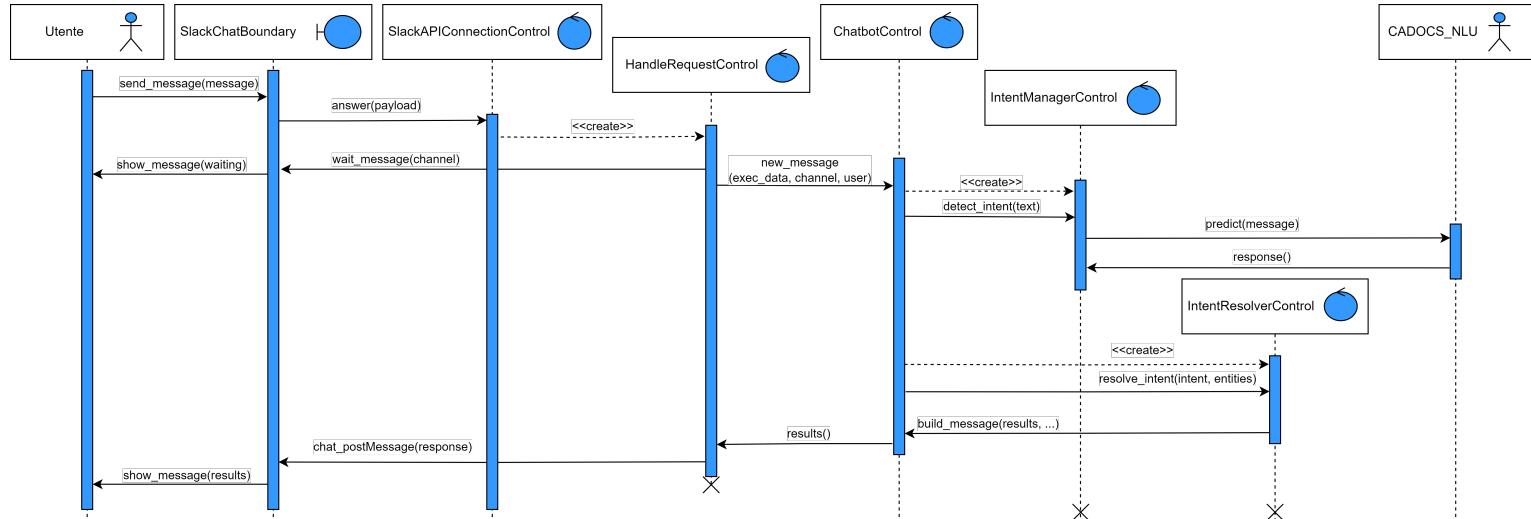


Figura 2.5: SD\_IN: Sequence Diagram della richiesta Info (Riferimento: UC\_IN)

### 2.4.3 Report

Il Sequence Diagram in Figura 2.6 è simile a quello in Figura 2.5, ma mostra invece il comportamento dinamico del software in risposta ad una richiesta di tipo *report*. In questo caso, viene effettuato un accesso all'Entity *LastReportEntity* per poter costruire la risposta da mostrare all'utente tramite *SlackChatBoundary*.

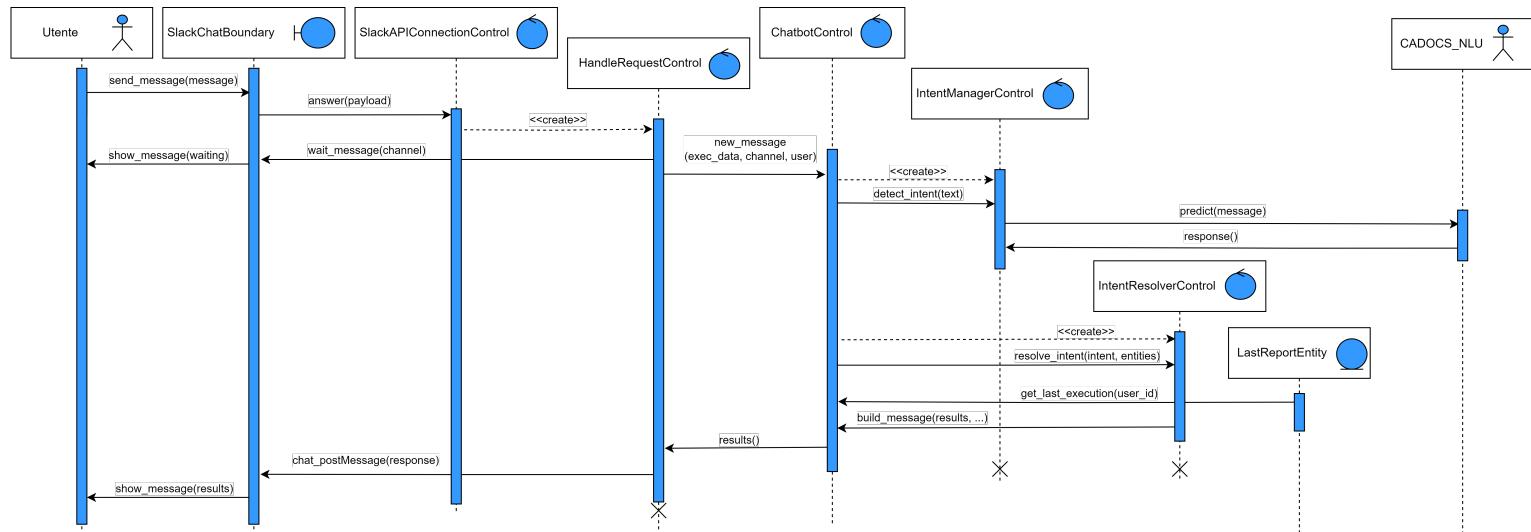
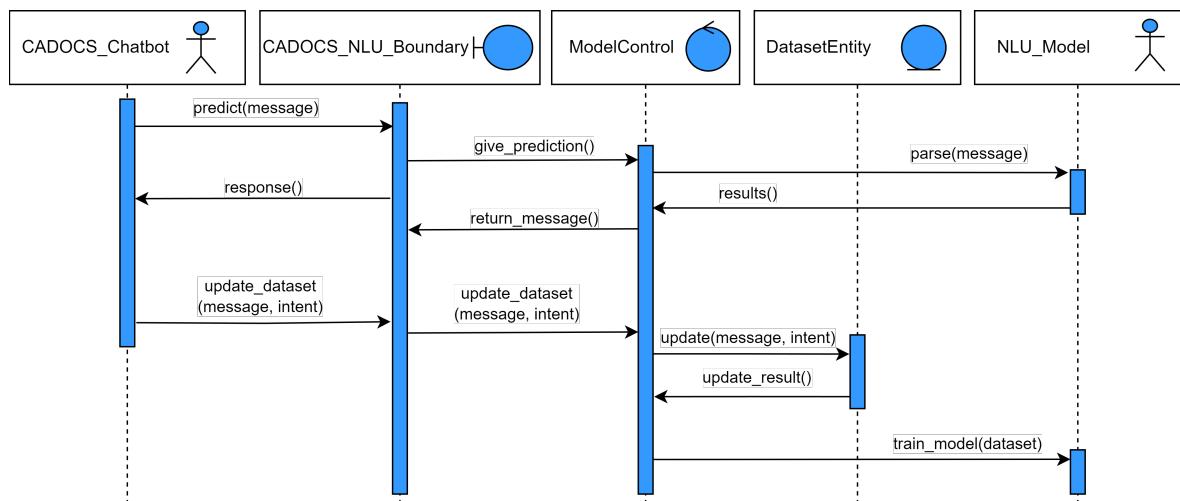


Figura 2.6: SD\_RE: Sequence Diagram della richiesta Report (Riferimento: UC\_RE)

## 2.4.4 Modello NLU

Il Sequence Diagram di Figura 2.7 è invece riferito al modulo CADOCS\_NLU, che riceve richieste HTTP dal modulo CADOCS, visto come un attore (*CADOCS\_ChatBot*). CADOCS\_NLU gestisce l’interazione con il modello NLU utilizzato per effettuare le predizioni. Questo è interpretato come un attore poiché memorizzato separatamente. Dopo aver ricevuto un messaggio di *update\_dataset*, il *ModelControl* procede con l’aggiornamento dell’Entity che rappresenta il dataset utilizzato per il training, noto come *DatasetEntity*. Il messaggio *update\_dataset* viene attivato nel momento in cui la confidence della previsione del modello è compresa tra 0.55 e 0.77 e l’intent predetto è confermato dall’utente come corretto. Ogni dieci aggiornamenti del dataset, viene effettuato il re-train del modello.



**Figura 2.7: SD\_NLU:** Sequence Diagram della richiesta di previsione al modello NLU