NaturalApi

Analisi dei requisiti

Versione	1.0.0
Approvazione	Simone Innocente
Redazione	Matteo Munari
	Giulio Umbrella
Verifica	Francesco Battistella
	Andrea Polo
Stato	Approvato
Uso	Esterno
Destinato a	FourCats
	TealBlue
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
Email	fourcats.unipd@gmail.com

Descrizione
Analisi dei requisiti del gruppo FourCats nella realizzazione del capitolato NaturalApi

Versione	Data	Nominativo	Ruolo	Descrizione	Verificatore	Esito
1.0.0	02-05-2020	Giulio Umbrella	Amministratore	Aggiornamento immagine UC5	Matteo Infantino	Positivo
0.9.0	01-04-2020	Matteo Munari	Analista	Ulteriori correzioni a use case e correzione numerazione scenari	Francesco Battistella	Positivo
	28-03-2020	Giulio Umbrella	Analista	Modificati UC13, UC23, UC27	Andrea Polo	Positivo
	27-03-2020	Matteo Munari	Analista	Aggiornato tracciamento dei requisiti	Andrea Polo	Positivo
	26-03-2020	Matteo Munari	Analista	Correzioni generali secondo suggerimenti RR	Andrea Polo	Positivo
0.0.2	06-03-2020	Matteo Munari	Analista	Modificati alcuni grafici UC	Francesco Battistella	Positivo
	10-02-2020	Giulio Umbrella	Analista	Aggiunti nuovi use case - UC11.1, UC11.2, UC11.3	Andrea Polo	Positivo
	08-02-2020	Matteo Munari	Analista	Aggiornata struttura use case	Andrea Polo	Positivo
	30-01-2020	Edoardo Tinto	Analista	Riorganizzazione e correzioni generali secondo suggerimenti RR	Andrea Polo	Positivo
0.0.1	04-01-2020	Giulio Umbrella	Analista	Correzione casi d'uso	Edoardo Tinto	Positivo
	02-01-2020	Matteo Munari	Analista	Correzione requisiti	Edoardo Tinto	Positivo
	01-01-2019	Matteo Munari	Analista	Inserita tabella tracciamento requisiti	Edoardo Tinto	Positivo
	31-12-2019	Matteo Munari	Analista	Conclusa stesura casi d'uso	Edoardo Tinto	Positivo
	30-12-2019	Matteo Munari	Analista	Conclusa stesura requisiti	Edoardo Tinto	Positivo

28-12-2019	Giulio Umbrella	Analista	Modificati requisiti	Edoardo Tinto	Positivo
21-12-2019	Giulio Umbrella	Analista	Modificati casi d'uso	Edoardo Tinto	Positivo
19-12-2019	Matteo Munari	Analista	Completata prima stesura casi d'uso	Edoardo Tinto	Positivo
17-12-2019	Giulio Umbrella	Analista	Modificati requisiti	Edoardo Tinto	Positivo
15-12-2019	Matteo Munari	Analista	Completata prima stesura requisiti	Andrea Polo	Positivo
13-12-2019	Matteo Munari	Analista	Iniziata stesura requisiti	Andrea Polo	Positivo
13-12-2019	Matteo Munari	Analista	Iniziata stesura casi d'uso	Andrea Polo	Positivo
12-12-2019	Matteo Munari	Analista	Conclusa stesura §3 - descrizione specifica	Andrea Polo	Positivo
12-12-2019	Matteo Munari	Analista	Conclusa stesura §2 - descrizione generale	Andrea Polo	Positivo
11-12-2019	Giulio Umbrella	Analista	Iniziata stesura §3 - descrizione specifica	Andrea Polo	Positivo
11-12-2019	Giulio Umbrella	Analista	Iniziata stesura §2 - descrizione genarale	Andrea Polo	Positivo
11-12-2019	Giulio Umbrella	Analista	Stesura introduzione	Andrea Polo	Positivo
11-12-2019	Giulio Umbrella	Amministratore	Creata struttura documento	Andrea Polo	Positivo

Indice

1 Introduzione

1.1 Scopo del documento

L'obiettivo di questo documento è illustrare i requisiti del capitolato C3 NaturalAPI per l'insegnamento di Ingegneria del software 19/20. Il documento prende come punto di partenza il capitolato fornito in data 15/11/2019 e il successivo approfondimento in data 29/11/2019.

1.2 Scopo del prodotto

1.2.1 Definizione del problema

Il contesto di applicazione principale di NaturalAPI è uno dei processi del ciclo di vita del software, la progettazione delle *API*. I responsabili devono scegliere i nomi e firme da assegnare ai metodi individuati in precedenza. L'identificazione di ciò che un software "fa" è il frutto della collaborazione tra *stakeholder* e avviene principalmente attraverso il linguaggio naturale. La stesura delle *API* è quindi analoga ad una traduzione; il linguaggio naturale deve essere tradotto in un linguaggio di programmazione. Il problema è che i responsabili delle *API* sono chiamati a scegliere i nomi in modo arbitrario. Questo processo presenta dei significativi difetti:

1. Eventuali ambiguità causate dal linguaggio naturale possono causare problemi;
2. La scelta dei singoli nomi è un processo lungo e noioso.

Quindi la stesura delle *API* rappresenta sia un collo di bottiglia che una possibile fonte di errori che verranno individuati solo dopo la codifica. Come introdotto nell'approfondimento in data 29/11/2019 ci sono delle soluzioni, ma richiedo un intenso lavoro manuale.

NB NaturalAPI può essere utilizzato anche nella fase di stesura dei requisiti come supporto per la scrittura degli user case. Vedi il successivo capitolo per maggiori dettagli.

1.2.2 Definizione della soluzione

NaturalAPI è un software di supporto per il processo di scrittura delle *API*. Possiamo usare un'analogia con la definizione di processo secondo lo standard ISO 9000, come illustrato in figura 1. Il bisogno che vogliamo soddisfare è la produzione di *API* . Il prodotto che soddisfa questo bisogno sono le *API* in un dato linguaggio di programmazione. L'user di NaturalAPI ha come obiettivo la produzione di *API* consistenti e chiare per tutti gli *stakeholder*. Per fare questo ha bisogno di confrontarsi con altri stakeholder. I vincoli passati a NaturalAPI sono i casi d'uso e l'analisi dei termini usati in una certa area di business. L'user può controllare i nomi di metodi suggeriti da NaturalAPI e proseguire con la definizione delle*API*. Le risorse consumate sono principalmente le ore lavoro impiegate. L'efficienza del processo può essere misurata dal numero di ore di lavoro per ottenere il prodotto finito. L'efficacia può essere misurata considerando il numero di errori e problemi causati dalla definizione sbagliata di un metodo e dal numero di ore di lavoro necessarie per effettuare una correzione.

Per richiesta del proponente, il software è modulato in tre diverse componenti, come illustrato in figura 2

1. **NaturalAPI Discover**: Analizza ed estrae i termini più utilizzati in una certa area di business;
2. **NaturalAPI Design**: Genera una pseudo-codifica ad alto livello delle *API*;
3. **NaturalAPI Develop**: Produce le *API* in un linguaggio di programmazione.

1.3 Flusso dell'applicazione

Il flusso dell'applicazione è il seguente:

1. Viene fornita documentazione rilevante a *NaturalAPI Discover* per estrarre un *Business Domain Language* (BDL) che contiene i termini più utilizzati all'interno del business;

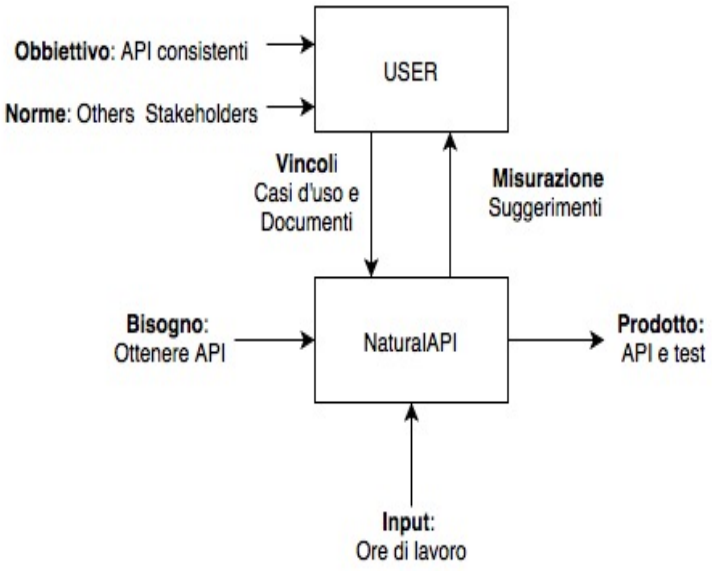

Figura 1: NaturalAPI come processo

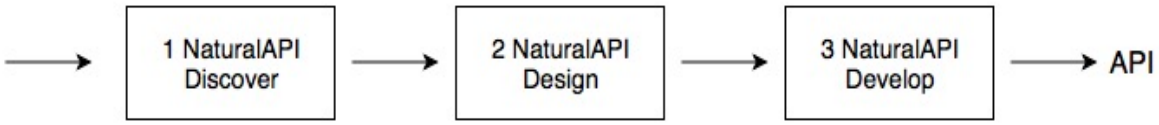

Figura 2: Workflow NaturalAPI generale

2. Il *BDL* viene passato a *NaturalAPI Design* assieme ai casi d'uso dell'applicazione per ottenere un *Business Application Language* (BAL);
3. Il *BAL* viene passato a *NaturalAPI Develop* che produce le *API* nel linguaggio di programmazione scelto.

1.4 Applicazioni pratiche e benefici

L'applicazione pratica più rilevante del software è la generazione automatiche delle *API* di un prodotto software in determinato linguaggio di programmazione. La procedura mira a raffinare progressivamente i termini utilizzati nella definizione delle funzioni implementate dal software. In questo modo:

1. I termini utilizzati hanno un significato consistente in tutto il progetto;
2. Eventuali ambiguità, dimenticanze e errori vengono individuate subito;
3. La produzione delle *API* diventa un processo standardizzato. Gli sviluppatori non devono occupare ore di lavoro a decidere come chiamare i metodi;
4. I programmatori possono concentrarsi sullo sviluppo di funzionalità e non sulla creazione di un modello concettuale di un business.

1.5 Glossario

All'interno del documento compaiono termini e sigle contrassegnati in *corsivo*. Per evitare di ripetere definizioni e spiegazioni, questi termini sono spiegati in maggior dettaglio nel Glossario.

1.6 Riferimenti

1.6.1 Normativi:

- Capitolato: https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C3.pdf

1.6.2 Informativi:

- Gherkin: https://cucumber.io/docs/gherkin/reference/
- Standard IEEE Std 830-1998: https://ieeexplore.ieee.org/document/720574
- Stanford Parser: https://nlp.stanford.edu/software/lex-parser.shtml
- Stanford Dependencies: https://nlp.stanford.edu/software/stanford-dependencies.shtml
- Swagger OpenAPI specification: https://swagger.io/specification/

1.7 Struttura del documento

Il presente documento procede per raffinamenti, entrando progressivamente nel dettaglio del comportamento dei tre moduli. La sezione 2 identifica gli aspetti fondamentali del problema. La sezione 3 entra nei dettaglio del funzionamento del sistema e presenta use case e elenco completo dei requisiti. La sezione 4 contiene i risultati del processo di verifica dei requisiti.

2 Descrizione generale

2.1 Prospettive sul prodotto

NaturalAPI è un tool di supporto. Per poter essere utilizzato bisogna prima stabilire:

- Il linguaggio naturale per i documenti e i casi d'uso (eg US english)
- Il *Business Domain* di riferimento.
- I bisogni che il software è chiamato a soddisfare.
- Il gruppo di *Stakeholder* che vuole sviluppare il software.

2.2 Funzioni del prodotto

La figura 3 mostra il funzionamento complessivo del software. Le funzioni di NaturalAPI possono essere illustrate sulla base dei suoi moduli principali, Discover, Design e Develop.

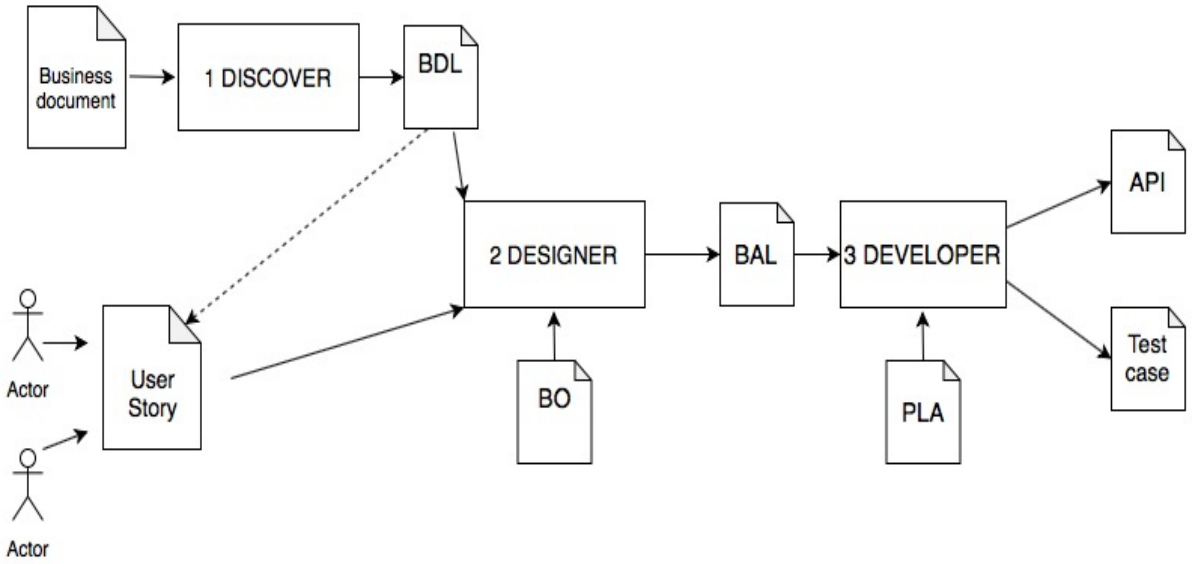

Figura 3: Workflow NaturalAPI Intermedio

2.2.1 NaturalAPI Discover

Lo scopo di *NaturalAPI Discover* è trovare i termini più utilizzati in un certo business e il modo in cui si relazionano. Il primo passaggio consiste nel raccolta di documenti relativi al business. Le fonti suggerite dal proponente possono essere: articoli di wiki-how, manuale, wiki ecc. Per documenti in lingua diversa dall'inglese servirà una traduzione. Il risultato ottenuto è un *Business Domain Language* che contiene i termini più utilizzati nel *BD*. Il *BDL* può anche essere utilizzato durante la stesura dei requisiti come punto di partenza nella scelta dei termini.

2.2.2 NaturalAPI Design

Il compito di *NaturalAPI Design* è trasformare la descrizione dei requisiti scritta in linguaggio naturale in uno pseudo linguaggio di alto livello, più simile alle *API* ma ancora comprensibile da un utente non sviluppatore. Per farlo utilizza il *BDL* prodotto da *NaturalAPI Discover*. *NaturalAPI Design* confronta i termini utilizzati nella descrizione dei requisiti con quelli del *BDL* e fornisce suggerimenti per utilizzare combinazioni di termini più appropriate al *BD*. Dopo che suggerimenti sono controllati dall'utente, viene prodotto un *Business Application Language*

2.2.3 NaturalAPI Develop

NaturalAPI Develop ha il compito di tradurre il *BAL* in *API* di un linguaggio di programmazione scelto e di produrre i test di integrazione per gli scenari di riferimento.

2.3 Caratteristiche utente

Il software nasce con lo scopo di essere adoperato a vantaggio degli sviluppatori di *API*, ma, in particolare nella fase iniziale, questo deve essere pensato per essere utilizzato oltre che da personale con competenze informatiche, anche da altri stakeholder non sviluppatori. Questi possono essere, ad esempio, product manager o esperti del dominio. Il contributo di questi *stakeholder* è determinante nel primo modulo, *NaturalAPI Discover*. La figura 4 mostra il ruolo degli stakeholder in ciascuna fase di NaturalAPI. In questo contesto misuriamo le fasi usando come milestone gli output prodotti: le *feature*, il *BDL*, il *BAL* e le *API* e test.

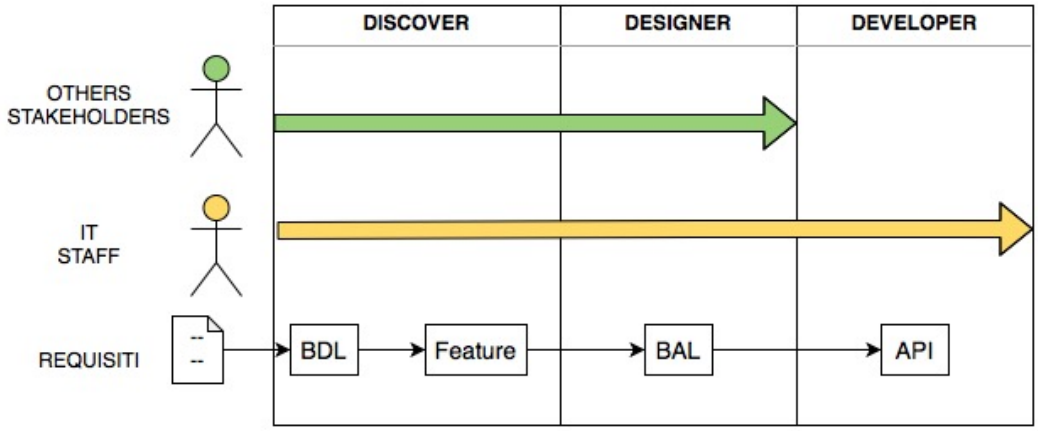

Figura 4: Users per ciascuna fase NaturalAPI

2.3.1 User NaturalAPI Discover

NaturalAPI Discover deve dare la possibilità di utilizzare i propri strumenti ad un'ampia cerchia di persone, la più vasta possibile, al fine di essere utilizzato assieme agli altri *stakeholder* e di aiutare il personale IT a raccogliere dati nella maniera più efficiente e efficace possibile. *NaturalAPI Discover* viene usato dagli *stakeholder* per produrre il *BDL*. Una volta prodotto il *BDL*, gli *stakeholder* possono utilizzare *NaturalAPI Discover* come mezzo automatico di perfezionamento degli *scenari BDD*. Le *feature* possono essere prodotte anche senza l'ausilio del *BDL*, ma il suo uso permetterebbe fin dall'inizio di utilizzare i termini nel modo più corretto possibile. Come si vede in figura 4, tutti gli *stakeholder* hanno un ruolo attivo in questa fase. Utilizzando *NaturalAPI Discover* si evidenzia una possibile fonte di ambiguità nei termini usati e lo *stakeholder* può modificare la terminologia degli *scenari* per renderli più chiari.

2.3.2 User NaturalAPI Design

L'utente principale di *NaturalAPI Design* è un qualsiasi sviluppatore, il quale ha il ruolo di approvare o modificare il risultato prodotto da *NaturalAPI Design*, cioè la conversione degli scenari in pseudolinguaggio (*BAL*). Lo sviluppatore deve quindi verificare che lo pseudolinguaggio prodotto sia corretto e facilmente comprensibile. A questo scopo, *NaturalAPI Design* deve andare incontro alle abitudini e le preferenze di queste tipologie di utenti, supportando l'esecuzione su qualsiasi tipologia di OS popolare presso gli sviluppatori, in particolare Ubuntu Linux. In questa fase, gli altri *stakeholder* possono avere ancora un funzione rilevante nella definizione delle *API*. Eventuali dubbi o ambiguità possono essere affrontati usando gli strumenti messi a disposizione da *NaturalAPI Design* per manipolare i suggerimenti ottenuti. Benché questa fase sia condotta da uno sviluppatore, ogni suggerimento di codice è affiancato alla suo corrispettiva feature e quindi anche gli altri *stakeholder* possono partecipare attivamente.

2.3.3 User NaturalAPI Develop

Gli strumenti messi a disposizione in *NaturalAPI Develop* si intendono riservati prettamente a sviluppatori, viste le conoscenze necessarie richieste per la strutturazione finale delle *API*. L'utente di *NaturalAPI Develop* deve verificare che la traduzione da pseudolinguaggio a linguaggio specifico sia corretta, apportando modifiche quando serve e, se necessario, fornendo regole aggiuntive per migliorare la traduzione e rispettare le convenzioni del linguaggio. Il prodotto finale di questa fase è essenzialmente del codice per un'applicazione, in questo senso il ruolo degli altri *stakeholder* è minimo o nullo.

3 Descrizione specifica

Questa sezione entra nel dettaglio di tutte le specifiche del software e delle tecnologie utilizzate.

3.1 Prospettive sul prodotto

La figura 5 mostra una panoramica completa del software.

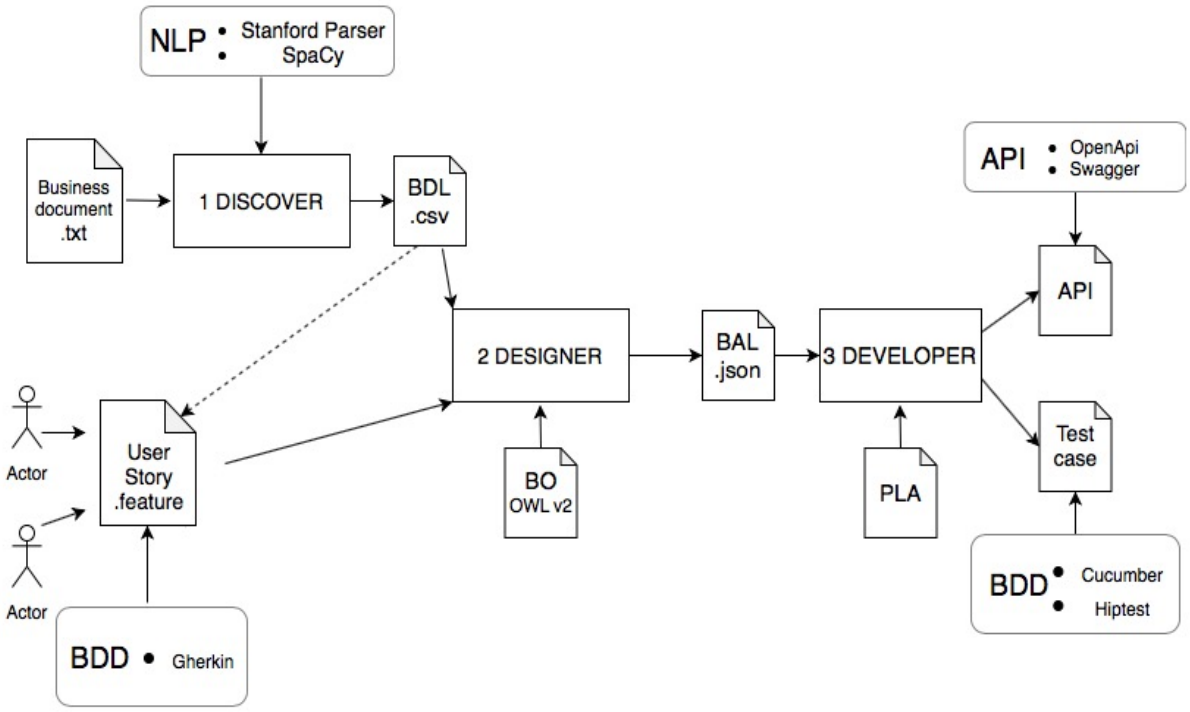

Figura 5: Workflow NaturalAPI Completo

3.2 Funzioni del prodotto

Le funzionalià delle tre componenti di NaturalAPI sono spiegate in dettaglio:

3.2.1 NaturalAPI Discover

NaturalAPI Discover utilizza tecniche di *Natural Language Processing* (NLP) per analizzare i documenti di testo forniti in input ed estrarre nomi, verbi e predicati e la frequenza con cui compaiono. I file di input in formato .txt sono relativi a un dato dominio e possono provenire da fonti diverse: articoli scientifici, wiki-how, quotidiani, manuali, pagine web ecc. L'analisi produce tre file .bdl, rispettivamente per nomi, verbi e predicati, che insieme formeranno un *Business Domain Language*. *NaturalAPI Discover* mette a disposizione anche una funzionalità di analisi di feature e scenari *BDD* per suggerire delle sostituzioni dei termini inerenti al *Business Domain*, con quelli contenuti nei file .bdl. In questo modo gli scenari risulteranno coerenti con il *BDL* e maggiormente comprensibili. *NaturalAPI Discover* richiede il seguente input e produce l'output indicato:

Input:

- documenti relativi al dominio
  - formato: plain text;

  - pattern nome file: [documentName].[language].txt

Output:

- lista dei verbi lemmatizzati e frequenze
  - formato: CSV;
  - pattern nome file: [projectName].verbs.[language].bdl
- lista dei nomi lemmatizzati e frequenze
  - formato: CSV;
  - pattern nome file: [projectName].names.[language].bdl
- lista dei predicati lemmatizzati e frequenze
  - formato: CSV;
  - pattern nome file: [projectName].predicates.[language].bdl

3.2.2 NaturalAPI Design

NaturalAPI Design utilizza il *BDL* ottenuto da *NaturalAPI Discover* e ontologie business specifiche per convertire gli *scenari* in formato *Gherkin* in un prototipo di *Business Application Language*, uno pseudolinguaggio che mette in evidenza le varie funzioni richieste dallo scenario e le entità coinvolte. Il prototipo di *BAL* dovrà poi essere approvato dall'utente prima di poter generare il file .bal. L'utente può apportare modifiche al prototipo per riorganizzare le funzioni e rendere il *BAL* più chiaro. Se invece l'utente non è soddisfatto del prototipo, *NaturalAPI Design* fornisce altri suggerimenti. *NaturalAPI Design* richiede il seguente input e produce l'output indicato:

Input:

- scenari e feature
  - formato: gherkin
  - pattern nome file: [featureName].feature
- lista dei verbi lemmatizzati e frequenze
- lista dei nomi lemmatizzati e frequenze
- lista dei predicati lemmatizzati e frequenze
- ontologie business
  - formato: Web Ontology Language (OWL) v2

Output:

- business application language
  - formato: OpenAPI v3 JSON Object
  - pattern nome file: [projectName].[operationId].bal

3.2.3 NaturalAPI Develop

NaturalAPI Develop utilizza il *BAL* per produrre le *API* in uno specifico linguaggio di programmazione. Per effettuare la traduzione *NaturalAPI Develop* necessita di un *PLA* per il linguaggio di destinazione, il quale vien fornito dall'utente. Come per *NaturalAPI Design* la produzione è interattiva e vengono forniti dei suggerimenti di *API* che dovranno essere valutati e accettati dall'utente prima di generare il file delle *API*. *NaturalAPI Develop* permette anche di modificare o aggiungere metodi alle *API* qualora l'utente lo reputi necessario. *NaturalAPI Develop* inoltre produce i test di integrazione per gli *scenari* utilizzati per la produzione del *BAL*. I test potranno essere usati per testare le *API* prodotte e sono relativi a un framework scelto dall'utente. *NaturalAPI Develop* richiede il seguente input e produce l'output indicato:

Input:

- business application language
- programming language adapter

Output:

- API relative al linguaggio scelto
- test di integrazione per il framework scelto

3.3 Casi d'uso

3.3.1 Attori

- **Sviluppatore**: utilizzatore principale di NaturalAPI, inteso come designer/sviluppatore di API;
- **Stakeholder**: product manager ed esperti di dominio;

3.3.2 UC1 - Estrazione di un BDL dai documenti

- **Scopo e descrizione:** Lo sviluppatore richiede la produzione di un BDL.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore seleziona uno o più documenti di testo da quelli che ha a disposizione. I documenti selezionati sono relativi ad uno stesso argomento.
- **Post-condizioni:** L'utente ottiene un Business Domain Language (BDL) contenente un elenco dei nomi, verbi e predicati contenuti nei documenti ricevuti in input e la relativa frequenza. Viene memorizzata l'associazione tra il BDL e i documenti utilizzati per produrlo.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la produzione del BDL.

3.3.3 UC2 - Aggiunta nuovi documenti a un BDL

- **Scopo e descrizione:** Lo sviluppatore richiede l'integrazione di un BDL esistente con altri documenti.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione dei BDL in formato .bdl e dei documenti in formato .txt relativi al BDL. Lo sviluppatore ha selezionato uno o più documenti di testo e il BDL da integrare.
- **Post-condizioni:** Il BDL viene integrato solamente con i nuovi documenti che non sono stati analizzati in precedenza. L'associazione tra BDL e documenti viene aggiornata.
- **Scenario Principale:**
  1. Lo sviluppatore avvia l'integrazione.

3.3.4 UC3 - Rimozione analisi di un documento da un BDL

- **Scopo e descrizione:** Lo sviluppatore richiede la rimozione di un documento da un BDL.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un BDL e un documento dall'elenco dei documenti utilizzati per produrlo.
- **Post-condizioni:** Il BDL viene aggiornato rimuovendo l'analisi del documento scelto. La lista dei documenti memorizzati viene aggiornata rimuovendo il documento scelto.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la rimozione;

3.3.5 UC4 - Creazione scenario BDD

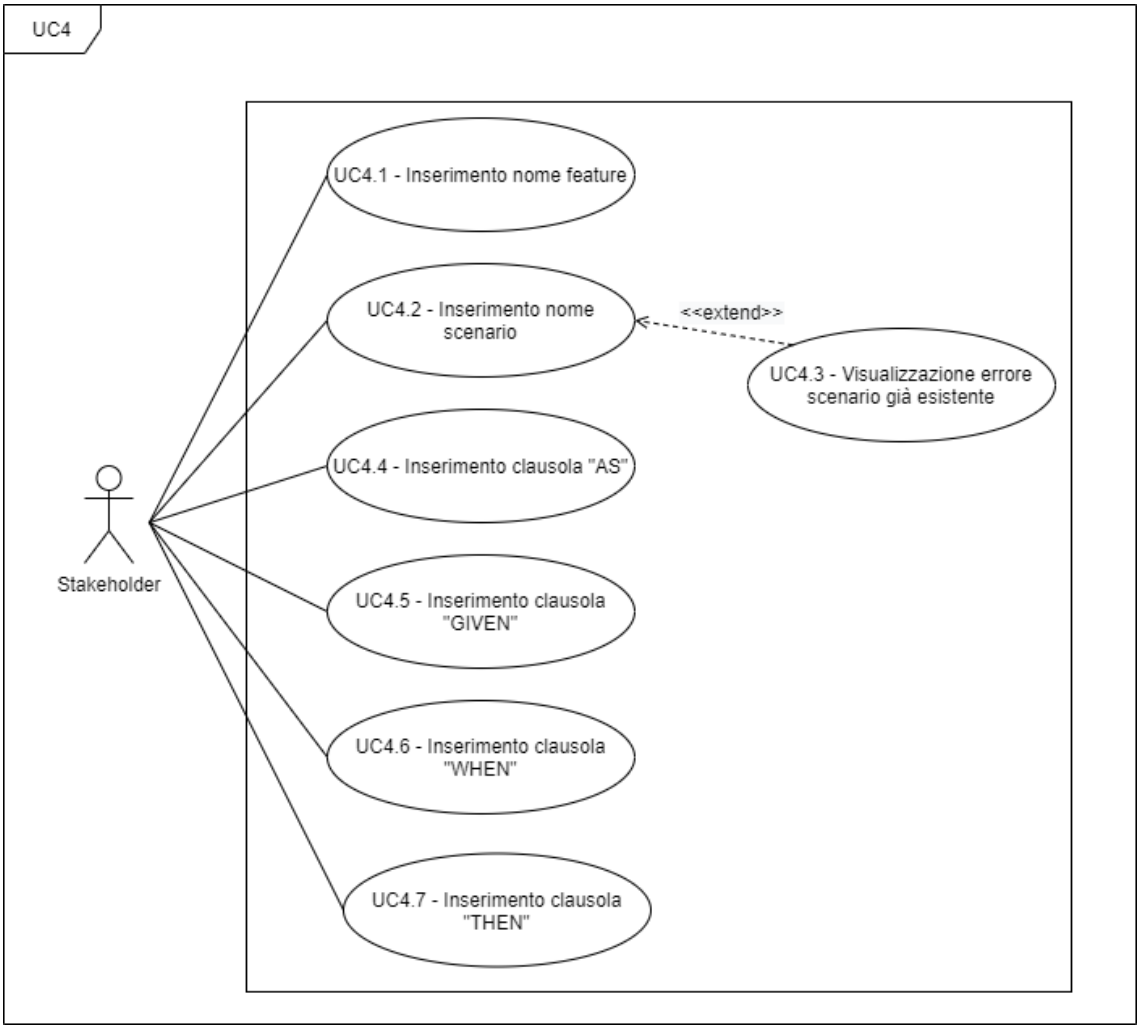

Figura 6: UC4 - Creazione scenario BDD

- **Scopo e descrizione:** Lo stakeholder crea un nuovo scenario BDD.
- **Attore primario:** Stakeholder.

- **Pre-condizioni:** Lo stakeholder ha identificato uno scenario da formalizzare.
- **Post-condizioni:** Viene creato un nuovo scenario BDD.
- **Scenario Principale:**
  1. Lo stakeholder inserisce la feature di riferimento [UC4.1];
  2. Lo stakeholder inserisce il nome dello scenario [UC4.2];
  3. Lo stakeholder può inserire o meno il testo per la clausola "AS" [UC4.4];
  4. Lo stakeholder inserisce il testo per la clausola "GIVEN" [UC4.5];
  5. Lo stakeholder inserisce il testo per la clausola "WHEN" [UC4.6];
  6. Lo stakeholder inserisce il testo per la clausola "THEN" [UC4.7].

3.3.6 UC4.1 - Inserimento nome feature

- **Scopo e descrizione:** Lo stakeholder inserisce il nome della feature.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder sta creando un nuovo scenario.
- **Post-condizioni:** NaturalAPI registra il nome della feature.
- **Scenario Principale:**
  1. Lo stakeholder inserisce il nome della feature di riferimento.

3.3.7 UC4.2 - Inserimento nome scenario

- **Scopo e descrizione:** Lo stakeholder inserisce il nome dello scenario.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder sta creando un nuovo scenario.
- **Post-condizioni:** NaturalAPI registra il nome del nuovo scenario.
- **Scenario Principale:**
  1. Lo stakeholder inserisce il nome del nuovo scenario.
- **Estensioni:**
  a) Se esiste già uno scenario con lo stesso nome viene visualizzato un messaggio d'errore [UC4.3].

3.3.8 UC4.3 - Visualizzazione errore scenario già esistente

- **Scopo e descrizione:** Lo stakeholder viene avvisato che non è possibile creare uno scenario con quel nome.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder richiede la creazione di uno scenario con un nome già utilizzato.
- **Post-condizioni:** NaturalAPI avvisa lo stakeholder che esiste già uno scenario con lo stesso nome.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che esiste già uno scenario con lo stesso nome.

3.3.9 UC4.4 - Inserimento clausola "AS"

- **Scopo e descrizione:** Lo stakeholder inserisce l'attore dello scenario.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder sta creando un nuovo scenario.
- **Post-condizioni:** NaturalAPI registra il contenuto della clausola "AS" del nuovo scenario.
- **Scenario Principale:**
  1. Lo stakeholder inserisce il contenuto della clausola "AS".

3.3.10 UC4.5 - Inserimento clausola "GIVEN"

- **Scopo e descrizione:** Lo stakeholder inserisce le pre-condizioni dello scenario.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder sta creando un nuovo scenario.
- **Post-condizioni:** NaturalAPI registra il contenuto della clausola "GIVEN" del nuovo scenario.
- **Scenario Principale:**
  1. Lo stakeholder inserisce il contenuto della clausola "GIVEN".

3.3.11 UC4.6 - Inserimento clausola "WHEN"

- **Scopo e descrizione:** Lo stakeholder inserisce la condizione di attivazione dello scenario.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder sta creando un nuovo scenario.
- **Post-condizioni:** NaturalAPI registra il contenuto della clausola "WHEN" del nuovo scenario.
- **Scenario Principale:**
  1. Lo stakeholder inserisce il contenuto della clausola "WHEN".

3.3.12 UC4.7 - Inserimento clausola "THEN"

- **Scopo e descrizione:** Lo stakeholder inserisce le post-condizioni dello scenario.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder sta creando un nuovo scenario.
- **Post-condizioni:** NaturalAPI registra il contenuto della clausola "THEN" del nuovo scenario.
- **Scenario Principale:**
  1. Lo stakeholder inserisce il contenuto della clausola "THEN".

3.3.13 UC5 - Modifica guidata scenario BDD

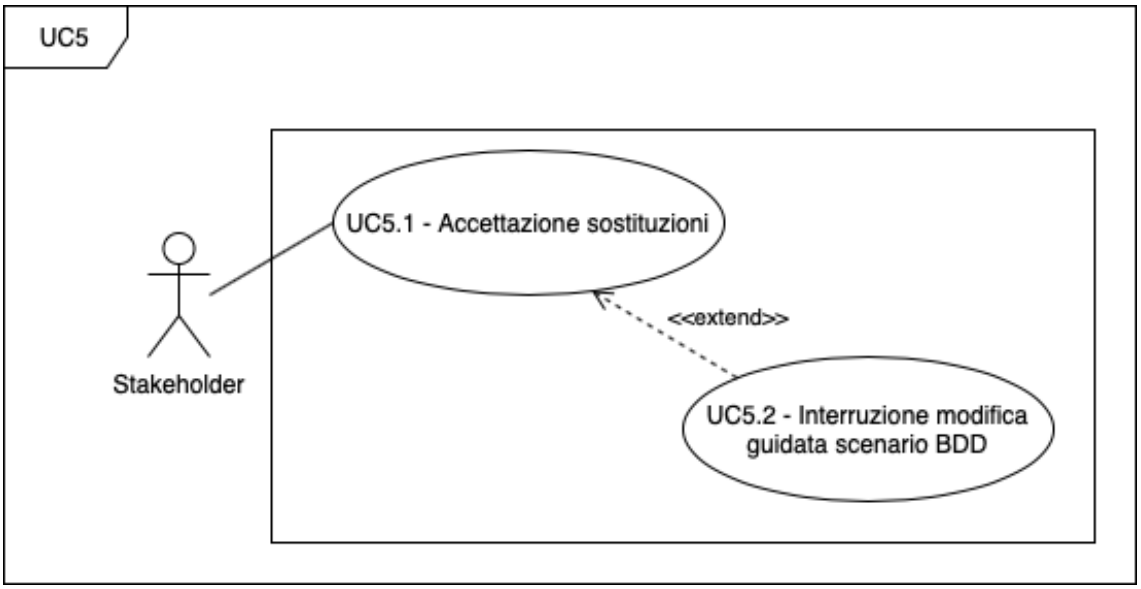

Figura 7: UC5 - Modifica guidata scenario BDD

- **Scopo e descrizione:** Lo stakeholder modifica uno scenario BDD con i termini presenti nel BDL.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder ha a disposizione degli scenari BDD e dei BDL. Lo stakeholder ha selezionato uno scenario e un BDL.
- **Post-condizioni:** Lo scenario viene modificato.
- **Scenario Principale:**
  1. Lo stakeholder avvia la modifica guidata;
  2. NaturalAPI Discover suggerisce delle sostituzioni per i termini con dei sinonimi più corretti;
  3. Lo stakeholder accetta le sostituzioni [UC5.1].

3.3.14 UC5.1 - Accettazione sostituzioni

- **Scopo e descrizione:** Lo stakeholder accetta le sostituzioni proposte.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder ha avviato la modifica guidata di uno scenario.
- **Post-condizioni:** Le sostituzioni proposte sono state approvate.
- **Scenario Principale:**
  1. Lo stakeholder accetta le sostituzioni proposte.
- **Estensioni:**
  a) Se lo stakeholder non accetta le sostituzioni la modifica viene interrotta [UC5.2].

3.3.15 UC5.2 - Interruzione modifica guidata scenario BDD

- **Scopo e descrizione:** La modifica di uno scenario viene annullata.
- **Attore primario:** Stakeholder.
- **Pre-condizioni:** Lo stakeholder non accetta le sostituzioni fornite dal sistema.
- **Post-condizioni:** Le modifiche apportate allo scenario vengono annullate.
- **Scenario Principale:**
  1. La modifica viene interrotta;
  2. I cambiamenti non vengono memorizzati.

3.3.16 UC6 - Generazione suggerimenti BAL

- **Scopo e descrizione:** L'utente richiede la produzione di un BAL.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione un BDL, degli scenari in formato Gherkin coerenti tra loro. Lo sviluppatore ha selezionato uno o più scenari Gherkin e un BDL.
- **Post-condizioni:** I suggerimenti di BAL vengono generati.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la produzione dei suggerimenti.

3.3.17 UC7 - Accettazione suggerimento BAL

- **Scopo e descrizione:** Lo sviluppatore accetta un suggerimento del BAL.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un suggerimento di un BAL.
- **Post-condizioni:** Il suggerimento scelto viene contrassegnato come accettato.
- **Scenario Principale:**
  1. Lo sviluppatore accetta il suggerimento selezionato.

3.3.18 UC8 - Rifiuto suggerimento BAL

- **Scopo e descrizione:** Lo sviluppatore rifiuta un suggerimento del BAL.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un suggerimento di un BAL.
- **Post-condizioni:** Il suggerimento scelto viene contrassegnato come rifiutato.
- **Scenario Principale:**
  1. Lo sviluppatore rifiuta il suggerimento selezionato.

3.3.19 UC9 - Modifica di una action

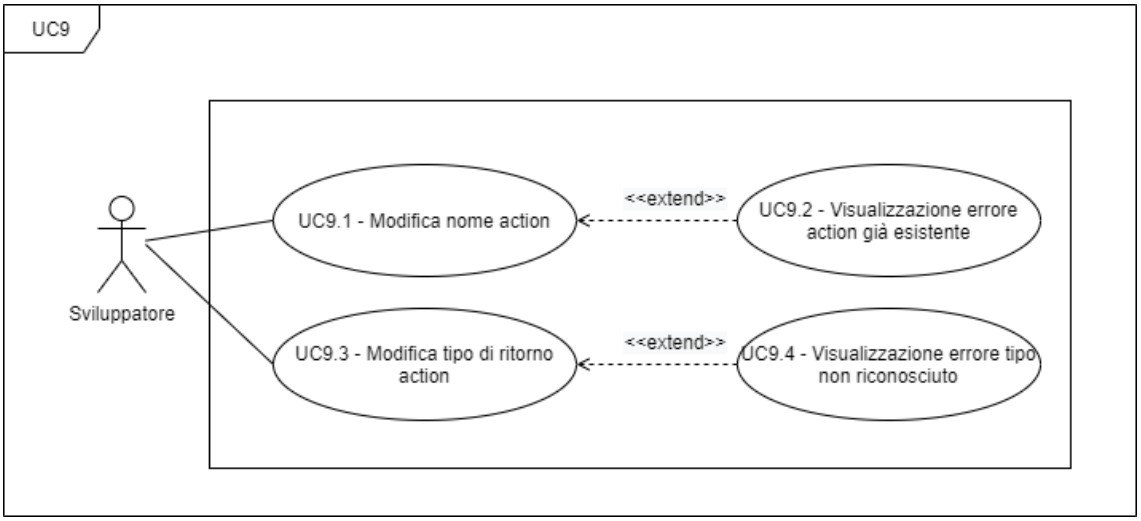

Figura 8: UC9 - Modifica di una action

- **Scopo e descrizione:** Lo sviluppatore modifica il nome di una action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha abilitato la modifica di una action.
- **Post-condizioni:** L'action viene modificata.
- **Scenario Principale:**
  1. Lo sviluppatore modifica il nome della action [UC9.1];
  2. Lo sviluppatore modifica il tipo di ritorno della action [UC9.3].

3.3.20 UC9.1 - Modifica nome action

- **Scopo e descrizione:** Lo sviluppatore modifica il nome di una action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando una action.
- **Post-condizioni:** Il nuovo nome inserito viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nuovo nome.
- **Estensioni:**
  a) Se esiste già una action con lo stesso nome viene visualizzato un messaggio d'errore [UC9.2].

3.3.21 UC9.2 - Visualizzazione errore action già esistente

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile utilizzare il nome scelto per la action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito il nome di una action già esistente.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che esiste già una action con quel nome.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che esiste già una action con lo stesso nome.

3.3.22 UC9.3 - Modifica tipo di ritorno action

- **Scopo e descrizione:** Lo sviluppatore modifica il tipo di ritorno di una action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando una action.
- **Post-condizioni:** Il tipo di ritorno inserito viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nuovo tipo di ritorno.
- **Estensioni:**

  a) Se il tipo inserito non è riconosciuto da NaturalAPI viene visualizzato un messaggio d'errore [UC9.4].

3.3.23 UC9.4 - Visualizzazione errore tipo non riconosciuto

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che il tipo inserito non è riconosciuto dal sistema.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito un tipo non riconosciuto da NaturalAPI.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che il tipo inserito non è valido.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che il tipo inserito non è valido.

3.3.24 UC10 - Creazione tipo custom

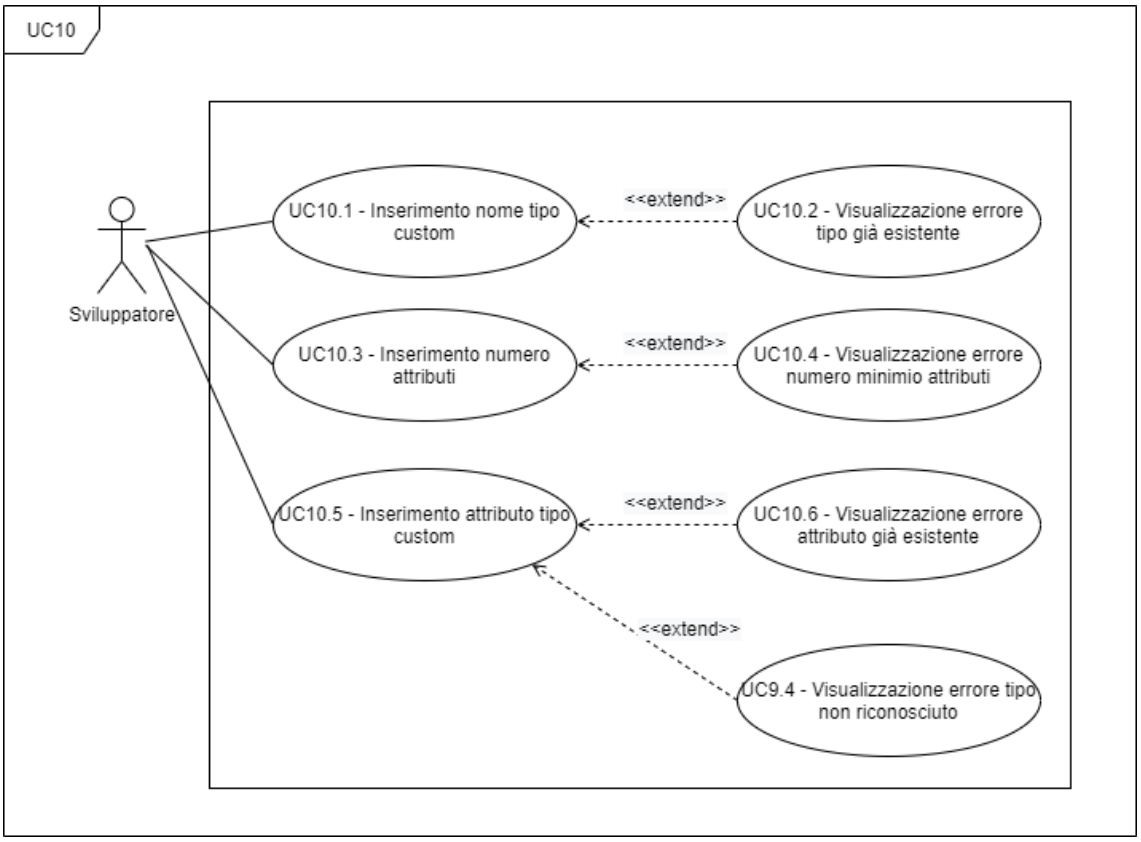

Figura 9: UC10 - Creazione tipo custom

- **Scopo e descrizione:** Lo sviluppatore inserisce un tipo customizzato.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Il tipo inserito in precedenza non è riconosciuto da NaturalAPI.
- **Post-condizioni:** Il nuovo tipo custom viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce un nome per il tipo custom [UC10.1];
  2. Lo sviluppatore inserisce il numero di attributi [UC10.3]
  3. Lo sviluppatore inserisce dei parametri[UC10.5];
  4. Lo sviluppatore conferma la creazione.

3.3.25 UC10.1 - Inserimento nome tipo custom

- **Scopo e descrizione:** Lo sviluppatore inserisce un nome per il tipo custom.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un tipo custom.
- **Post-condizioni:** Il nome inserito viene registrato.

- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nome.
- **Estensioni:**

  a) Se il nome inserito corrisponde a un tipo riconosciuto da NaturalAPI viene viualizzato un errore [UC10.2].

3.3.26 UC10.2 - Visualizzazione errore tipo già esistente

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile utilizzare il nome scelto per il tipo custom.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito il nome di un tipo già esistente.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che esiste già un tipo con quel nome.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che esiste già un tipo con il nome inserito.

3.3.27 UC10.3 - Inserimento numero attributi

- **Scopo e descrizione:** Lo sviluppatore inserisce il numero di attributi che comporranno il tipo custom.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un tipo custom.
- **Post-condizioni:** Il tipo custom dovrà contenere il numero di attributi scelto.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il numero di attributi.
- **Estensioni:**

  a) Se il numero inserito è minore di uno, viene visualizzato un errore [UC10.4].

3.3.28 UC10.4 - Visualizzazione errore numero minimio attributi

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile creare un tipo custom con meno di un attributo.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito un numero minore di uno.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che il numero inserito non è valido.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che il numero inserito non è valido.

3.3.29 UC10.5 - Inserimento attributo tipo custom

- **Scopo e descrizione:** Lo sviluppatore inserisce un attributo per definire un tipo custom.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un tipo custom.
- **Post-condizioni:** L'attributo inserito inserito viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nome dell'attributo;
  2. Lo sviluppatore inserisce il tipo dell'attributo.
- **Estensioni:**

  a) Se il nome inserito è già stato inserito in precedenza per un altro attributo dello stesso tipo custom, viene visualizzato un errore [UC10.6];

  b) Se il tipo inserito non è riconosciuto da NaturalAPI viene visualizzato un errore [UC9.4].

3.3.30 UC10.6 - Visualizzazione errore attributo già esistente

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che esiste già un attributo con lo stesso nome.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito un nome già inserito in precedenza.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che il nome inserito non è disponibile.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che esiste già un attributo con quel nome.

3.3.31 UC11 - Aggiunta di un object

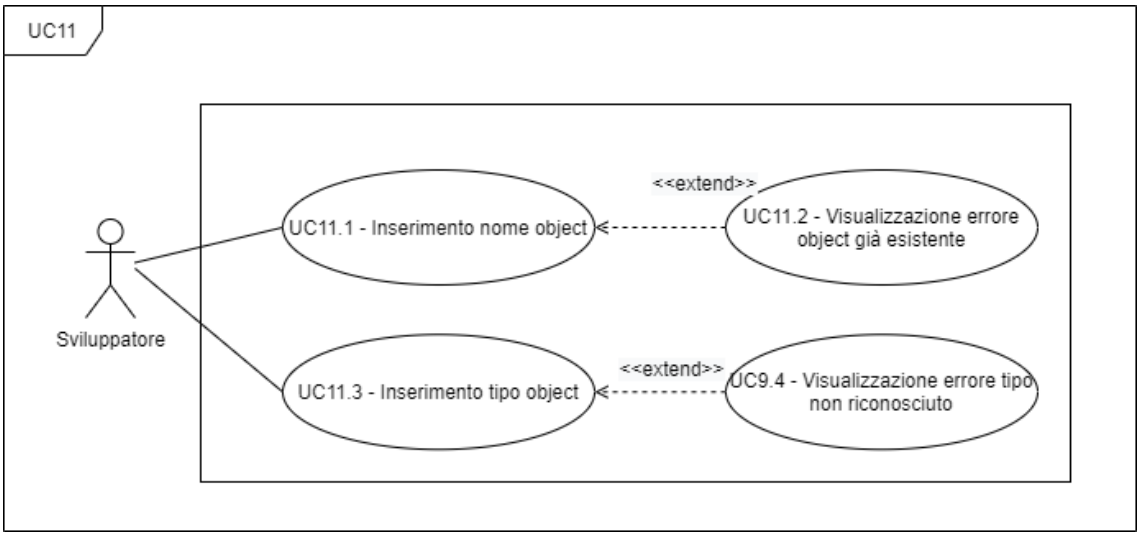

Figura 10: UC11 - Aggiunta di un object

- **Scopo e descrizione:** Lo sviluppatore aggiunge un object a una action.

- **Attore primario:** Sviluppo.
- **Pre-condizioni:** Lo sviluppatore ha selezionato la funzionalità di aggiunta object ad una action.
- **Post-condizioni:** Viene aggiunto un nuovo object.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nome dell'object [UC11.1];
  2. Lo sviluppatore inserisce il tipo dell'object [UC11.3].

3.3.32 UC11.1 - Inserimento nome object

- **Scopo e descrizione:** Lo sviluppatore inserisce il nome del nuovo object.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta aggiungendo un nuovo object a una action.
- **Post-condizioni:** Il nome del nuovo object viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nome dell'object.
- **Estensioni:**
  a) Se la action contiene già un object con lo stesso nome viene visualizzato un messaggio d'errore [UC11.2].

3.3.33 UC11.2 - Visualizzazione errore object già esistente

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile utilizzare il nome scelto per l'object.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito il nome di un object già presente nella action selezionata.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che esiste già un object con quel nome.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che esiste già un object con lo stesso nome.

3.3.34 UC11.3 - Inserimento tipo object

- **Scopo e descrizione:** Lo sviluppatore inserisce il tipo del nuovo object.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta aggiungendo un nuovo object a una action.
- **Post-condizioni:** Il tipo del nuovo object viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il tipo dell'object.
- **Estensioni:**
  a) Se il tipo inserito non è riconosciuto da NaturalAPI viene visualizzato un messaggio d'errore [UC9.4].

3.3.35 UC12 - Rimozione di un object

- **Scopo e descrizione:** Lo sviluppatore rimuove un object da una action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un object di una action.
- **Post-condizioni:** L'object scelto viene rimosso.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la rimozione.

3.3.36 UC13 - Modifica di un object

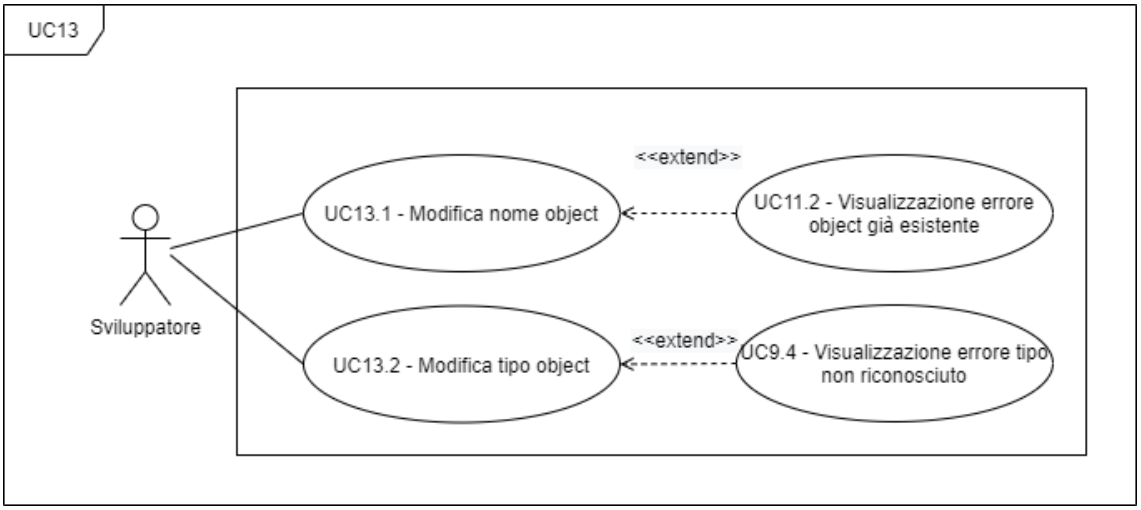

Figura 11: UC13 - Modifica di un object

- **Scopo e descrizione:** Lo sviluppatore modifica un object di una action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un object e ne ha abilitato la modifica.
- **Post-condizioni:** L'object selezionato viene modificato.
- **Scenario Principale:**
  1. Lo sviluppatore modifica il nome dell'object [UC13.1];
  2. Lo sviluppatore modifica il tipo dell'object [UC13.2].

3.3.37 UC13.1 - Modifica nome object

- **Scopo e descrizione:** Lo sviluppatore modifica il nome di un object.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando un object.
- **Post-condizioni:** Il nuovo nome dell'object viene registrato.

- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nuovo nome.
- **Estensioni:**

  a) Se la action contiene già un object con lo stesso nome viene visualizzato un messaggio d'errore [UC11.2].

3.3.38 UC13.2 - Modifica tipo object

- **Scopo e descrizione:** Lo sviluppatore modifica il tipo di un object.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando un object.
- **Post-condizioni:** Il nuovo tipo dell'object viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nuovo tipo.
- **Estensioni:**

  a) Se il tipo inserito non è riconosciuto da NaturalAPI viene visualizzato un messaggio d'errore [UC9.4].

3.3.39 UC14 - Merge di due suggerimenti

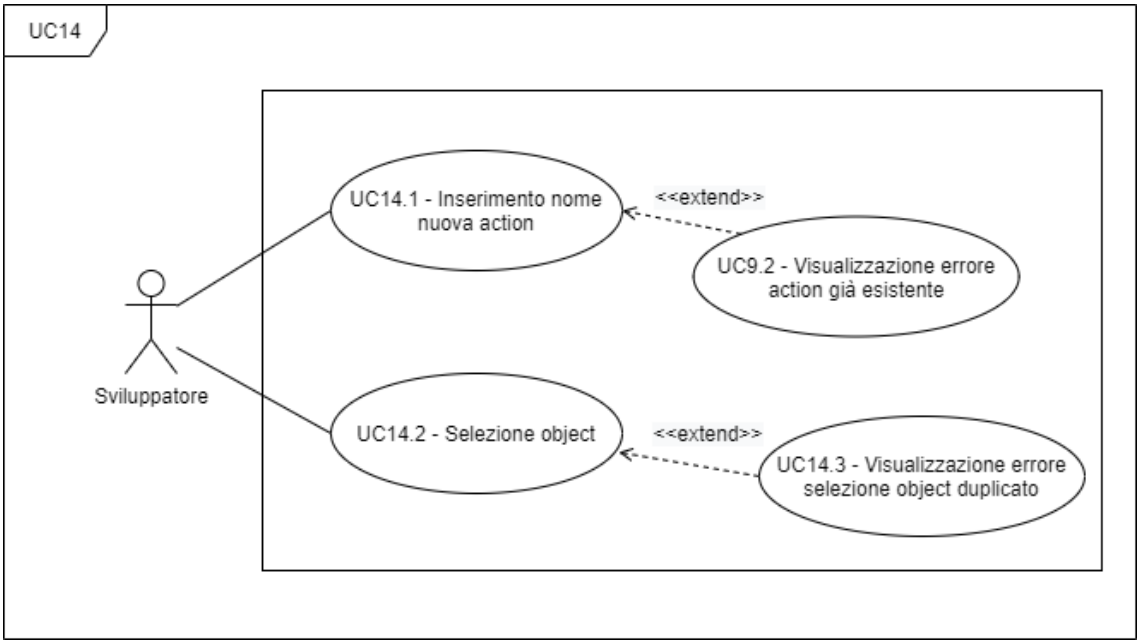

Figura 12: UC14 - Merge di due suggerimenti

- **Scopo e descrizione:** Due suggerimenti simili vengono fusi in uno.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione l'elenco di suggerimenti di un BAL. Lo sviluppatore ha selezionato due action da fondere.

- **Post-condizioni:** I due suggerimenti selezionati sono rimpiazzati da un nuovo suggerimento.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nome della nuova action [UC14.1];
  2. Lo sviluppatore seleziona gli object delle due action da conservare nella nuova action [UC14.2].

3.3.40 UC14.1 - Inserimento nome nuova action

- **Scopo e descrizione:** Viene inserito il nome della nuova action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato due suggerimenti da fondere.
- **Post-condizioni:** Viene creato un nuovo suggerimento vuoto con il nome scelto.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il nome della nuova action.
- **Estensioni:**
  a) Se esiste già una action con lo stesso nome viene visualizzato un messaggio d'errore [UC9.2].

3.3.41 UC14.2 - Selezione object

- **Scopo e descrizione:** Lo sviluppatore vuole selezionare un elenco di object.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato due suggerimenti da fondere. Lo sviluppatore ha a disposizione l'elenco degli object dei due suggerimenti selezionati.
- **Post-condizioni:** Vengono selezionati degli object.
- **Scenario Principale:**
  1. Lo sviluppatore seleziona gli object dall'elenco.
- **Estensioni:**
  a) Se lo sviluppatore seleziona due object con lo stesso nome viene visualizzato un messaggio d'errore [UC14.3].

3.3.42 UC14.3 - Visualizzazione errore selezione object duplicato

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile selezionare entrambi gli object contemporaneamente.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato due object con lo stesso nome.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che è già stato selezionato un object con quel nome.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che è già stato selezionato un object con lo stesso nome.

3.3.43 UC15 - Split di un suggerimento

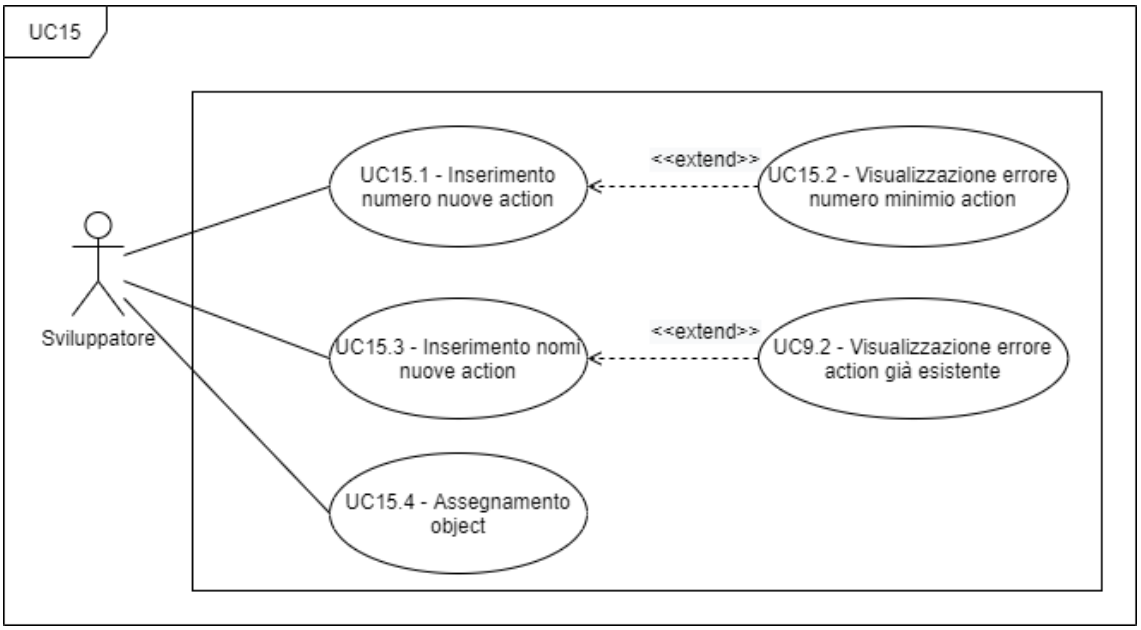

Figura 13: UC15 - Split di un suggerimento

- **Scopo e descrizione:** Un suggerimento viene suddiviso in più action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione l'elenco di suggerimenti di un BAL.
- **Post-condizioni:** Il suggerimento selezionato viene rimpiazzato da action più semplici.
- **Scenario Principale:**
  1. Lo sviluppatore seleziona il suggerimento da suddividere;
  2. Lo sviluppatore indica il numero di action in cui suddividere il suggerimento [UC15.1];
  3. Lo sviluppatore inserisce i nomi delle nuove action [UC15.3];
  4. Lo sviluppatore suddivide gli object del suggerimento tra le varie action [UC15.4].

3.3.44 UC15.1 - Inserimento numero nuove action

- **Scopo e descrizione:** Lo sviluppatore definisce il numero di action ottenute dallo split.
- **Attore primario:** Sviluppatore
- **Pre-condizioni:** Lo sviluppatore ha selezionato un suggerimento da suddividere.
- **Post-condizioni:** Il sistema memorizza il numero di nuove action da creare.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce un numero;
  2. Lo sviluppatore conferma l'inserimento.
- **Estensioni:**

  a) Se lo sviluppatore inserisce un numero minore di due viene visualizzato un errore [UC15.2].

3.3.45 UC15.2 - Visualizzazione errore numero minimio action

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile suddividere una action in meno di due parti.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito un numero minore di due.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che il numero inserito non è valido.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che il numero inserito non è valido.

3.3.46 UC15.3 - Inserimento nomi nuove action

- **Scopo e descrizione:** Viene inserito il nome delle nuove action create.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** È stato inserito il numero di action da creare.
- **Post-condizioni:** Vengono create le nuove action con i nomi inseriti dallo sviluppatore.
- **Scenario Principale:**
  1. Per ogni nuova action creata lo sviluppatore inserisce un nome;
  2. Lo sviluppatore conferma l'inserimento.
- **Estensioni:**
  a) Se esiste già una action con lo stesso nome viene visualizzato un messaggio d'errore [UC9.2].

3.3.47 UC15.4 - Assegnamento object

- **Scopo e descrizione:** Gli object del suggerimento originale vengono suddivisi nelle nuove action.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Sono state create le nuove action. Lo sviluppatore ha a disposizione l'elenco degli object del suggerimento originario.
- **Post-condizioni:** Tutti gli object vengono suddivisi.
- **Scenario Principale:**
  1. Per ogni object lo sviluppatore seleziona l'action a cui assegnarlo.
  2. Lo sviluppatore conferma le modifiche.

3.3.48 UC16 - Assegnamento di una action a un gruppo

- **Scopo e descrizione:** Lo sviluppatore inserisce una action in un gruppo.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione i suggerimenti del BAL e un elenco dei gruppi disponibili. Lo sviluppatore ha selezionato un suggerimento e un gruppo.
- **Post-condizioni:** L'action viene assegnata al gruppo.
- **Scenario Principale:**
  1. Lo sviluppatore avvia l'assegnamento.

3.3.49 UC17 - Rimozione di una action da un gruppo

- **Scopo e descrizione:** Lo sviluppatore rimuove una action dal gruppo a cui era assegnata.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un gruppo a cui sono assegnate delle action. Lo sviluppatore ha selezionato una action.
- **Post-condizioni:** L'action viene rimossa dal gruppo.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la rimozione.

3.3.50 UC18 - Generazione BAL

- **Scopo e descrizione:** Viene prodotto il BAL.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** I suggerimenti relativi a un BAL sono tutti contrassegnati come rifiutato o accettato.
- **Post-condizioni:** Viene generato il BAL. Viene memorizzata l'associazione tra il BAL e gli scenari utilizzati per produrlo.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la produzione del BAL.

3.3.51 UC19 - Aggiunta nuove feature e scenari a un BAL

- **Scopo e descrizione:** Lo sviluppatore aggiunge delle feature/scenari a un BAL già prodotto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione nuove feature o scenari e un BAL. Lo sviluppatore ha selezionato uno o più scenari e il BAL da integrare.
- **Post-condizioni:** Il BAL viene convertito nei suggerimenti che lo hanno prodotto, già contrassegnati come accettati. A questi vengono aggiunti i nuovi suggerimenti creati, relativi ai nuovi scenari.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la generazione dei suggerimenti per i nuovi scenari.

3.3.52 UC20 - Generazione suggerimenti API

- **Scopo e descrizione:** Lo sviluppatore ottiene un prototipo di API per il linguaggio di programmazione scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione dei BAL e dei PLA. Lo sviluppatore ha selezionato un BAL e un PLA.
- **Post-condizioni:** I suggerimenti vengono prodotti e sono sottoposti all'approvazione dello sviluppatore.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la produzione dei suggerimenti.

3.3.53 UC21 - Creazione PLA

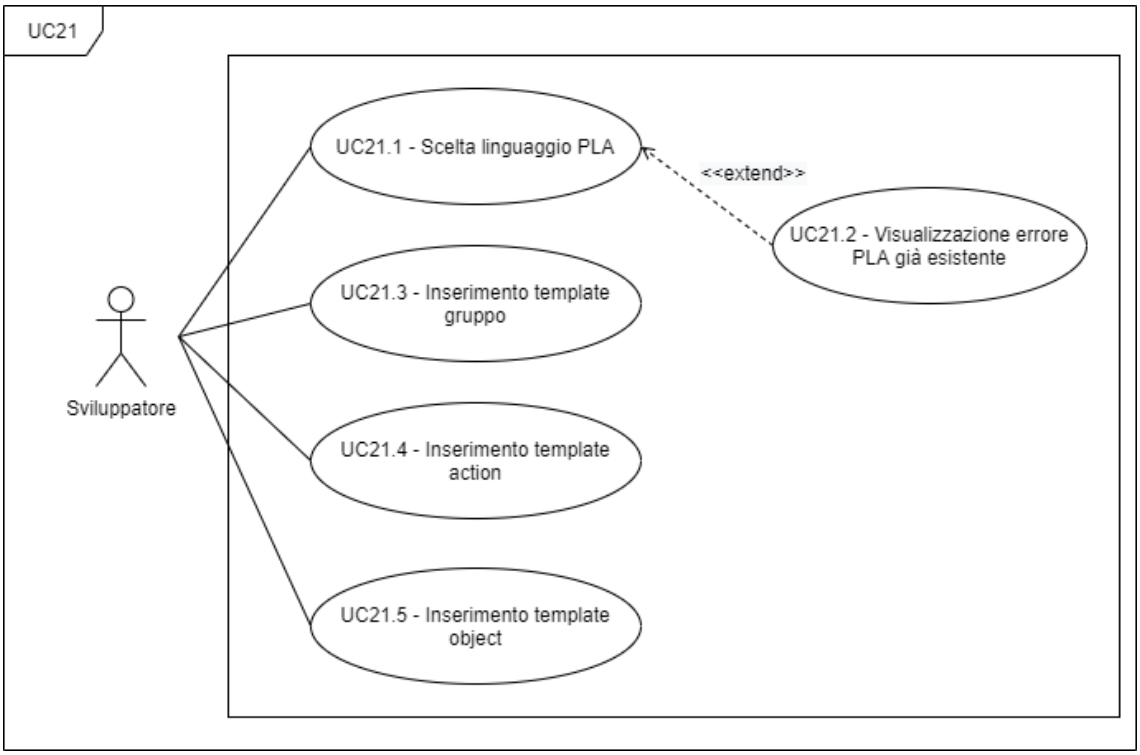

Figura 14: UC21 - Creazione PLA

- **Scopo e descrizione:** Lo sviluppatore crea un PLA per il linguaggio.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore vuole convertire un BAL in un linguaggio di programmazione di sua scelta.
- **Post-condizioni:** Viene creato un PLA per il linguaggio di programmazione scelto dallo sviluppatore.
- **Scenario Principale:**
  1. Lo sviluppatore specifica il linguaggio di programmazione del PLA [UC21.1];
  2. Lo sviluppatore inserisce il template di conversione per un gruppo [UC21.3];
  3. Lo sviluppatore inserisce il template di conversione per una action [UC21.4];
  4. Lo sviluppatore inserisce il template di conversione per un object [UC21.5].

3.3.54 UC21.1 - Scelta linguaggio PLA

- **Scopo e descrizione:** Lo sviluppatore sceglie il linguaggio di programmazione per il PLA.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un PLA.
- **Post-condizioni:** Il linguaggio scelto viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il linguaggio di programmazione.

- **Estensioni:**

  a) Se esiste già un PLA per lo stesso linguaggio viene visualizzato un errore [UC21.2].

3.3.55 UC21.2 - Visualizzazione errore PLA già esistente

- **Scopo e descrizione:** Lo sviluppatore viene avvisato che non è possibile creare un PLA per il linguaggio scelto perché già esistente.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha inserito un linguaggio per cui esiste già un PLA.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che esiste già un PLA per quel linguaggio.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che esiste già un PLA per il linguaggio scelto.

3.3.56 UC21.3 - Inserimento template gruppo

- **Scopo e descrizione:** Lo sviluppatore definisce il template per eseguire la conversione automatica di un gruppo da BAL al linguaggio scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un PLA.
- **Post-condizioni:** Il template inserito viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il template.

3.3.57 UC21.4 - Inserimento template action

- **Scopo e descrizione:** Lo sviluppatore definisce il template per eseguire la conversione automatica di una action da BAL al linguaggio scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un PLA.
- **Post-condizioni:** Il template inserito viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il template.

3.3.58 UC21.5 - Inserimento template object

- **Scopo e descrizione:** Lo sviluppatore definisce il template per eseguire la conversione automatica di un object da BAL al linguaggio scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta creando un PLA.
- **Post-condizioni:** Il template inserito viene registrato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il template.

3.3.59 UC22 - Modifica PLA

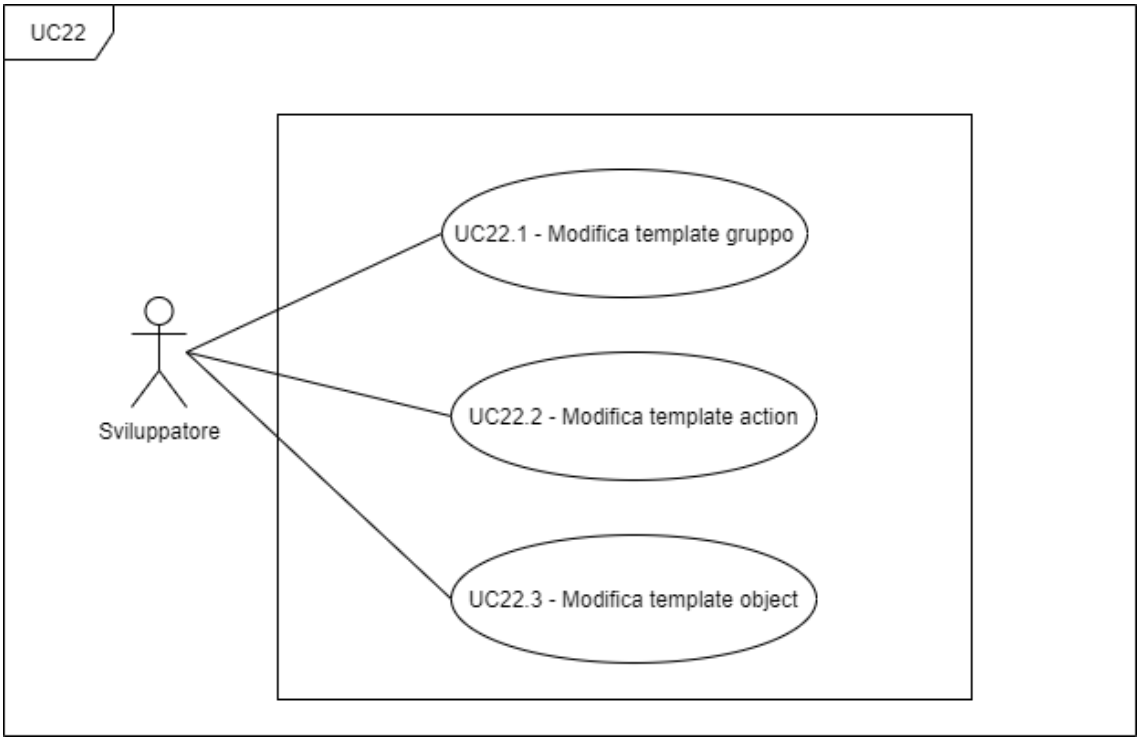

Figura 15: UC22 - Modifica PLA

- **Scopo e descrizione:** Lo sviluppatore modifica un PLA esistente per adattarlo alle sue necessità.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha a disposizione un PLA.
- **Post-condizioni:** Il PLA viene modificato con le nuove regole.
- **Scenario Principale:**
  1. Lo sviluppatore modifica il template di conversione per un gruppo [UC22.1];
  2. Lo sviluppatore modifica il template di conversione per una action [UC22.2];
  3. Lo sviluppatore modifica il template di conversione per un object [UC22.3].

3.3.60 UC22.1 - Modifica template gruppo

- **Scopo e descrizione:** Lo sviluppatore modifica il template per eseguire la conversione automatica di un gruppo da BAL al linguaggio scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando un PLA.
- **Post-condizioni:** Le modifiche vengono registrate.
- **Scenario Principale:**
  1. Lo sviluppatore modifica il template precedente.

3.3.61 UC22.2 - Modifica template action

- **Scopo e descrizione:** Lo sviluppatore modifica il template per eseguire la conversione automatica di una action da BAL al linguaggio scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando un PLA.
- **Post-condizioni:** Le modifiche vengono registrate.
- **Scenario Principale:**
  1. Lo sviluppatore modifica il template precedente.

3.3.62 UC22.3 - Modifica template object

- **Scopo e descrizione:** Lo sviluppatore modifica il template per eseguire la conversione automatica di un object da BAL al linguaggio scelto.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando un PLA.
- **Post-condizioni:** Le modifiche vengono registrate.
- **Scenario Principale:**
  1. Lo sviluppatore modifica il template precedente.

3.3.63 UC23 - Accettazione suggerimento API

- **Scopo e descrizione:** Lo sviluppatore accetta un suggerimento del prototipo API.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un suggerimento del prototipo API.
- **Post-condizioni:** Il suggerimento viene contrassegnato come accettato.
- **Scenario Principale:**
  1. Lo sviluppatore accetta il suggerimento selezionato.

3.3.64 UC24 - Modifica suggerimento API

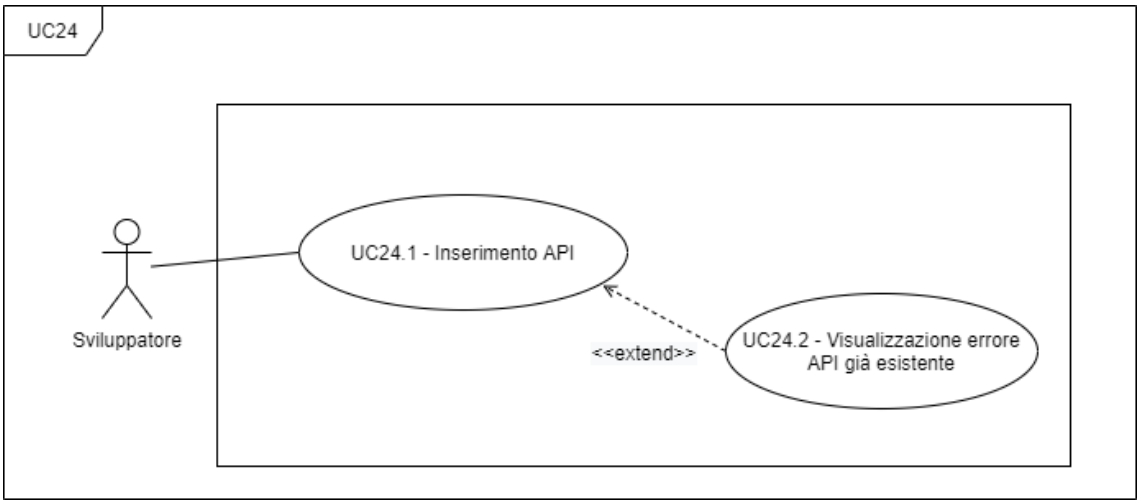

Figura 16: UC24 - Modifica suggerimento API

- **Scopo e descrizione:** Lo sviluppatore modifica un suggerimento del prototipo API.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un suggerimento del prototipo API e ne ha abilitato la modifica.
- **Post-condizioni:** Il suggerimento viene modificato.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce l'API corretta [UC24.1].

3.3.65 UC24.1 - Inserimento API

- **Scopo e descrizione:** Lo sviluppatore corregge un suggerimento di API con il testo inserito.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore sta modificando un suggerimento di API.
- **Post-condizioni:** Il suggerimento viene rimpiazzato dal testo inserito.
- **Scenario Principale:**
  1. Lo sviluppatore inserisce il testo dell'API.
- **Estensioni:**

  a) Se il testo inserito coincide con un altro suggerimento di API viene visualizzato un errore [UC24.2].

3.3.66 UC24.2 - Visualizzazione errore API già esistente

- **Scopo e descrizione:** Lo sviluppatore ha modificato un suggerimento di API rendendolo uguale a un suggerimento già esistente.
- **Attore primario:** Sviluppatore.

- **Pre-condizioni:** Lo sviluppatore ha inserito il testo di una API già esistente.
- **Post-condizioni:** NaturalAPI avvisa lo sviluppatore che esiste già un API identica.
- **Scenario Principale:**
  1. Viene visualizzato un messaggio d'errore che indica che il testo inserito è identico a quello di una API già esistente.

3.3.67 UC25 - Generazione API e test

- **Scopo e descrizione:** Lo sviluppatore ottiene le API e i test di integrazione.
- **Attore primario:** Sviluppatore.
- **Pre-condizioni:** Lo sviluppatore ha selezionato un framework per il quale sviluppare i test tra quelli forniti da NaturalAPI Develop.
- **Post-condizioni:** Vengono prodotte le API partendo dai suggerimenti accettati e i test di unità relativi agli scenari di partenza. Viene memorizzata l'associazione tra le API, il BAL e il PLA utilizzati per produrle.
- **Scenario Principale:**
  1. Lo sviluppatore avvia la produzione di API e test.

3.4 Requisiti

3.4.1 Classificazione requisiti

I requisiti sono stati classificati sulla base del ruolo svolgono nel definire un elemento di NaturalAPI:

- **RF**: Requisiti funzionali, descrivono ciò che il sistema dovrebbe fare;
- **RQ**: Requisiti di qualità: caratteristiche dell'output o dell'esecuzione;
- **RV**: Requisiti di vincolo, caratteristiche o vincoli che il sistema deve avere.

E sulla base della loro utilità strategica per il proponente:

- **Obbligatori**: Irrinunciabili per qualcuno degli stakeholder;
- **Desiderabili**: Non strettamente necessari ma a valore aggiunto riconoscibile;
- **Opzionali**: Relativamente utili oppure contrattabili avanti nel progetto.

3.4.2 Requisiti funzionali

Di seguito viene riportato l'elenco completo dei requisiti individuati. Le fonti sono il Capitolato presentato dal proponente, le discussioni con il proponente e le decisioni interne prese dal gruppo. Per ciascuna fonte è indicata la posizione della descrizione del requisito. Sono inoltre riportati i casi d'uso associati.

Tabella 2: Tabella dei requisiti funzionali

Requisito	Descrizione	Classificazione	Fonte
RF1	NaturalAPI Discover deve analizzare documenti di testo ed estrarre un BDL	Obbligatorio	Capitolato §4.1.1.1

Tabella 2 (continuazione)

Requisito	Descrizione	Classificazione	Fonte
RF1.1	NaturalAPI Discover deve estrarre un insieme di verbi dai documenti	Desiderabile	Capitolato §3.1 C
RF1.2	NaturalAPI Discover deve estrarre un insieme di nomi dai documenti	Desiderabile	Capitolato §3.1 C
RF1.3	NaturalAPI Discover deve estrarre un insieme di predicati dai documenti	Desiderabile	Capitolato §3.1 C
RF2	NaturalAPI Discover può trattare materiale scritto in lingua diversa dall'inglese	Opzionale	Capitolato §4.1.2
RF2.1	I Documenti per il BDL in una lingua diversa dall'inglese vengono tradotti	Opzionale	Capitolato §3.1 C
RF2.2	Feature e scenari scritti in una lingua diversa dall'inglese vengono tradotti	Opzionale	Capitolato §4.1.2
RF3	L'utente può selezionare uno o più documenti da fornire a NaturalAPI Discover	Obbligatorio	Capitolato §4.1.1.1
RF3.1	NaturalAPI Discover deve associare al BDL i documenti utilizzati per produrlo	Desiderabile	Interno 1
RF3.2	NaturalAPI Discover non deve analizzare più volte uno stesso documento	Desiderabile	Interno 1
RF3.3	L'utente può integrare il BDL con altri documenti	Obbligatorio	Interno 1
RF3.4	L'utente può rimuovere l'analisi di un documento dal BDL	Desiderabile	Interno 1
RF4	L'utente può selezionare un parser tra quelli messi a disposizione dal NaturalAPI Discover	Desiderabile	Interno 2
RF5	L'utente può utilizzare NaturalAPI come aiuto per la produzione di scenari e feature BDD	Desiderabile	Capitolato §3.1 D
RF5.1	L'utente può visualizzare un BDL	Desiderabile	Interno 3
RF5.2	NaturalAPI deve fornire dei suggerimenti di modifica dei termini di uno scenario, basandosi sul BDL	Desiderabile	Interno 3
RF5.3	NaturalAPI deve riconoscere sinonimi all'interno di un BDL	Desiderabile	Interno 3

Tabella 2 (continuazione)

Requisito	Descrizione	Classificazione	Fonte
RF5.4	NaturalAPI permette di apportare le modifiche a tutte le occorrenze del termine scelto	Desiderabile	Interno 3
RF5.5	NaturalAPI deve fornire un ambiente di scrittura degli scenari BDD con suggerimenti a run-time	Desiderabile	Interno 3
RF5.6	NaturalAPI deve fornire una visualizzazione dei termini più frequenti di un BDL per aiutare l'utente a redigere scenari BDD	Desiderabile	Interno 3
RF6	NaturalAPI Design deve utilizzare un BDL, degli scenari BDD e una BO opzionale per produrre un BAL	Obbligatorio	Capitolato §4.1.1.2
RF7	L'utente sceglie uno o più scenari BDD da fornire al sistema	Obbligatorio	Capitolato §4.1.1.2
RF8	L'utente sceglie un BDL da fornire al sistema	Obbligatorio	Capitolato §4.1.1.2
RF9	L'utente può integrare BDL e BDD con una BO da fornire al sistema	Obbligatorio	Capitolato §4.1.1.2
RF10	NaturalAPI Design elabora i suggerimenti per il BAL	Obbligatorio	Capitolato §4.1.1.2
RF10.1	NaturalAPI Design fornisce dei suggerimenti per ciascuno scenario	Obbligatorio	Capitolato §4.1.1.2
RF10.2	I suggerimenti sono composti da action e object	Desiderabile	Interno 4
RF11	L'utente accetta un suggerimento di BAL	Obbligatorio	Capitolato §4.1.1.2
RF11.1	NaturalAPI Design contrassegna il suggerimento come accettato	Obbligatorio	Interno 5
RF12	L'utente rifiuta un suggerimento di BAL	Obbligatorio	Interno 5
RF12.1	NaturalAPI Design contrassegna il suggerimento come rifiutato	Obbligatorio	Interno 5
RF13	L'utente può modificare un suggerimento di BAL	Obbligatorio	Capitolato §4.1.1.2
RF13.1	L'utente può modificare il nome dell'action	Obbligatorio	Interno 5

Tabella 2 (continuazione)

Requisito	Descrizione	Classificazione	Fonte
RF13.2	L'utente può aggiungere un object	Obbligatorio	Interno 5
RF13.3	L'utente può rimuovere un object	Obbligatorio	Interno 5
RF13.4	L'utente può modificare il nome di un object	Obbligatorio	Interno 5
RF13.5	L'utente può modificare il tipo di un object	Obbligatorio	Interno 5
RF14	L'utente può raggruppare due suggerimenti di BAL in uno	Desiderabile	Capitolato §3.1 E
RF14.1	L'utente può scegliere il nome della nuova action	Desiderabile	Interno 5
RF14.2	L'utente può selezionare quali object conservare	Desiderabile	Interno 5
RF15	L'utente può suddividere un suggerimento di BAL in più parti	Desiderabile	Capitolato §3.1 E
RF15.1	L'utente può decidere in quante action suddividere il suggerimento	Desiderabile	Interno 5
RF15.2	L'utente può nominare le varie action ottenute dallo split	Desiderabile	Interno 5
RF15.3	L'utente può decidere come suddividere gli object del suggerimento tra le action ottenute dallo split	Desiderabile	Interno 5
RF16	L'utente può creare un gruppo	Desiderabile	Capitolato §3.1 E
RF16.1	L'utente può assegnare un nome a un gruppo	Desiderabile	Interno 5
RF16.2	L'utente può eliminare un gruppo	Desiderabile	Interno 5
RF16.3	L'utente può assegnare una action ad un gruppo	Desiderabile	Interno 5
RF16.4	L'utente può rimuovere una action da un gruppo	Desiderabile	Interno 5
RF17	NaturalAPI Design deve associare le feature utilizzate per produrre il BAL al BAL prodotto	Desiderabile	Interno 1
RF17.1	L'utente può aggiungere feature da integrare nel BAL	Desiderabile	Interno 1

Tabella 2 (continuazione)

Requisito	Descrizione	Classificazione	Fonte
RF17.2	NaturalAPI Design deve calcolare il BAL solo per le feature aggiuntive e non svolgere di nuovo i calcoli per quelle già analizzate	Desiderabile	Interno 1
RF18	NaturalAPI Develop deve utilizzare un BAL e un PLA per produrre API e integration test	Obbligatorio	Capitolato §4.1.2.3
RF19	L'utente può creare un PLA	Opzionale	Capitolato §3.1 F
RF20	L'utente può integrare un PLA con delle regole aggiuntive	Desiderabile	Interno 1
RF21	L'utente può scegliere il framework per il quale produrre i test tra quelli supportati dal sistema	Obbligatorio	Capitolato §3.1 F
RF22	NaturalAPI Develop deve fornire le API, da sottoporre all'approvazione dell'utente	Obbligatorio	Capitolato §4.1.2.3
RF23	L'utente può accettare un suggerimento delle API	Obbligatorio	Capitolato §4.1.2.3
RF24	L'utente può modificare un suggerimento delle API	Obbligatorio	Capitolato §4.1.2.3
RF25	NaturalAPI Developer deve associare il BAL utilizzato per produrre le API alle API prodotte	Desiderabile	Interno 1
RF25.1	NaturalAPI Develop deve ricalcolare le API relative a un BAL solo se il BAL o il PLA hanno subito modifiche	Desiderabile	Interno 1
RF26	NaturalAPI Develop deve creare e aggiornare code repository già esistenti per salvare le API	Opzionale	Capitolato §2.1
RF27	L'utente può utilizzare tutte le funzionalità in un unico workflow	Desiderabile	Capitolato §4.1.2
RF28	Il software deve essere accessibile tramite due delle seguenti modalità: -Linea di comando -GUI -interfaccia web REST	Obbligatorio	Capitolato §4.2.1

Tabella 2 (continuazione)

Requisito	Descrizione	Classificazione	Fonte
RF29	I file di input/output devono avere i seguenti formati: -Documenti di testo: .txt -scenari: Gherkin (.feature) -Business Domain Language: formato CSV (.bdl) -Business Ontologies: OWL v2 -Business Application Language: OpenAPI v3 JSON Object (.bal)	Obbligatorio	Capitolato §4.4.1
RF30	I termini contenuti nel BDL devono essere in lingua Inglese	Obbligatorio	Capitolato §3.1 C

3.4.3 Requisiti prestazionali

Non sono stati individuati particolari requisiti riguardanti le performance. In ogni caso, dato che NaturalAPI Discover utilizza un parser esterno, le prestazioni del sistema sono strettamente vincolate da quelle del parser scelto.

3.4.4 Requisiti di qualità

Di seguito viene riportato l'elenco completo dei requisiti individuati. I requisiti contrassegnati come "Interno" sono stati introdotti per fornire un prodotto di qualità superiore rispetto al minimo richiesto dal capitolato.

Tabella 3: Tabella dei requisiti di qualità

Requisito	Descrizione	Classificazione	Fonte
RQ1	La struttura del software deve seguire il modello di Clean Architecture	Desiderabile	Interno
RQ2	La documentazione relativa al prodotto finito deve essere scritta in Inglese	Obbligatorio	Interno
RQ3	Lo sviluppo del software deve seguire le Norme di progetto	Obbligatorio	Interno
RQ4	Lo sviluppo del software deve seguire il Piano di qualifica	Obbligatorio	Interno
RQ5	Deve essere prodotto un Manuale utente	Obbligatorio	Interno
RQ6	La consegna del prodotto va effettuata entro e non oltre Settembre 2020	Obbligatorio	Committente

Tabella 3 (continuazione)

Requisito	Descrizione	Classificazione	Fonte
RQ7	Il codice sorgente deve essere versionato e reso pubblico tramite GitHub	Obbligatorio	Capitolato §5

3.4.5 Requisiti di vincolo

Di seguito viene riportato l'elenco completo dei requisiti individuati.

Tabella 4: Tabella dei requisiti di vincolo

Requisito	Descrizione	Classificazione	Fonte
RV1	Il sistema deve essere composto da tre moduli indipendenti denominati NaturalAPI Discover, NaturalAPI Design e Natural API Develop	Obbligatorio	Capitolato §4.1.1
RV2	Il software deve essere supportato da almeno un sistema operativo tra Ubuntu Linux, macOS, Windows	Obbligatorio	Capitolato §4.2.2
RV3	Il software deve utilizzare tecniche di NLP per produrre il BDL	Obbligatorio	Capitolato §2.2
RV4	Il BDL deve presentarsi come un elenco di verbi, nomi, predicati e relativa frequenza	Obbligatorio	Capitolato §4.1.1.1

3.4.6 Fonti interne

I requisiti indicati come interni non sono riconducibili al capitolato, ma sono emersi dalla discussione con il proponente o per la necessita dovuta alle prime idee di progettazione. Di seguito sono indicate le motivazioni per la definizione delle fonti interne.

3.4.6.1 Interno Decisioni del gruppo: i requisiti marcati come "Interno" sono emersi da considerazioni interne al gruppo.

3.4.6.2 Interno 1 Propagazione delle modifiche: Dal paragrafo 3.1 C del capitolato emerge che le modifiche dovrebbero essere propagate facilmente verso il resto dell'applicazione. Dato il funzionamento dei tre moduli, viene inteso come una propagazione in avanti, da Discover a Design e da Design a Develop. Quindi, l'integrazione di un BDL con altri documenti deve portare a una modifica ai BAL che lo utilizzano e un aggiunta di scenari a un BAL deve portare all'aggiornamento delle API. Inoltre l'aggiunta di nuovi documenti/scenari non deve comportare la rielaborazione completa del BDL/BAL per ragionevoli motivi di efficienza,

3.4.6.3 Interno 2 Scelta del Parser: Siccome il capitolato consiglia fortemente l'utilizzo di tecniche di NLP, ai fini del progetto verrà utilizzato un parser già sviluppato che le implementa. L'utente deve quindi poter scegliere, in base alla disponibilità, il parser supportato che più lo soddisfa.

3.4.6.4 Interno 3 Aiuto per la produzione di scenari: Come suggerito dal capitolato e in seguito chiarito con il proponente NaturalAPI Discover può essere utilizzato da parte degli stakeholder per produrre gli scenari. Perciò NaturalAPI Discover può mettere a disposizione una funzionalità per la scrittura di scenari Gherkin e fornire dei suggerimenti di modifica per gli scenari, basandosi sul BDL.

3.4.6.5 Interno 4 Struttura dei suggerimenti: Nel capitolato non è specificato come devono essere strutturati i suggerimenti forniti da NaturalAPI Design. Discutendo con il proponente è stato deciso che i suggerimenti sono formati da $action_{\mathrm{G}}$ e $object_{\mathrm{G}}$, prevedendo l'utilizzo degli standard object di OpenAPI.

3.4.6.6 Interno 5 Pruning: Discutendo con il proponente del paragrafo 3.1 E del capitolato è emerso che sui suggerimento è necessario svolgere un $pruning_{\mathrm{G}}$ per scartare i suggerimenti non desiderati. NaturalAPI Design deve quindi permettere di rifiutare e accettare suggerimenti, oltre che a permettere la modifica da parte dell'utente. Inoltre le funzionalità di merge, group e split vengono intese come:
merge: fusione di due suggerimenti in uno, scegliendo cosa tenere di ciascuno
group: raggruppare più suggerimenti strettamente collegati
split: suddividere un suggerimento valutato come troppo complesso in più parti, specificando la suddivisione da effettuare.

3.4.7 Tracciamento requisiti

Tabella 5: Tabella di tracciamento Fonte-Requisito

Fonte	Requisiti
Capitolato §2.1	RF26
Capitolato §2.2	RV5
Capitolato §3.1 C	RF1.1 RF1.2 RF1.3 RF2.1 RV3
Capitolato §3.1 D	RF5
Capitolato §3.1 E	RF14 RF15 RF16
Capitolato §3.1 F	RF19 RF21
Capitolato §4.1.1	RV1

Tabella 5 (continuazione)

Fonte	Requisiti
Capitolato §4.1.1.1	RF1 RF3 RV6
Capitolato §4.1.1.2	RF6 RF7 RF8 RF9 RF10 RF10.1 RF11 RF13
Capitolato §4.1.2	RF2 RF2.2 RF27
Capitolato §4.1.2.3	RF18 RF22 RF23 RF24
Capitolato §4.2.1	RF28
Capitolato §4.2.2	RV4
Capitolato §4.3.1	RV4
Capitolato §4.4.1	RF29
Capitolato §5	RQ7
Interno	RQ1 RQ2 RQ3 RQ4 RQ5

Tabella 5 (continuazione)

Fonte	Requisiti
Interno 1	RF3.1 RF3.2 RF3.3 RF3.4 RF17 RF17.1 RF17.2 RF20 RF25 RF25.1
Interno 2	RF4
Interno 3	RF5.1 RF5.2 RF5.3 RF5.4 RF5.5 RF5.6
Interno 4	RF10.2
Interno 5	RF11.1 RF12 RF12.1 RF13.1 RF13.2 RF13.3 RF13.4 RF13.5 RF14.1 RF14.2 RF15.1 RF15.2 RF15.3 RF16.1 RF16.2 RF16.3 RF16.4

Tabella 6: Tabella di tracciamento Use Case-Requisito

Fonte	Requisiti
UC1	RF1 RF1.1 RF1.2 RF1.3
UC2	RF3.3
UC3	RF3.4
UC4	RF5.5
UC4.1	-
UC4.2	-
UC4.3	-
UC4.4	-
UC4.5	-
UC4.6	-
UC4.7	-
UC5	RF5.2 RF5.3 RF5.4 RF5.5
UC5.1	RF5.4
UC5.2	-
UC6	RF6 RF7 RF8 RF10
UC7	RF11 RF11.1
UC8	RF12 RF12.1
UC9	RF13 RF13.1

Tabella 6 (continuazione)

Fonte	Requisiti
UC9.1	-
UC9.2	-
UC9.4	-
UC9.4	-
UC10	RF9
UC10.1	-
UC10.2	-
UC10.3	-
UC10.4	-
UC10.5	-
UC10.6	-
UC11	RF13 RF13.2
UC11.1	-
UC11.2	-
UC11.3	-
UC12	RF13 RF13.3
UC13	RF13 RF13.4 RF13.5
UC13.1	RF13.4
UC13.2	RF13.5
UC14	RF14 RF14.1 RF14.2
UC14.1	RF14.1

Tabella 6 (continuazione)

Fonte	Requisiti
UC14.2	RF14.2
UC14.3	-
UC15	RF15 RF15.1 RF15.2 RF15.3
UC15.1	RF15.1
UC15.2	RF15.2
UC15.3	RF15.3
UC16	RF16.3
UC17	RF16.4
UC18	RF6 RF17
UC19	RF17.1 RF17.2
UC20	RF18 RF22
UC21	RF19
UC21.1	-
UC21.2	-
UC21.3	-
UC21.4	-
UC21.5	-
UC22	RF20
UC22.1	-
UC22.2	-
UC22.3	-

Tabella 6 (continuazione)

Fonte	Requisiti
UC23	RF23
UC24	RF24
UC24.1	-
UC24.2	-
UC25	RF18 RF25

Tabella 7: Tabella di tracciamento Requisito-Fonte

Requisito	Fonte
RF1	Capitolato §4.1.1.1
RF1.1	Capitolato §3.1 C
RF1.2	Capitolato §3.1 C
RF1.3	Capitolato §3.1 C
RF2	Capitolato §4.1.2
RF2.1	Capitolato §3.1 C
RF2.2	Capitolato §4.1.2
RF3	Capitolato §4.1.1.1
RF3.1	Interno 1
RF3.2	Interno 1
RF3.3	Interno 1

Tabella 7 (continuazione)

Requisito	Fonte
RF3.4	Interno 1
RF4	Interno 2
RF5	Capitolato §3.1 D
RF5.1	Interno 3
RF5.2	Interno 3
RF5.3	Interno 3
RF5.4	Interno 3
RF5.5	Interno 3
RF5.6	Interno 3
RF6	Capitolato §4.1.1.2
RF7	Capitolato §4.1.1.2
RF8	Capitolato §4.1.1.2
RF9	Capitolato §4.1.1.2
RF10	Capitolato §4.1.1.2
RF10.1	Capitolato §4.1.1.2
RF10.2	Interno 4
RF11	Capitolato §4.1.1.2
RF11.1	Interno 5
RF12	Interno 5
RF12.1	Interno 5

Tabella 7 (continuazione)

Requisito	Fonte
RF13	Capitolato §4.1.1.2
RF13.1	Interno 5
RF13.2	Interno 5
RF13.3	Interno 5
RF13.4	Interno 5
RF13.5	Interno 5
RF14	Capitolato §3.1 E
RF14.1	Interno 5
RF14.2	Interno 5
RF15	Capitolato §3.1 E
RF15.1	Interno 5
RF15.2	Interno 5
RF15.3	Interno 5
RF16	Capitolato §3.1 E
RF16.1	Interno 5
RF16.2	Interno 5
RF16.3	Interno 5
RF16.4	Interno 5
RF17	Interno 1
RF17.1	Interno 1
RF17.2	Interno 1

Tabella 7 (continuazione)

Requisito	Fonte
RF18	Capitolato §4.1.2.3
RF19	Capitolato §3.1 F
RF20	Interno 1
RF21	Capitolato §3.1 F
RF22	Capitolato §4.1.2.3
RF23	Capitolato §4.1.2.3
RF24	Capitolato §4.1.2.3
RF25	Interno 1
RF25.1	Interno 1
RF26	Capitolato §2.1
RF27	Capitolato §4.1.2
RF28	Capitolato §4.2.1
RF29	Capitolato §4.4.1
RF30	Capitolato §3.1 C
RQ1	Interno
RQ2	Interno
RQ3	Interno
RQ4	Interno

Tabella 7 (continuazione)

Requisito	Fonte
RQ5	Interno
RQ6	Committente
RQ7	Capitolato §5
RV1	Capitolato §4.1.1
RV2	Capitolato §4.2.2
RV3	Capitolato §2.2
RV4	Capitolato §4.1.1.1

Tabella 8: Tabella di tracciamento Requisito-Use Case

Requisito	Use Case
RF1	UC1
RF1.1	UC1
RF1.2	UC1
RF1.3	UC1
RF3.3	UC2
RF3.4	UC3
RF5.2	UC5
RF5.3	UC5
RF5.4	UC5 UC5.1
RF5.5	UC4 UC5

Tabella 8 (continuazione)

Requisito	Use Case
RF6	UC6 UC18
RF7	UC6
RF8	UC6
RF9	UC10
RF10	UC6
RF11	UC7
RF11.1	UC7
RF12	UC8
RF12.1	UC8
RF13	UC9 UC11 UC12 UC13
RF13.1	UC9
RF13.2	UC11
RF13.3	UC12
RF13.4	UC13 UC13.1
RF13.5	UC13 UC13.2
RF14	UC14
RF14.1	UC14 UC14.1
RF14.2	UC14 UC14.2
RF15	UC15

Tabella 8 (continuazione)

Requisito	Use Case
RF15.1	UC15 UC15.1
RF15.2	UC15 UC15.2
RF15.3	UC15 UC15.3
RF16.3	UC16
RF16.4	UC17
RF17	UC18
RF17.1	UC19
RF17.2	UC19
RF18	UC20 UC25
RF19	UC21
RF20	UC22
RF22	UC20
RF23	UC23
RF24	UC24
RF25	UC25

3.5 Considerazioni

I requisiti identificati coprono tutti i requisiti obbligatori per la realizzazione del software e alcune caratteristiche opzionali. In caso durante la realizzazione del progetto emergano eventuali funzionalità aggiuntive, il documento verrà aggiornato.

4 Verifica requisiti

I requisiti sono stati verificati come specificato nel paragrafo 3.4.3.2 delle Norme di progetto. Dopo alcune fasi di walkthrough, i membri del gruppo hanno stabilito una checklist di caratteristiche dei requisiti da verificare. Questo processo è stato eseguito più volte durante la stesura del documento e quando ritenuto necessario. Per semplicità, gli esiti vengono inseriti nel documento di analisi dei requisiti. Se necessario, verrano stesi dei documenti specifici con i risultati delle verifiche successive. Di seguito sono riportati gli esiti della verifica prima della consegna dei documenti prima della RR del 21/01/2020.

1. **Requisiti Identificabili**
   Tutti i requisiti sono identificabili sono un ID unico che riporta il tipo di requisito e un intero incrementale.

2. **Requisiti Tracciabili**
   Tutti i requisiti sono tracciabili all'origine. La tabelle con l'elenco dei requisiti riportano la fonte del requisito, il capitolato o le discussioni interne. Inizialmente, i requisiti riportavano solo l'indicazione di "Capitolato" o "Interno". Successivamente sono state inserite indicazioni più precise.

3. **Requisiti Verificabili**
   Fino a questo punto, tutti i requisiti sembrano verificabile. Abbiamo creato una serie di test disponibili nel Piano di qualifica che quindi hanno due scopi, capire se il requisito è verificabile e fornire una serie (se pur iniziale) di test di sistema. Questo è coerente con il modello a V, che prevede di iniziare a pensare ai test già nella fase di analisi.

4. **Requisiti Classificabili**
   I requisiti sono classificati sulla base del loro ruolo nel progetto e nella loro importanza per il proponente. Durante la verifica, ci siamo accorti di non aver inserito la definizione delle metriche di valutazione e abbiamo corretto i documenti inserendole.

5. **Requisiti Completi**
   I requisiti individuati coprono tutte le richieste fatte dal proponente.

6. **Requisiti Realizzabili**
   Una piccola parte dell'analisi è stata dedicata ad alcune delle tecnologie impiegate per l'implementazione di NaturalAPI, lo Stanford parser e il formato openAPI. I particolare abbiamo discusso col proponente come implementare NaturalAPI Design in modo tale da essere coerenti con il tool utilizzato per gestire il formato openAPI, Swagger. Dall'analisi preliminare le tecnologie suggerite dal proponente soddisfano i requisiti e sembrano essere alla nostra portata. Ulteriori approfondimenti sono ancora necessari.

7. **Requisiti Validi**
   Per stabilire la validità dei requisiti abbiamo utilizzato il capitolato e in caso di dubbi abbiamo dialogato col proponente. In questi termini, i requisiti che abbiamo discusso maggiormente sono:

   - RF5 per capire in che modo NaturalAPI debba fornire suggerimenti per le feature.
   - In generale, i requisiti relativi a NaturalAPI Discover per definire meglio il ruolo degli stakeholder.
   - i requisiti relativi al pruning RF11, RF.12.