Documento dei Requisiti di Sistema

Article Analyzer

Artico Filippo, Colella Pierpaolo, Pasinato Alberto, Scantamburlo Mattia

Versione 1.0

Data di aggiornamento: 14/06/2023

Esame di Elementi di Ingegneria del Software – Prof. Emanuele Di Buccio

A.A. 2022-2023

1. Prefazione

Questo documento ha l'obiettivo di illustrare le funzionalità del sistema "Article Analyzer", l'interazione dell'utente con lo stesso, i suoi vincoli e le sue condizioni di utilizzo.
Il documento espone i requisiti funzionali e non funzionali del sistema, inoltre contiene una descrizione dei suoi componenti, la sua struttura, i vari casi d'uso e un'illustrazione delle relazioni tra essi mediante modelli grafici.

Questo documento si rivolge a chiunque voglia usufruire dei servizi offerti dal programma, quindi a un generico utente il cui scopo è il medesimo di "Article Analyzer", descritto nei successivi capitoli. Si rivolge inoltre a ingegneri del sistema, ingegneri del test del sistema e ingegneri della manutenzione del sistema, e più in generale a chiunque possa essere utile avere delle informazioni sullo scopo, struttura, funzioni e requisiti del progetto.

2. Introduzione

2.1 Scopo del progetto

Il sistema "Article Analyzer" descritto in questo documento ha come obiettivo la realizzazione di un sistema in grado di scaricare articoli da testate giornalistiche online resi disponibili da diverse sorgenti, di estrarre i 50 termini[1] più "importanti" nell'insieme degli articoli scaricati e inserirli in un file di testo.
Con "importanti" si fa riferimento al "peso[2]" di un termine, ovvero verranno visualizzati i 50 termini con peso maggiore che appaiono negli articoli analizzati.
Nel conteggio dei termini più pesanti non vengono considerati, e quindi vengono ignorati nel momento di analisi di ciascun articolo, i cosiddetti "termini comuni[3]", con la diretta conseguenza che non verranno mai visualizzati nella lista dei termini più pesanti.

2.2 Panoramica del documento

Il documento tratterà i seguenti argomenti:

- *Capitolo 3*: glossario descrittivo dei termini tecnici utilizzati nel documento.
- *Capitolo 4*: descrizione dei requisiti dell'utente funzionali e non funzionali, dei servizi forniti all'utente.
- *Capitolo 5*: descrizione della struttura generale del sistema, descrizione delle sue componenti e delle loro funzioni principali.
- *Capitolo 6*: specifica dei Requisiti del Sistema, funzionali e non funzionali.
- *Capitolo 7*: rappresentazione grafica delle relazioni tra le componenti che formano il sistema mediante modelli grafici UML. Rappresentazione grafica dei casi d'uso con relativa descrizione testuale.

---

1 *Termine*: vedi 3. Glossario: 3.1 Termine/Parola
2 *Peso*: vedi 3. Glossario: 3.2 Peso (di un termine)
3 *Termini comuni*: vedi 3. Glossario: 3.3 Termine comune

3. Glossario

Le seguenti definizioni sono utili alla comprensione di parole chiave utilizzate in questo documento:

3.1 Termine/Parola: parola che compare nel testo dell'articolo (titolo + corpo).

3.2 Peso (di un termine): numero di articoli in cui il termine compare, indipendentemente dal numero di volte in cui il termine considerato appare all'interno dello stesso articolo (e.g. se un articolo appare in solo 2 articoli tra i 1000 analizzati, e all'interno di questi 2 articoli appare per un totale di 10 volte, il suo *peso* sarà 2).

3.3 Termine comune: termine che appare frequentemente negli articoli e quindi di poco interesse per l'analisi dei termini più pesanti (sono esempi di termini comuni gli articoli quali *a*, *the*, ecc, oppure verbi quali *be*, *have*, *get*, ecc). Nello specifico, ai fini del progetto, un termine è considerato termine comune se appare nel seguente file (modificabile dall'utente in base alle proprie esigenze): "article_analyzer\resources\blacklist\words.txt".

3.4 Sorgente: qualsiasi tipo di input che possa essere accettato dal sistema per l'elaborazione degli articoli, attraverso file locali o contenuti ottenibili tramite l'interazione con servizi online (API).

3.5 API: (Application Programming Interface) interfaccia pubblica di un'applicazione resa disponibile online che lavora da intermediario con altre applicazioni, utilizzata nel sistema come sorgente per ottenere articoli interagendo direttamente con uno specifico giornale.

3.6 API Key: è un codice univoco utilizzato da un'API per identificare l'applicazione o l'utente chiamante. Le chiavi API sono utilizzate per tracciare e controllare chi utilizza un'API e come la utilizza, nonché per autenticare e autorizzare le applicazioni, in modo simile a come funzionano i nomi utente e le password.

3.7 URL: (Uniform Resource Locator) è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet. È l'elemento che ci permette di trovare un sito web, cioè l'indirizzo che si digita nel browser quando si cerca una pagina o un file.

3.8 Utente: la persona, o le persone, che utilizzano o interagiscono direttamente con il sistema.

3.9 Log (file): file in cui vengono salvati gli articoli analizzati durante un'esecuzione del programma. Il log file diventa una delle tre diverse sorgenti da cui analizzare gli articoli: si ha la possibilità di analizzare gli articoli già analizzati durante una precedente esecuzione del programma.

3.10 API Endpoint: luogo digitale esposto tramite l'API dal quale l'API riceve le richieste e invia le risposte. Ogni endpoint è un URL che fornisce la posizione di una risorsa sul server dell'API.

3.11 Token: (o token lessicale), è un blocco di testo categorizzato, normalmente costituito da caratteri indivisibili chiamati lessemi. I token sono frequentemente definiti come espressioni regolari, che sono comprese da un analizzatore lessicale. L'analizzatore lessicale legge in un flusso di lessemi e li categorizza in token.

3.12 UML: (Unified Modeling Language) è un linguaggio che permette, tramite l'utilizzo di modelli grafici, di analizzare, descrivere, specificare e documentare un sistema software.

3.13 Standard input: canale a cui giunge il flusso di dati in ingresso al programma, il quale effettua un'operazione di lettura dell'inserimento dell'utente

4. Definizione dei requisiti dell'utente

Questo capitolo definisce le funzionalità che il sistema deve essere in grado di eseguire per assecondare le richieste dell'utente e come queste saranno portate a termine, chiarificandone le limitazioni e minime aspettative.

4.1 Requisiti funzionali

4.1.4 L'utente deve essere in grado di specificare al sistema **diversi tipi di sorgenti**[4] come input per l'elaborazione di articoli.

4.1.5 Il sistema deve poter accettare come sorgente un **file contenente articoli o un servizio da cui scaricarli**, in base a quanto specificato dall'utente nella fase di input.

4.1.6 L'utente deve essere in grado di specificare al sistema se eseguire **solo il download** degli articoli da un servizio online, **solo l'estrazione dei termini** dagli articoli, **o entrambe le azioni** in sequenza.

4.1.7 Il sistema deve visualizzare un **messaggio di errore** nel caso in cui la sorgente specificata non sia raggiungibile.

4.1.8 Dopo aver verificato l'input, il sistema deve elaborare gli articoli specificati dall'utente e visualizzare in output/**inserire in un file** le *50 parole* (o meno in caso non siano presenti 50 parole analizzabili) *di peso maggiore* negli articoli analizzati.

4.1.9 Nell'elaborazione degli articoli, il sistema deve **scartare i termini comuni** dal conteggio di frequenza, confrontandole con una lista che può essere modificata dall'utente in base alle proprie esigenze.

4.2 Requisiti non funzionali (o di qualità)

4.2.1 L'interfaccia utente è **limitata** alla finestra di un terminale con cui l'utente può avviare l'esecuzione del sistema.

4.2.2 L'utente deve essere in grado di operare il sistema **senza previa conoscenza** del suo funzionamento, ma **sfruttando i messaggi di errore** visualizzati per aiutare nell'apprendimento del corretto utilizzo del sistema in *al più 2 tentativi*.

4.2.3 All'utente è data la possibilità di specificare il **numero di termini** da visualizzare nella classifica finale, nel caso fosse nel suo interesse osservare un numero di termini diverso dai 50 di default.

4.2.4 Per facilitare la **verifica del progresso** durante il download degli articoli dall'API Endpoint[5], viene visualizzata graficamente nel terminale una *barra in percentuale* e il *tempo stimato* insieme a quello *trascorso* per la procedura.

4.2.5 In caso di errato inserimento da parte dell'utente, anziché terminare la propria esecuzione e perdere il progresso dell'utente, il sistema deve *chiedere nuovamente l'inserimento* dei campi necessari nella fase di input.

4.2.6 L'utente deve essere in grado di **visualizzare i risultati** della propria ricerca *in meno di un minuto*.

4.2.7 L'utente è in grado di **accedere ai file** del sistema e modificarli secondo le proprie esigenze, inoltre deve poter accedere alle cartelle contenenti le **risorse del progetto** per poter specificare delle *sorgenti diverse* utilizzando le cartelle preesistenti.

4.2.8 Il sistema deve poter funzionare anche in *assenza di connessione* qualora si vogliano utilizzare **sorgenti locali** già precedentemente scaricate, ma è necessaria una

[4] Sorgente: vedi 3. Glossario: 3.4 Sorgente
[5] API Endpoint: vedi 3. Glossario: 3.10 API Endpoint

connessione per poter sfruttare le funzionalità che sfruttano i **servizi online**, altrimenti il sistema visualizza un *messaggio di errore*.

4.2.9 Il sistema deve essere in grado di ricevere una risposta dai **servizi online** entro *15 secondi* dopo aver tentato di stabilire una connessione.

4.2.10 Il sistema **non deve pubblicare o condividere dati personali** forniti dall'utente, ad esempio *chiavi private* da utilizzare per la comunicazione con servizi online.

4.2.11 Il sistema è in grado di gestire carichi di lavoro diversi *in tempi accettabili ma variabili* in base alle specifiche dell'hardware utilizzato dall'utente e alla dimensione delle sorgenti o al numero degli articoli specificati durante la fase di input, che possono rallentare l'elaborazione dei dati e la visualizzazione dell'output, ma deve elaborare i dati in modo efficiente e tale da permettere l'analisi di **1000 articoli**.

5. Architettura del sistema

Il sistema deve essere in grado di scaricare articoli (fase di **download**) da testate giornalistiche online resi disponibili dalle due seguenti sorgenti:

1. File CSV
2. The Guardian API

A tal fine si identifica un componente del sistema che rappresenta un **generico articolo**, che ha come ruolo quello di contenere: **il contenuto** dello stesso (titolo+corpo), gli attributi che permettono di identificare l'articolo all'interno del sito della testata giornalistica da cui proviene.

Per scaricare gli articoli dalla sorgente *1: File CSV*, è necessario un componente in grado di **analizzare il file CSV** e di **estrarre tutti gli articoli** in esso contenuti.

Per scaricare gli articoli dalla sorgente *2: The Guardian API*, essendo essa una sorgente *remota*, è necessario un componente in grado di stabilire un protocollo di **interazione tra il sistema e l'API Endpoint**. Questo componente deve quindi **connettersi** all'API Endpoint, deve inviare una **richiesta** all' API Endpoint in cui specifica quanti e quali articoli vuole ottenere, e deve ricevere dall'API Endpoint una **risposta** dalla quale il componente deve poi **estrarre tutti gli articoli** contenuti.

Dopo la fase di download, deve essere effettuata la **persistenza su file** degli articoli usando lo stesso formato per tutti gli articoli di tutte le sorgenti.

Per questo scopo è necessaria la presenza di un componente in grado di **raccogliere gli articoli** estratti e scaricati dai due componenti appena descritti nella fase di download degli articoli. Questi articoli vanno **inseriti all'interno di un file** che ha un formato comune indipendentemente dalla sorgente da cui sono stati ricavati gli articoli (il file appena descritto prende il nome, in questo documento e in generale nella totalità del progetto, di *file di log*).

Il *file di log* costituisce un **terzo tipo di sorgente**: con questo file infatti, grazie a un componente in grado di analizzarlo e di **estrarne tutti gli articoli** in esso contenuti, è possibile chiedere di analizzare articoli già **precedentemente scaricati**, evitando per esempio, in caso gli articoli contenuti in esso provenissero dall'API del The Guardian, di dover essere collegati ad una rete internet.

Per raggiungere lo scopo finale del sistema, ovvero la visualizzazione e stampa dei 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) con peso maggiore tra gli articoli analizzati, è necessario un componente in grado di **analizzare il contenuto di tutti gli articoli** scaricati nella fase di download, **dividere tutti i termini** in essi contenuti (concetto di "*token*[6]") e **tenere conto del peso** di ogni termine considerato, per infine consegnare i 50 (o un numero diverso se esplicitato dall'utente in fase di input) token aventi peso maggiore.

[6] *Token*: vedi 3. Glossario: 3.11 Token

6. Specifica dei requisiti del sistema

6.1. Interfacce esterne

6.1.1. ***File system e memoria***: il sistema e l'utente possono accedere e interagire con il file system del sistema operativo all'interno della directory in cui viene avviata la sua esecuzione, quindi devono avere i permessi necessari per utilizzare memoria di archiviazione del dispositivo dell'utente.

6.1.2. ***Modalità di stampa***: il sistema **visualizza a video**, nel terminale in cui è stato eseguito, le 50 (o in numero alterato in caso sia stato diversamente specificato dall'utente, o minore se il numero di termini analizzabili risultasse inferiore) parole con peso maggiore e allo stesso tempo le **stampa in un file di testo** in formato ".txt" (con nome specificato dall'utente) all'interno della cartella "article_analyzer/resources/results". Questa operazione di stampa avviene seguendo le seguenti due regole: le parole vengono scritte in *ordine decrescente di peso* e, in caso di peso uguale tra due o più termini, essi vengono stampati in *ordine lessicograficamente crescente*.

6.1.3. ***Interazione con l'utente***: il sistema deve fornire la possibilità di essere eseguito direttamente **da linea di comando** oltre che supportare **interazione dinamica** con l'utente da *standard input*.

6.1.4. ***Interazione con API e servizi online***: il sistema supporta la comunicazione tramite **protocollo HTTP** con le interfacce rese disponibili in rete, in particolare per l'API Endpoint della testata giornalistica online "The Guardian".

6.2. Funzioni

6.2.1. L'utente deve essere in grado di **specificare al sistema diversi tipi di sorgenti** come input per l'elaborazione di articoli, le diverse sorgenti selezionabili sono le seguenti:

1. *file con estensione ".json"* contenente articoli *già analizzati* in precedenti esecuzioni del programma (**file di log**);
2. *file con estensione ".csv"* che descrive i seguenti campi per ogni articolo contenuto in esso: *Identifier* (identificatore univoco dell'articolo), *URL* (dell'articolo), *Title* (titolo dell'articolo), *Body* (corpo dell'articolo), *Date* (data di pubblicazione), *Source Set* (quotidiano), *Source* (sezione in cui è stato pubblicato l'articolo);
3. *l'API* della testata giornalistica online "The Guardian", che produce *risposta in formato ".json"*.

6.2.2. Il sistema deve poter supportare **nuove tipologie di sorgenti** oltre a quelle già specificate.

6.2.3. Per estrarre i termini ed il loro peso, il sistema deve *partire dai file in cui gli articoli sono memorizzati*.

6.2.4. L'utente deve poter specificare se eseguire **solo il download**, **solo l'estrazione dei termini** a partire dai file in cui sono stati memorizzati gli articoli, o **entrambe** le azioni in sequenza.

6.2.5. Nel caso la sorgente specificata dall'utente **sia il file di log** in formato ".json", all'utente deve venire richiesto il nome del file da cui analizzare gli articoli, che deve essere inserito in input *escludendo l'estensione finale*. Il sistema analizza

successivamente gli articoli contenuti nel file specificato stampando i 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) con peso maggiore seguendo la *modalità di stampa (6.1.2)*.

6.2.6. Nel caso la sorgente specificata **non sia il file di log** in formato ".json", all'utente deve venire richiesto il nome del file su cui salvare gli articoli analizzati (esso *sarà il file di log* in formato ".json" per eventuali utilizzi futuri).

6.2.7. Se la sorgente specificata è il **file con formato ".csv"**, all'utente viene richiesto di inserire il nome del file sorgente e il sistema analizza i file contenuti in questo file stampando le 50 parole (o un numero diverso se esplicitato dall'utente in fase di input) con peso maggiore seguendo la modalità di stampa.

6.2.8. Se la sorgente specificata è **l'API** del "The Guardian", il sistema chiede se utilizzare la *API Key* di default o se l'utente vuole inserire la propria, successivamente chiede l'inserimento dei seguenti campi per **creare l'URL** e fare la richiesta al sito del "The Guardian": eventuali *queries*, eventuali *tags* e il numero di articoli da analizzare. Infine chiede all'utente se desidera visualizzare a video le 50 parole con peso maggiore e in caso di risposta affermativa il sistema le visualizza in output seguendo la *modalità di stampa (6.1.2)*.

6.2.9. Il sistema deve **verificare la validità dell'input** inserito dall'utente e visualizzare eventuali **messaggi di errore** in caso non sia nel formato previsto.

6.2.10. L'utente ha **accesso alle cartelle interne** al progetto e può pertanto aggiungere e modificare file, in particolare potrebbe essere nel suo interesse:

1. inserire *nuovi file sorgente* con formato ".csv", che vanno inseriti nella cartella "article_analyzer\resources\CSV_sources";
2. modificare la *lista di termini comuni*, e quindi da ignorare ai fini del calcolo dei pesi dei termini contenuti nell'articolo, contenuta nel file "article_analyzer\resources\blacklist\words.txt".

6.3. Requisiti di prestazione

6.3.1. La componente per le comunicazioni con i servizi online (client) attraverso il protocollo HTTP deve stabilire una **connessione** al massimo *entro 5 secondi*, deve effettuare una **richiesta** *entro 5 secondi* e deve ricevere una **risposta** *entro 5 secondi*, per un'interazione totale massima di *15 secondi*.

6.3.2. Il sistema deve poter **scaricare ed analizzare** *1000 articoli* ottenuti mediante l'API del "The Guardian" ed **estrarre ed analizzare** *1000 articoli* da un file in formato ".csv", efficientemente.

6.4. Limitazioni di design

6.4.1. Per evitare complicazioni durante la progettazione, si assume che articoli ottenuti dalla *stessa sorgente* abbiano gli *stessi attributi*.

6.4.2. Per il proprio funzionamento, il sistema necessita di operare un terminale o un ambiente di sviluppo integrato o usufruire di altri metodi per la compilazione del codice.

6.4.3. Per essere utilizzata, l'API di "The Guardian" necessita di una *chiave personale per sviluppatori*, ottenibile gratuitamente, che viene richiesta all'utente per sostituire quella di testing.

6.5. Qualità di sistema

6.5.1. Affidabilità

6.5.1.1. Il sistema deve essere disponibile per l'utilizzo per *24 ore al giorno* e *365 giorni all'anno*.

6.5.1.2. I **servizi online** hanno *disponibilità indipendente dal sistema* e potrebbero risultare indisponibili in qualsiasi momento, *causando errori* nella ricerca di sorgenti remote.

6.5.2. Sicurezza

6.5.2.1. Le comunicazioni con servizi online vengono effettuate attraverso la protezione del **protocollo HTTP**, o HTTPS in caso disponibile e appropriatamente specificato durante la *formazione dell'URL* necessario a interagire con l'API.

6.5.2.2. La *chiave privata* utilizzata dall'utente per contattare l'API deve essere raggiungibile **solo localmente** dal sistema e **memorizzata per riutilizzo futuro** nel file "article_analyzer/resources/private/private.properties", *senza essere altrimenti condivisa*.

6.5.2.3. Dopo la fase di download, deve essere effettuata la **persistenza su file** in formato ".json" degli articoli, *indipendentemente dalla tipologia di sorgente utilizzata*.

6.5.3. Manutenibilità

6.5.3.1. L'interazione con l'API di "The Guardian", in quanto di terze parti, *può mutare* in base a futuri cambiamenti effettuati alla risposta inviata dall'API stessa, quindi eventuali modifiche del sistema potrebbero risultare necessarie e devono essere apportate *in parallelo ad aggiornamenti effettuati a quest'ultima*, in modo tale da garantire il funzionamento previsto al momento della progettazione del sistema.

6.5.3.2. Per permettere il supporto di nuove sorgenti in futuro, le componenti devono prevedere un certo livello di **modularità** che consenta ad ingegneri del software di apportare modifiche in base alle proprie esigenze, ad esempio tramite *astrazioni* o *interfacce* per implementazioni più specifiche.

6.5.3.3. Il sistema deve poter supportare **nuove strutture per memorizzare ed avere accesso ai termini più importanti**.

6.5.3.4. Il sistema deve poter supportare **nuove modalità di memorizzazione ed accesso agli articoli**.

6.5.4. Portabilità

6.5.4.1. Il progetto deve poter essere eseguibile su qualsiasi sistema operativo che supporti **Java**.

6.5.4.2. Il progetto è sviluppato a partire dalla versione **Java 8 (JDK 1.8)**, che permette il corretto funzionamento di tutte le componenti e *librerie esterne* del sistema.

7. Modelli del sistema

7.1 Casi d'uso

7.1.1 UC1: Analisi di articoli *da file di log*

Attori: utente, sistema
Precondizioni: L'utente deve sapere il nome del file di log presente al percorso "article_analyzer/resources/log/"
Basic flow:

1) L'utente seleziona di voler effettuare l'analisi di articoli da un file preesistente, precedentemente caricato o generato nell'apposita cartella di log, contenente le sorgenti in formato ".json".
2) Il sistema riceve in input (tramite argomenti da terminale o come user input durante l'esecuzione) il nome del file di log da cui estrarre i termini dagli articoli *già analizzati*.
3) Il sistema, dopo aver validato l'input, verifica la presenza di un file "article_analyzer\resources\blacklist\words.txt" contenente i termini comuni, da ignorare durante il conteggio.
4) Il sistema elabora gli articoli presenti nel file di log e visualizza i 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) più "pesanti".

Alternative flow:

1) Se l'utente inserisce il nome di un file inesistente o in formato errato, il sistema restituisce un errore.
2) Se il contenuto del file non rispetta il formato utilizzato per i file di log, il sistema restituisce un errore.

Post conditions:
La frequenza con cui compaiono i 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) "rilevanti" negli articoli viene restituita in output, i risultati sono salvati in un file di testo e viene generato un file di log della ricerca effettuata contenente gli articoli analizzati.

7.1.2 UC2: Analisi di articoli *da file CSV*

Attori: utente, sistema
Precondizioni:
L'utente deve sapere il nome del file di CSV da considerare. Questo deve essere salvato nel percorso "article_analyzer/resources/CSV_sources/" e deve essere suddiviso nei campi "*Identifier,URL,Title,Body,Date,Source Set,Source*"
Basic flow:

1) L'utente seleziona di effettuare l'analisi di articoli da un file CSV, precedentemente caricato nell'apposita cartella contenente le sorgenti in formato *.csv*.
2) Il sistema riceve in input (tramite argomenti da terminale o come user input durante l'esecuzione) il nome del file CSV da cui gli articoli da analizzare, suddivisi nei campi come descritto nelle precondizioni, e il nome del file in cui salvare i risultati della ricerca effettuata.
3) Il sistema, dopo aver validato l'input, verifica la presenza di un file "article_analyzer\resources\blacklist\words.txt" contenente dei termini da ignorare durante il conteggio.
4) Il sistema elabora gli articoli presenti nel file CSV e visualizza i 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) più "pesanti".

Alternative flow:
1) Se l'utente inserisce il nome di un file inesistente o in formato errato, il sistema restituisce un errore.
2) Se il contenuto del file non rispetta il formato utilizzato per i file CSV, il sistema restituisce un errore.

Post conditions:
La frequenza con cui compaiono i 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) più "pesanti" negli articoli viene restituita in output, i risultati sono salvati in un file di testo e viene generato un file di log della ricerca effettuata contenente gli articoli analizzati.

7.1.3 UC3: Analisi di articoli *da API Endpoint di "The Guardian"*

Attori: utente, sistema, API

Precondizioni:
L'utente deve avere una API Key personale per interagire con il servizio online fornito dal The Guardian da specificare durante la fase di user input o da inserire in "article_analyzer/resources/private/private.properties".

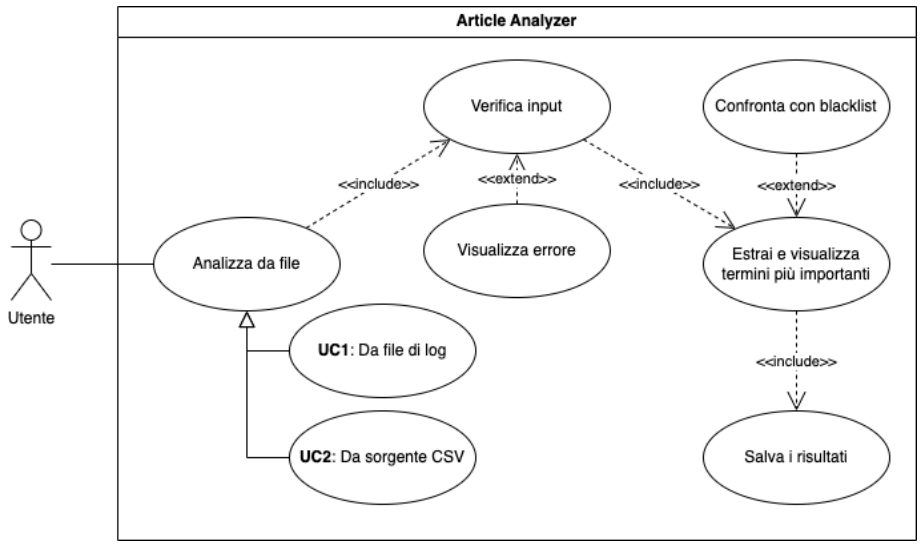
Basic flow:
1) L'utente seleziona di effettuare l'analisi di articoli dall'API di "The Guardian", seguendo le precondizioni riguardanti l'API key.
2) Il sistema riceve in input (tramite argomenti da terminale o come user input durante l'esecuzione) l'API Key, il nome del file in cui salvare i risultati della ricerca effettuata, e genera una richiesta all'API Endpoint in base alle richieste di argomento, tag e numero massimo di articoli da scaricare ed analizzare.
3) Il sistema, dopo aver validato l'input, verifica la presenza di un file "article_analyzer\resources\blacklist\words.txt" contenente dei termini da ignorare durante il conteggio.
4) Il sistema elabora gli articoli scaricati e visualizza i 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) più "pesanti".

Alternative flow:
1) Se l'utente inserisce un'API Key errata, il sistema restituisce un errore.
2) Se la richiesta all'API comprende argomenti, tag o numero massimo di articoli errato, il sistema restituisce un errore.

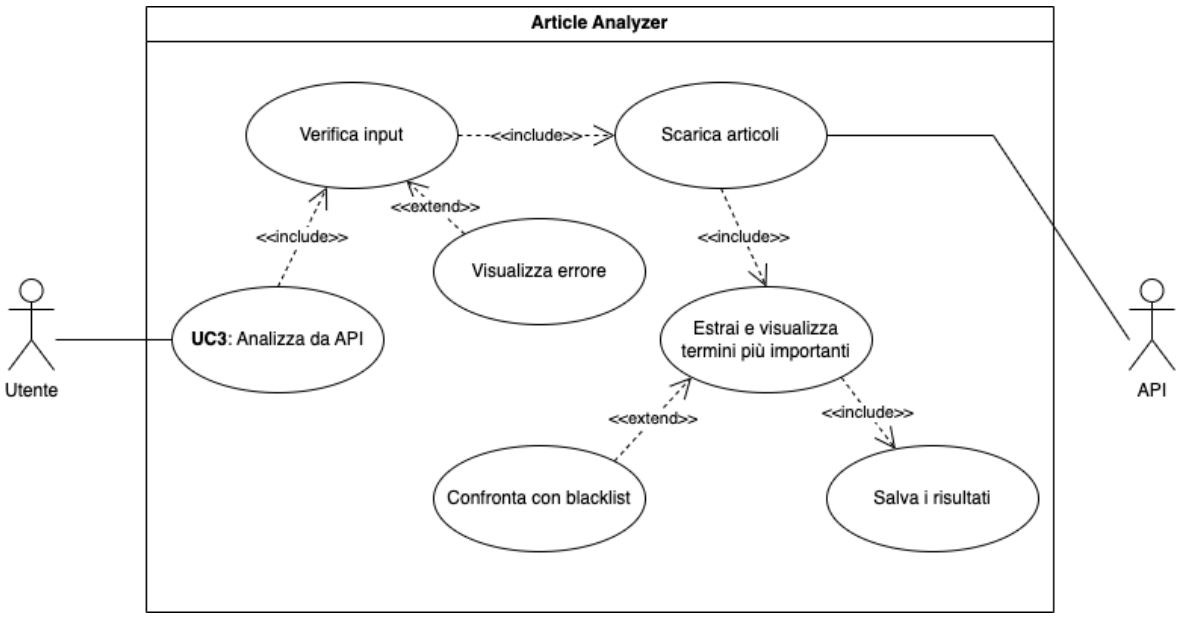
Post conditions:
Viene richiesto all'utente se desidera visualizzare in output la lista dei 50 termini (o un numero diverso se esplicitato dall'utente in fase di input) più "pesanti", in caso di risposta affermativa il sistema li visualizza in output. I risultati sono salvati in un file di testo e viene generato un file di log della ricerca effettuata contenente gli articoli scaricati ed analizzati.

7.1.4 Diagrammi di casi d'uso

UC1 e **UC2**:

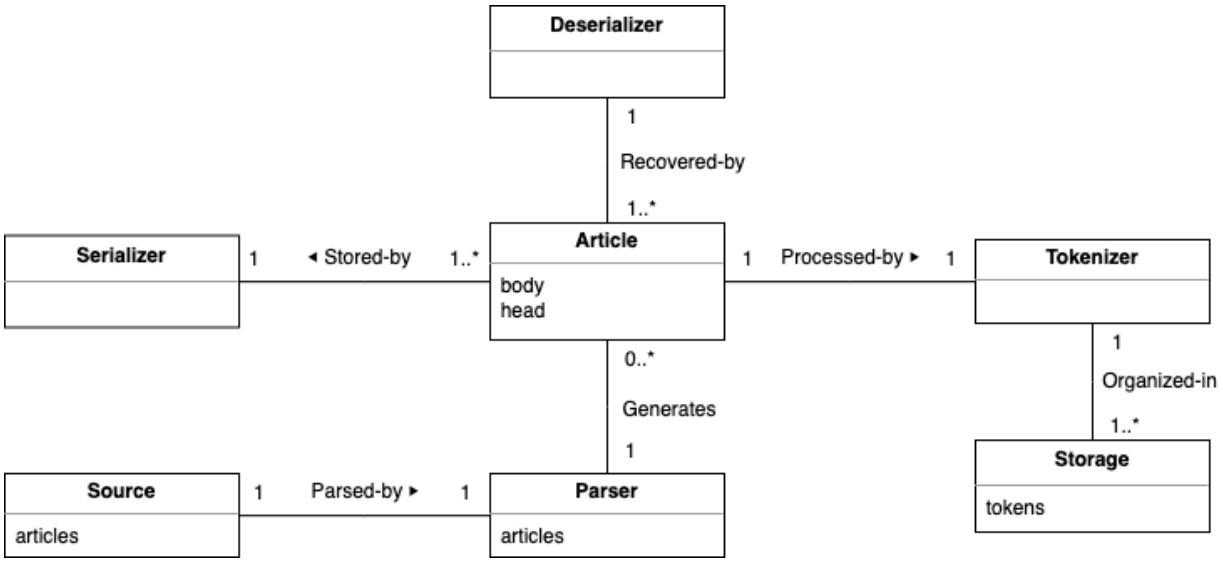


UC3:

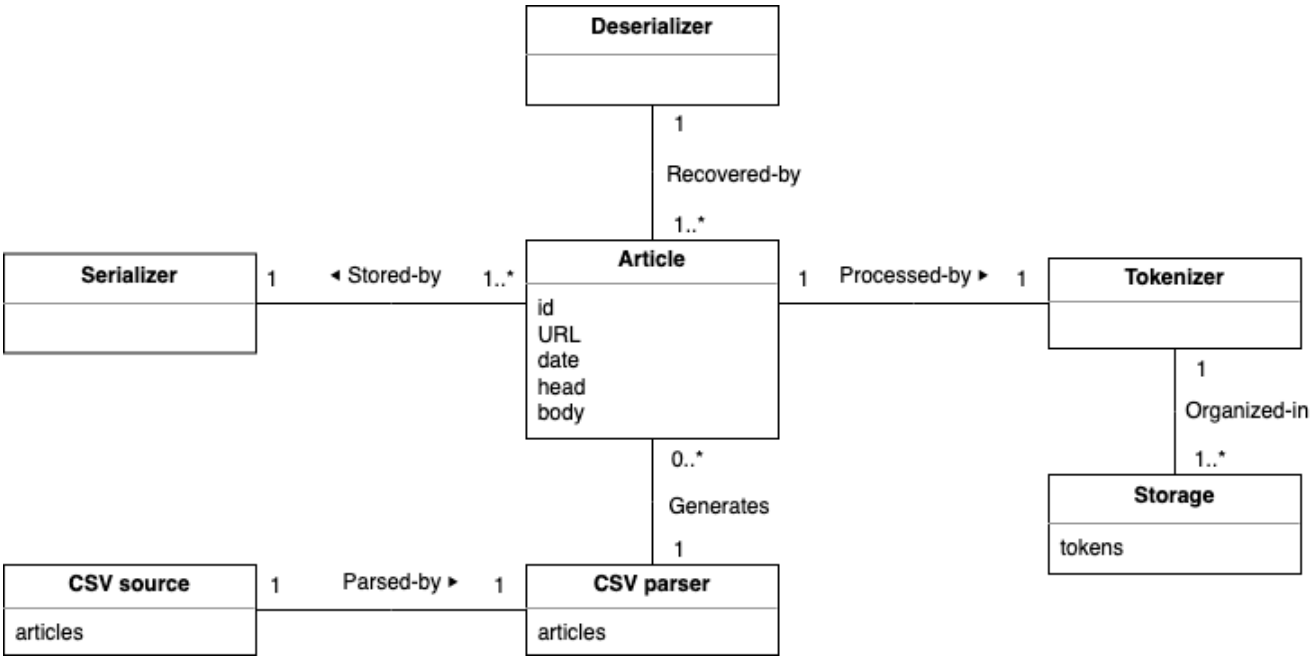


7.2 Modello di dominio

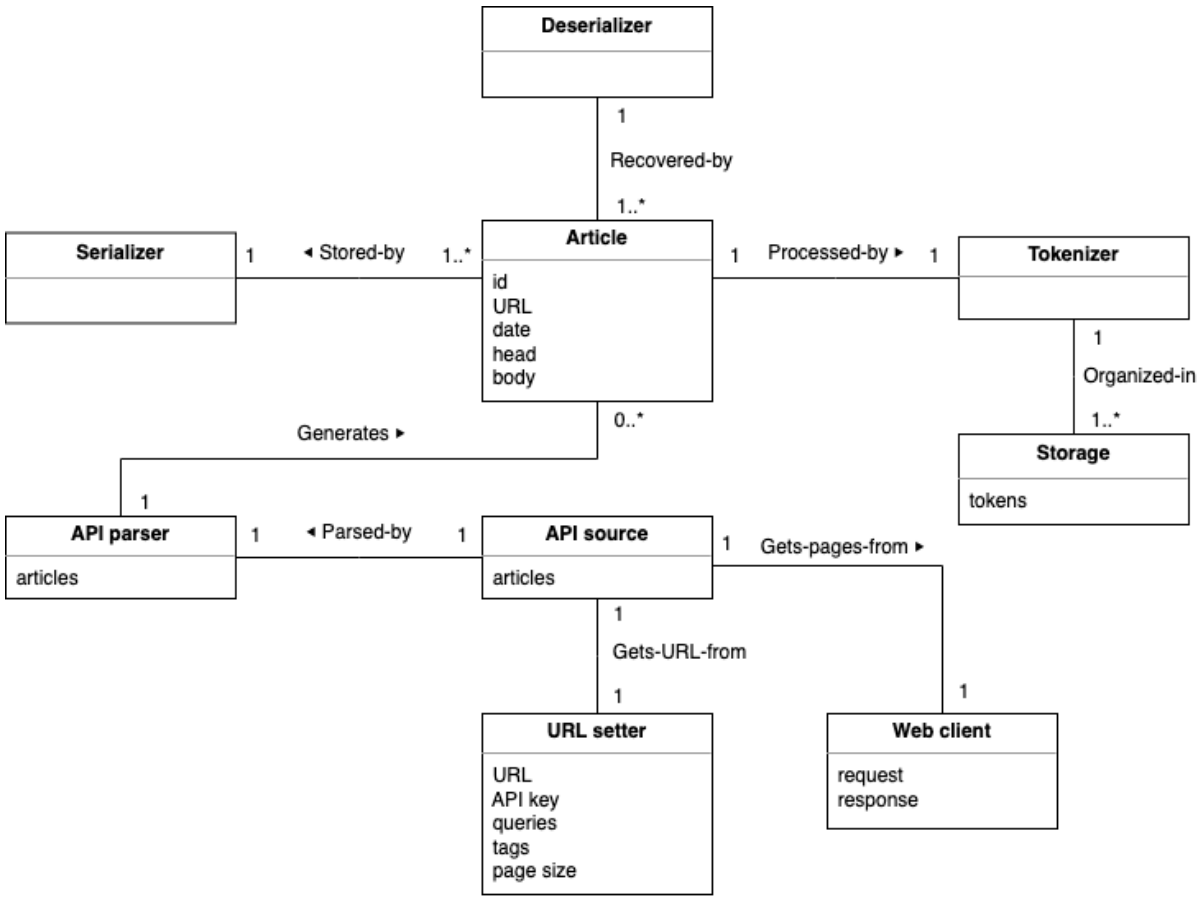
UC1:



UC2:

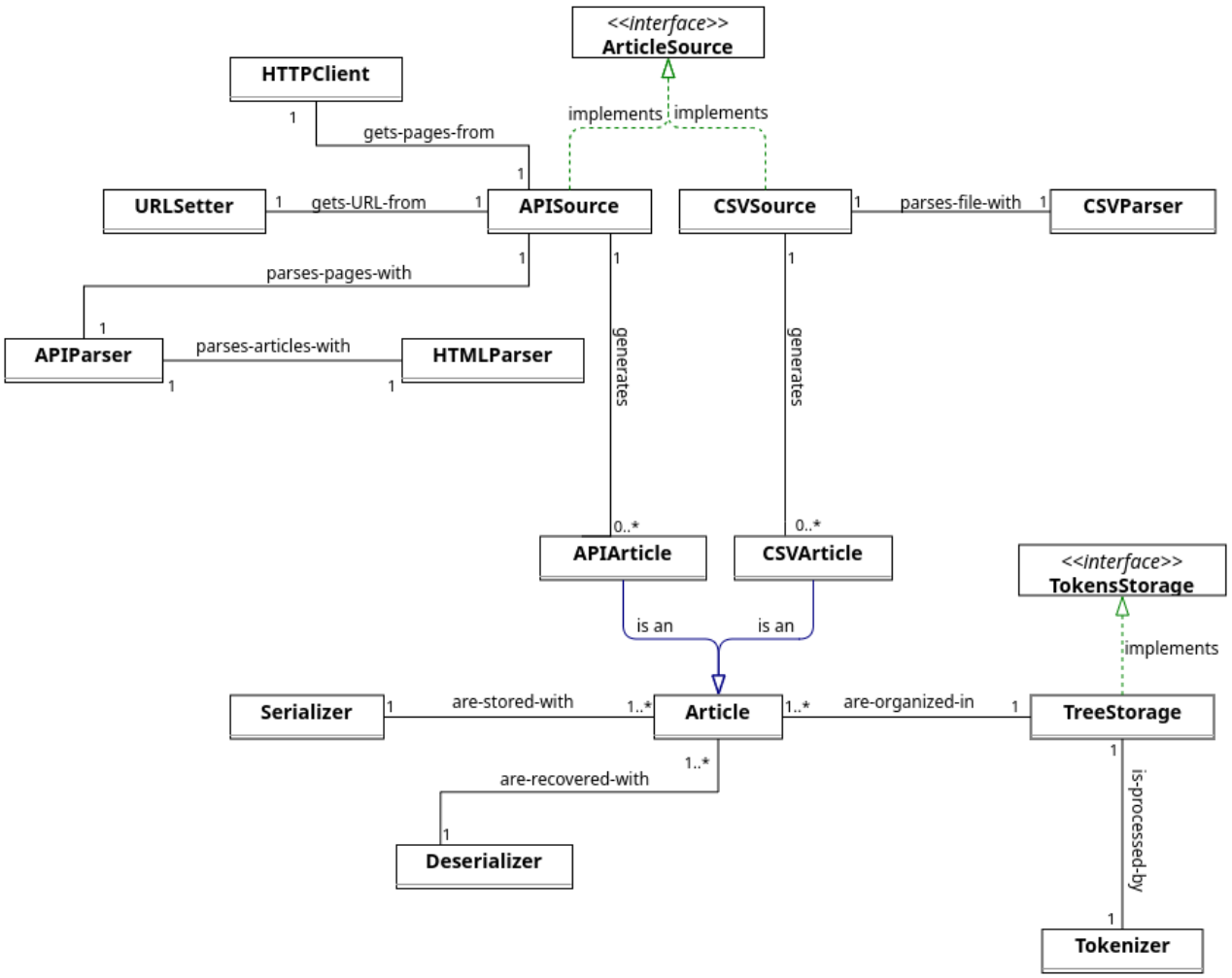

Deserializer
1
Recovered-by
1..*
Serializer
1
Stored-by
1..*
Article
id
URL
date
head
body
1
Processed-by
1
Tokenizer
1
Organized-in
1..*
Storage
tokens
0..*
Generates
1
CSV source
articles
1
Parsed-by
1
CSV parser
articles

UC3:

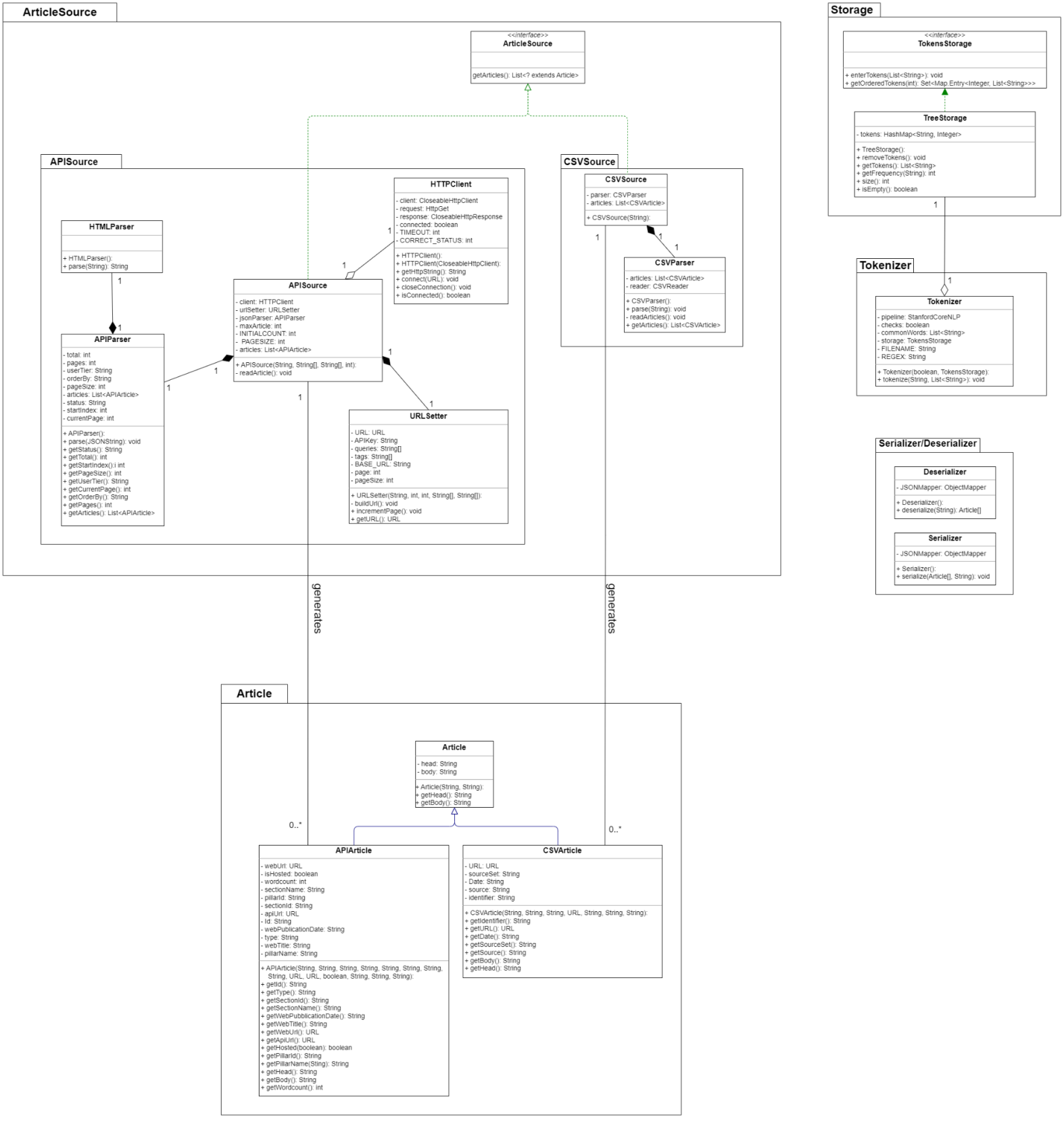

Deserializer
1
Recovered-by
1..*
Serializer
1
Stored-by
1..*
Article
id
URL
date
head
body
1
Processed-by
1
Tokenizer
1
Organized-in
1..*
Storage
tokens
Generates
0..*
1
API parser
articles
1
Parsed-by
1
API source
articles
1
Gets-pages-from
1
Gets-URL-from
1
1
URL setter
URL
API key
queries
tags
page size
1
Web client
request
response

7.3 Diagramma di classe

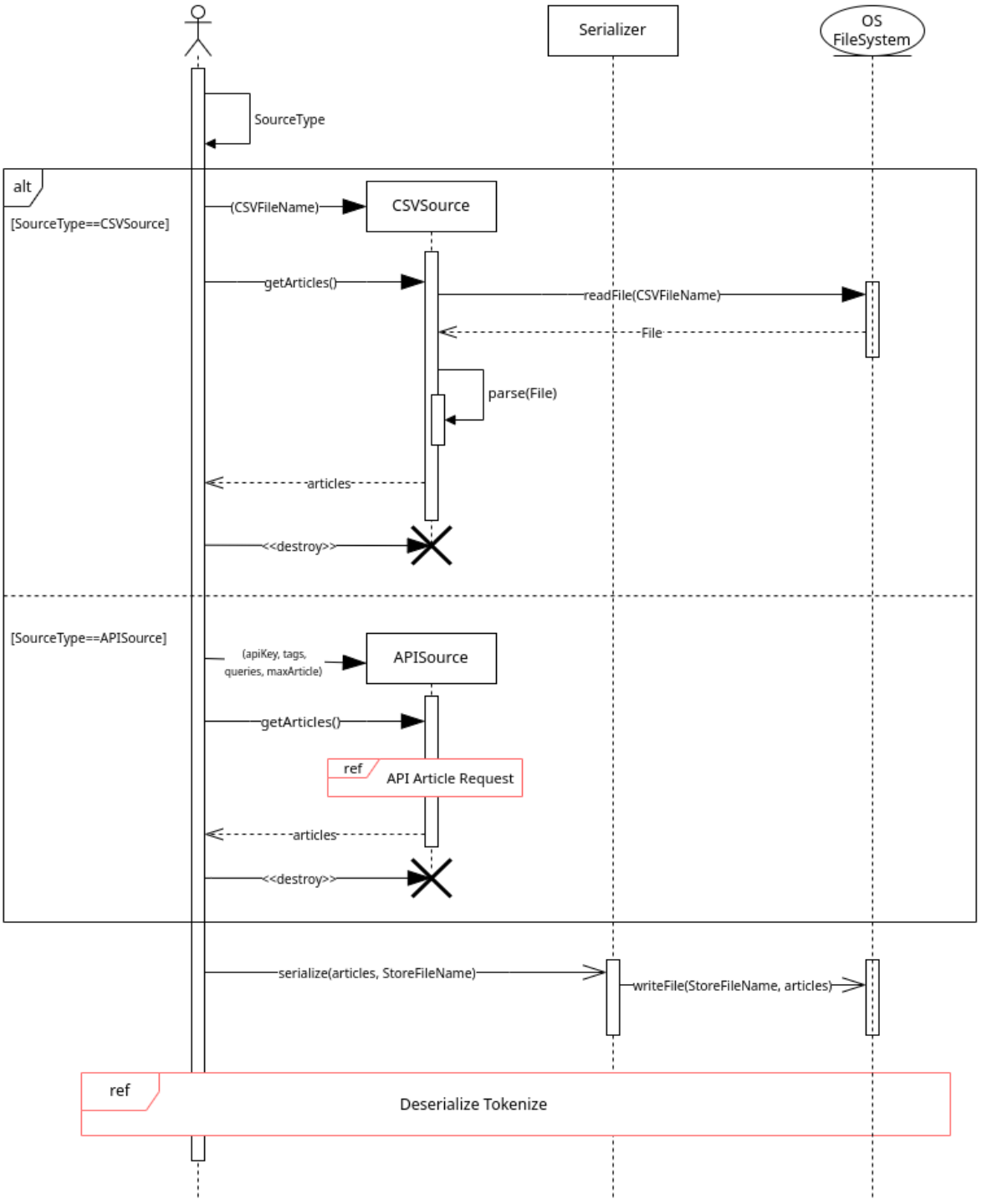
7.3.1 Diagramma di classe dinamico

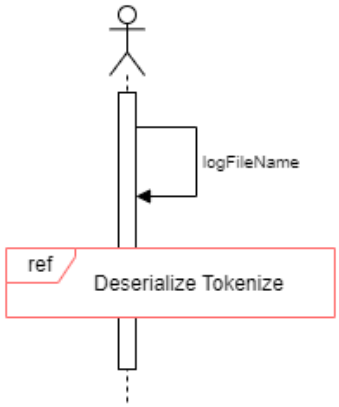


7.3.2 Diagramma di classe statico

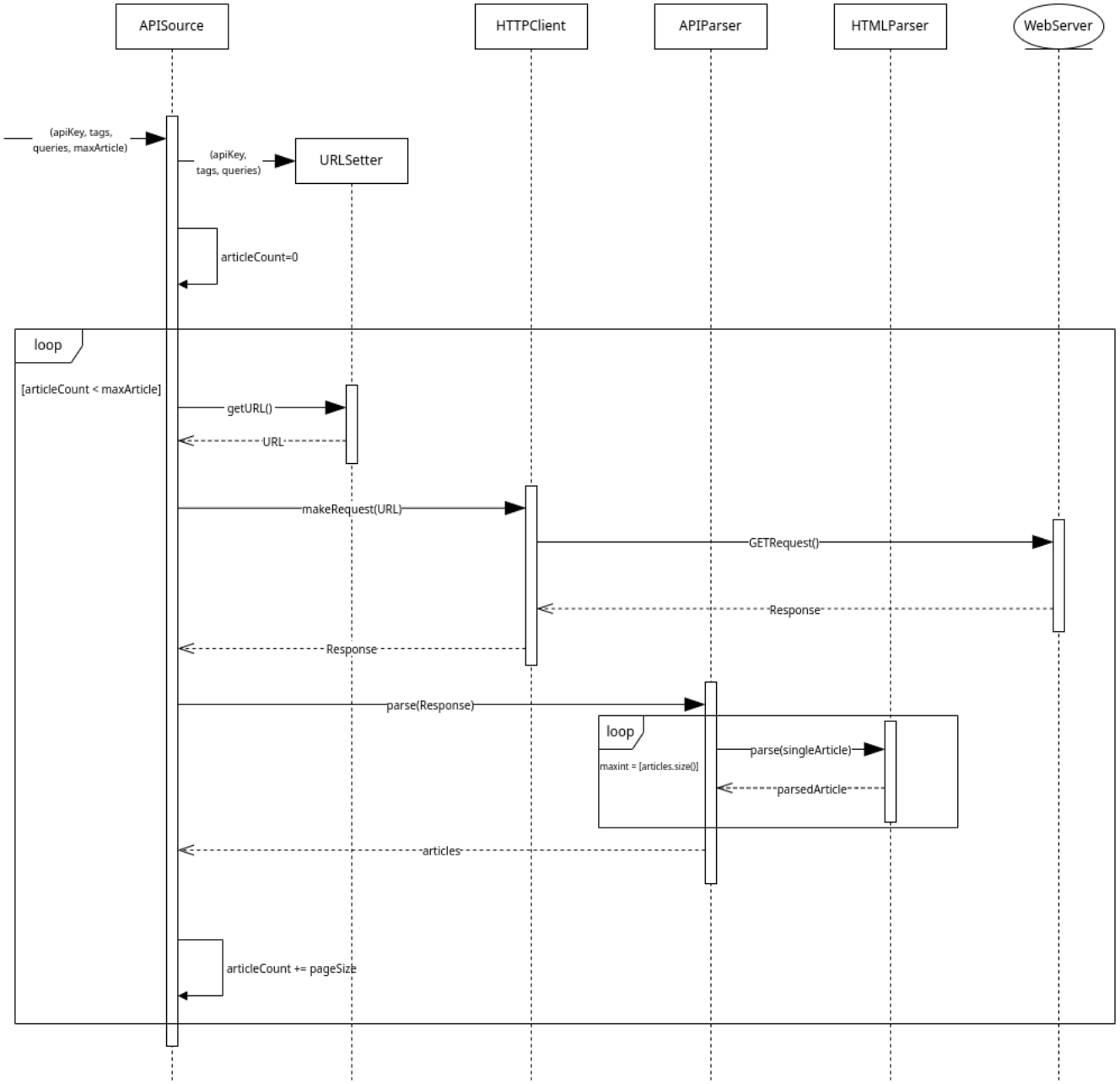


7.4 Diagramma di sequenza

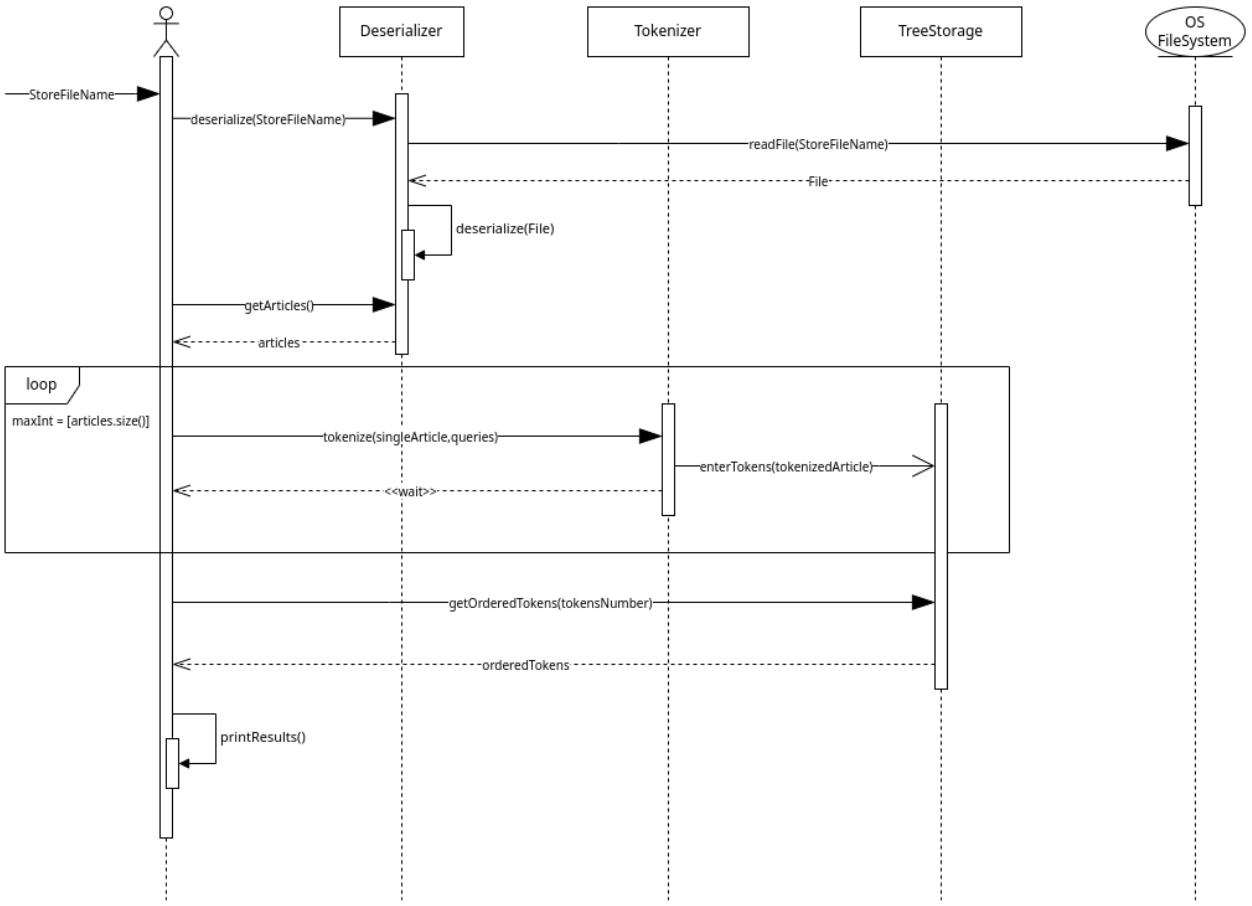




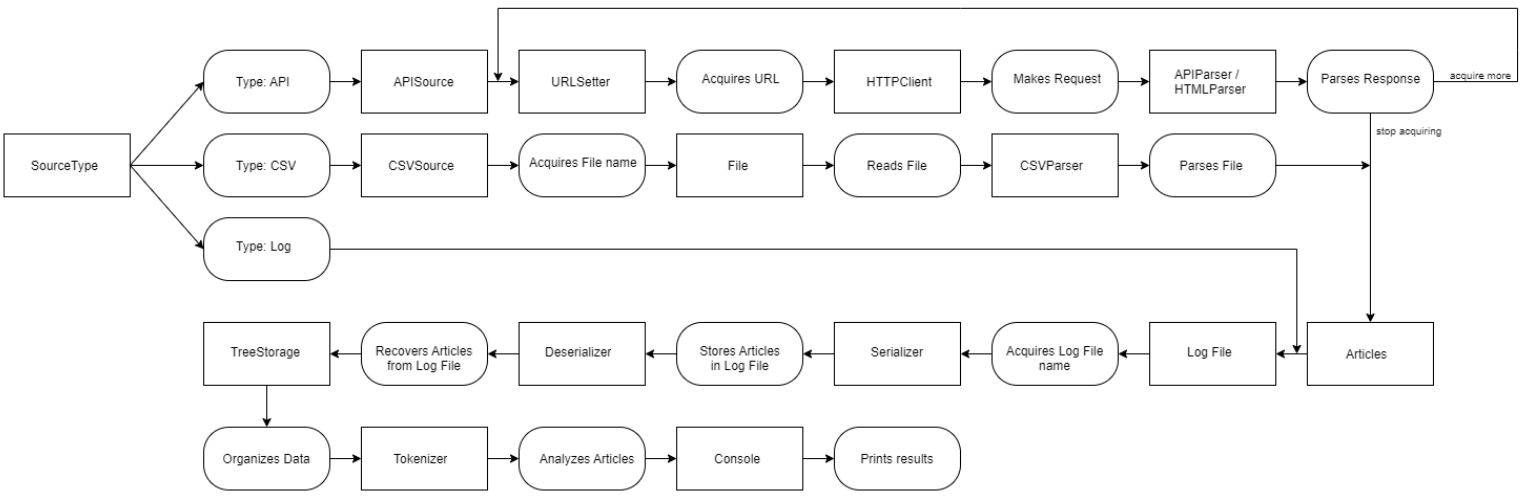
REF: API Article Request:



REF: Deserialize Tokenize:



7.5 Diagramma di attività



Indice