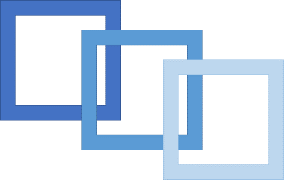
**Wordparser**

Andres Donadi, Christian Poli, Francesco Elia Rizzo, Bernardo Simonetto

Elementi di Ingegneria del Software, Ingegneria Informatica, Canale 1, UNIPD, a.a. 2022-23

**INDICE**

[Documentazione 2](#_Toc137583235)

[Introduzione 2](#_Toc137583236)

[Scenario 2](#_Toc137583237)

[Implementazione 5](#_Toc137583238)

# Documentazione

## Introduzione

Wordparser è un software scritto in Java che analizza e conta le parole in una raccolta di articoli scaricati da sorgenti online diverse.

## Scenario

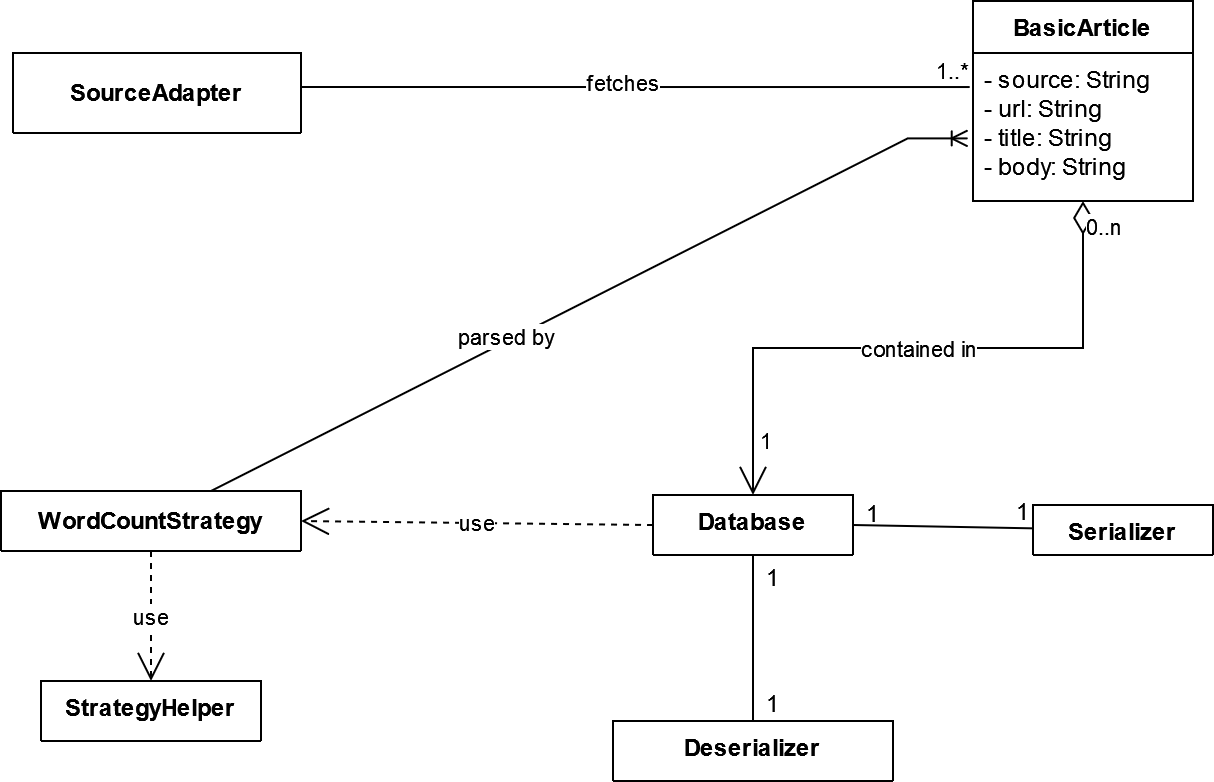
Wordparser risponde alla richiesta di individuare ed estrarre i termini più ricorrenti in una collezione di articoli provenienti dai siti Web differenti testate giornalistiche.

Una volta scaricati, gli articoli sono memorizzati in un file JSON che funge da base di dati; il formato JSON fornisce una struttura standard di codifica delle informazioni associate ciascun articolo. Gli oggetti di dominio sono:

* articolo;
* sorgente.

Articoli e sorgenti sono legati tramite un adattatore, che gestisce il reperimento degli articoli da ogni fonte; gli articoli sono memorizzati nel file JSON, la cui interfaccia è descritta da un oggetto che solo per semplicità si chiamerà *database*.

La conversione degli articoli da un formato a un altro (oggetto 🡪 JSON, JSON 🡪 oggetto) è gestita tramite oggetti fittizi (*Pure Fabrication*)che si sono definiti *serializzatore* e *deserializzatore*; la manipolazione degli articoli, ovvero l’estrazione e il conteggio dei termini, è invece in capo a un altro oggetto fittizio, designato come *WordCountStrategy*.



*Modello di dominio: serve a rappresentare gli oggetti del dominio, reali e fittizi, astraendo dall’aspettto funzionale e implementativo (oggetto del modello di progettazione, rappresentato tramite diagramma di classe)*

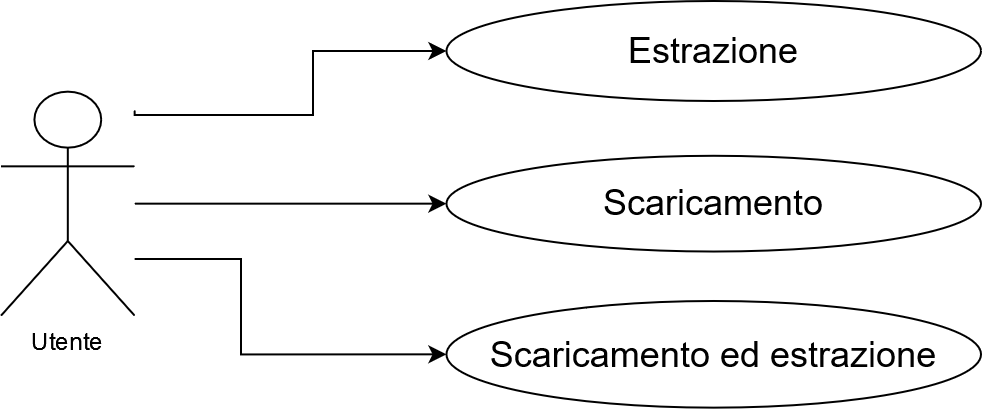
L’interfaccia utente consiste in tre comandi lanciati da terminale all’esecuzione del programma. L’utente accede agli articoli:

* in fase di scaricamento tramite il server sul quale sono ospitati;
* in fase di estrazione tramite il file JSON.

L’utente può effettuare nello stesso tempo le due operazioni invocando un apposito comando (v. Manuale). Le funzionalità di scaricamento ed analisi sono parte dell’interfaccia di accesso alla base di dati.

I casi d’uso sono:

* **scaricamento**:
  + **attori**: utente;
  + **descrizione**: l’utente richiede una raccolta di articoli a un server o a una risorsa locale;
  + **dati**: raccolta di articoli;
  + **stimolo**: comando inviato dall’utente tramite terminale;
  + **risposta**: invio della raccolta di articoli oppure fallimento dell’operazione;
  + **commenti**: l’operazione fallisce se le sorgenti non sono reperibili al percorso indicato;
* **estrazione**:
  + **attori**: utente;
  + **descrizione**: l’utente richiede al software di analizzare gli articoli scaricati e di contarne le parole in modo da individuare le 50 più ricorrenti;
  + **dati**: titolo e testo degli articoli, parole e frequenza delle parole;
  + **stimolo**: comando inviato dall’utente tramite terminale;
  + **risposta**: costruzione di un file di testo contenente una mappa <String, Integer> dove la chiave è la parola e il valore è la sua frequenza, oppure fallimento dell’operazione;
  + **commenti**: l’operazione fallisce se gli articoli non sono stati scaricati;
* **scaricamento** ed **estrazione**: combina i casi d’uso precedenti.



*Diagramma dei casi d’uso*

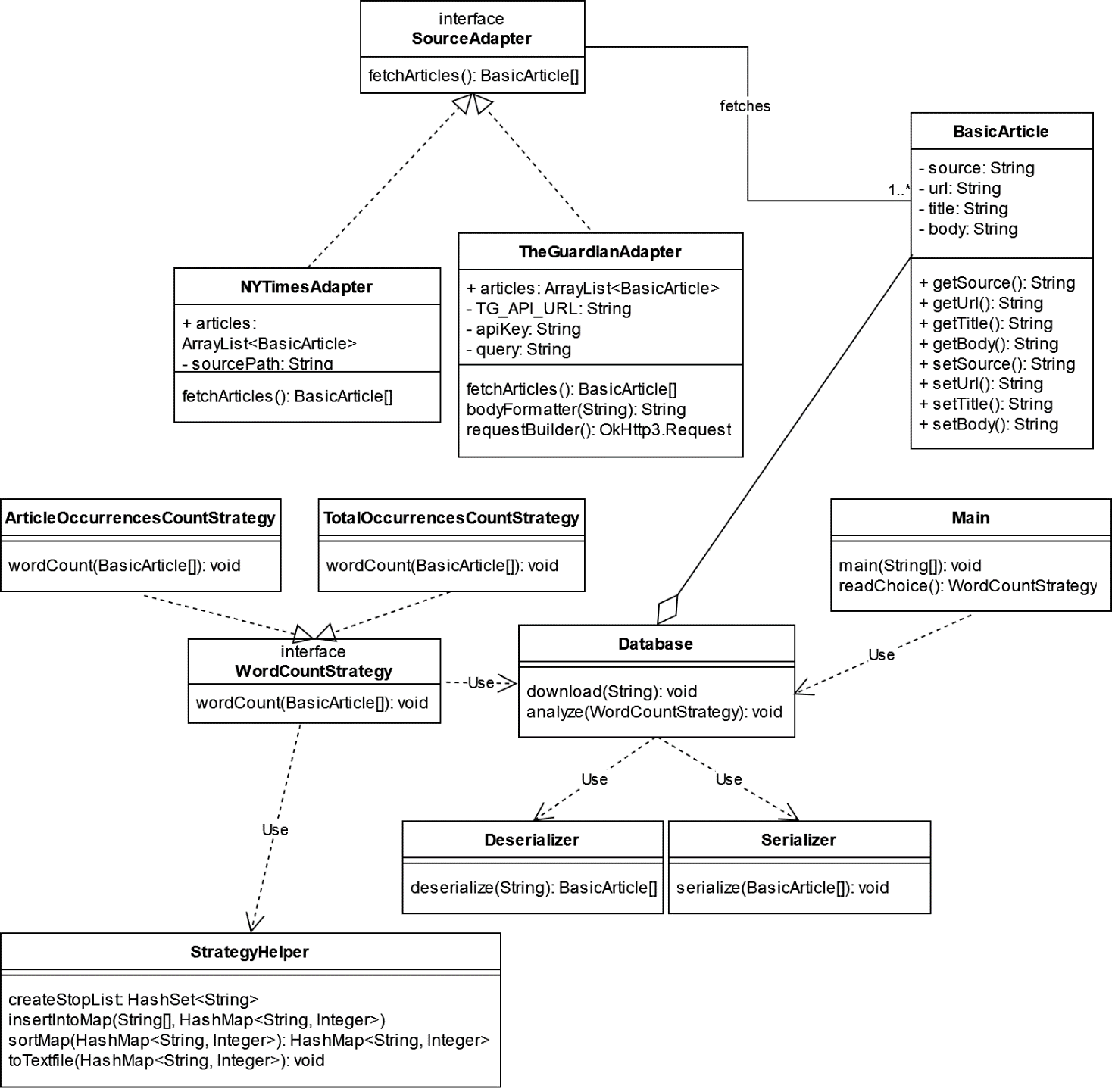
…

## Implementazione

Wordparser è stato sviluppato in linguaggio Java, in particolare usando le API di Java SE 8.

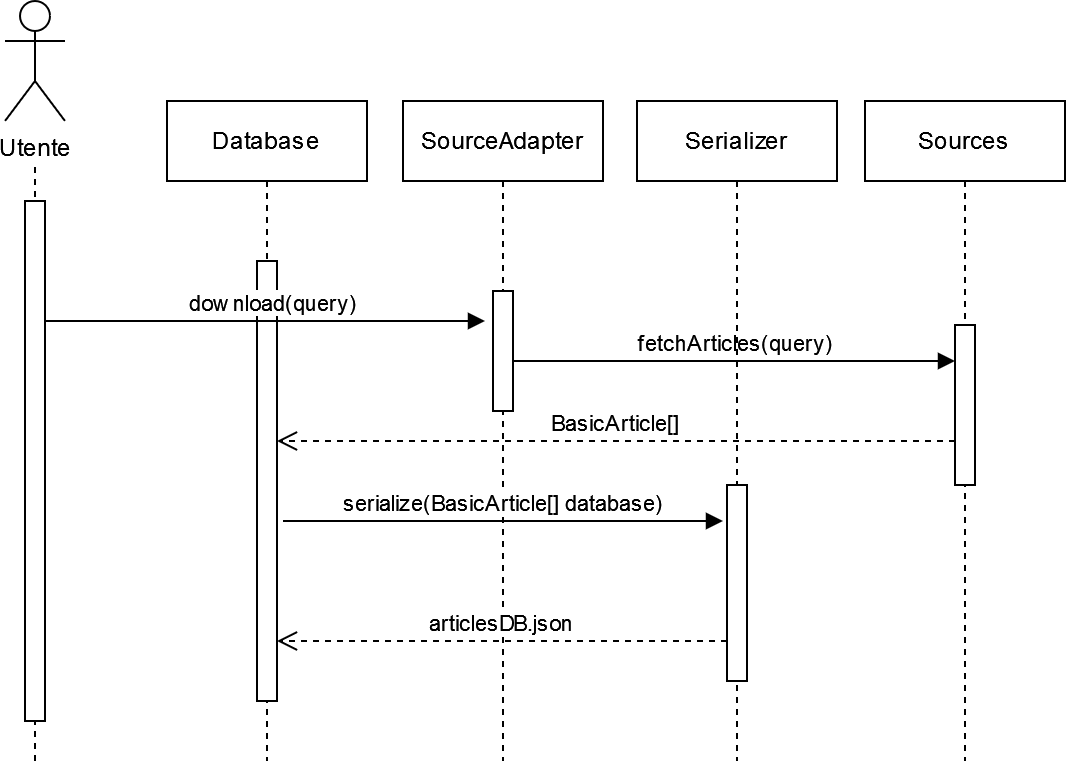
Il progetto si articola in 9 classi e 2 interfacce, di cui la classe denominata Main eroga l’interfaccia utente, mentre le rimanenti forniscono i diversi servizi, ovvero:

* **interazione con le fonti**: è garantita dall’interfaccia SourceAdapter, che, idealmente implementando il *pattern Adapter*, è realizzata dalle classi:
  + NYTimesAdapter: serve ad accedere a un file CSV preformato contenente una selezione di 1000 articoli della testata americana The New York Times estratti utilizzando come termini di ricerca *nuclear power*;
  + TheGuardianAdapter: fornisce l’interfaccia con i server della testata britannica The Guardian; specificata una chiave di accesso per l’autenticazione apiKey e un termine di ricerca query, attraverso una richiesta HTTPS viene restituito un file JSON contenente 200 articoli per ogni chiamata all’*endpoint* di ricerca (<http://content.guardianapis.com/search>);
* **gestione della persistenza**: è garantita dalla classe Database, che si appoggia alle classi:
  + **Serializer**: è costituita dal solo metodo serialize(BasicArticle[] articles), che serve a serializzare un array di articoli, ovvero a mapparlo in un file (in particolare in un file JSON);
  + **Deserializer**: è costituita dal solo metodo deserialize(String databasePath), che

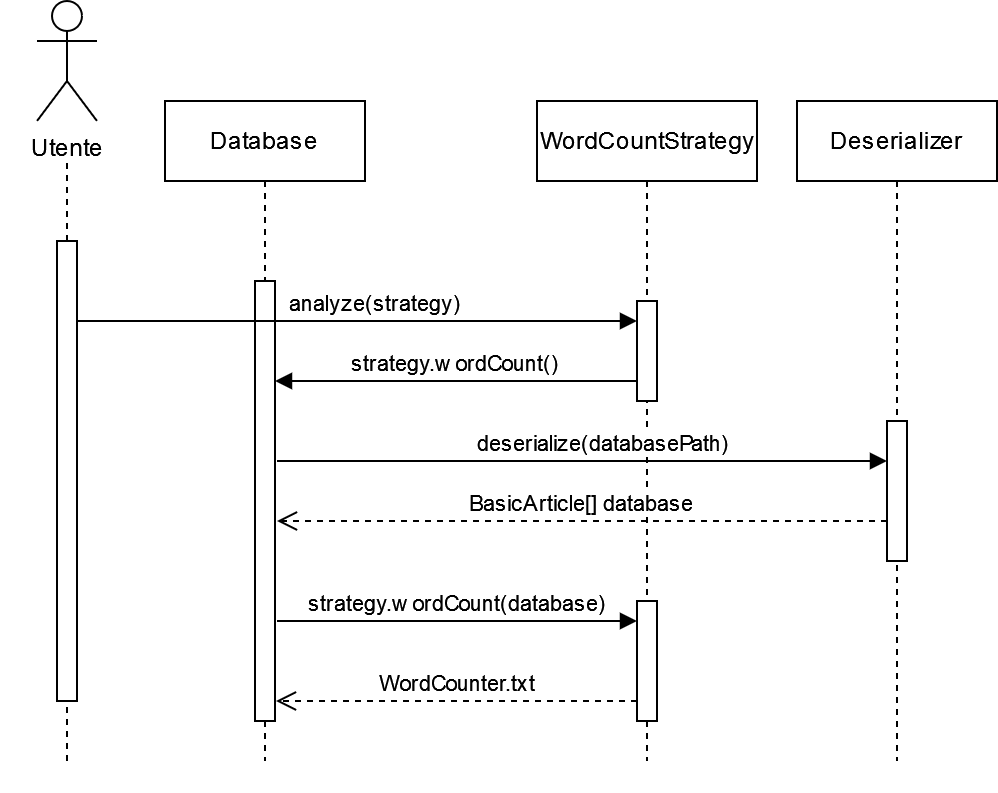


I casi d’uso sono:

* **scaricamento**: l’utente lancia il comando di scaricamento all’adattatore per le sorgenti, che invia una richiesta di accesso alle fonti a cui segue l’invio di una collezione di articoli, poi serializzata in un file JSON;



* **estrazione**: l’utente lancia il comando di analisi degli articoli all’apposita componente, che dopo aver deserializzato il file JSON conta le parole, secondo un criterio opportunamente specificato al momento della richiesta, e restituisce all’utente un file di testo contenente una mappa delle 50 parole più ricorrenti con la relativa frequenza;



* **scaricamento ed estrazione**: combina i due casi d’uso precedenti.

