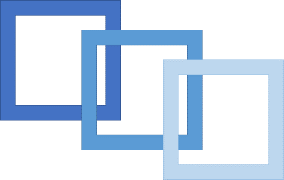
**Wordparser**

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](file:///D:\EIS_ProjectDescription.docx#_Toc137671960)

[Introduzione 2](file:///D:\EIS_ProjectDescription.docx#_Toc137671961)

[Scenario 2](file:///D:\EIS_ProjectDescription.docx#_Toc137671962)

[Implementazione 5](file:///D:\EIS_ProjectDescription.docx#_Toc137671963)

[Manuale 9](file:///D:\EIS_ProjectDescription.docx#_Toc137671964)

[Dipendenze 9](file:///D:\EIS_ProjectDescription.docx#_Toc137671965)

[Istruzioni per installare ed eseguire 10](file:///D:\EIS_ProjectDescription.docx#_Toc137671966)

# 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 di differenti testate giornalistiche.

Gli articoli sono modellati come un tipo BasicArticle caratterizzato dai seguenti campi (tutti privati):

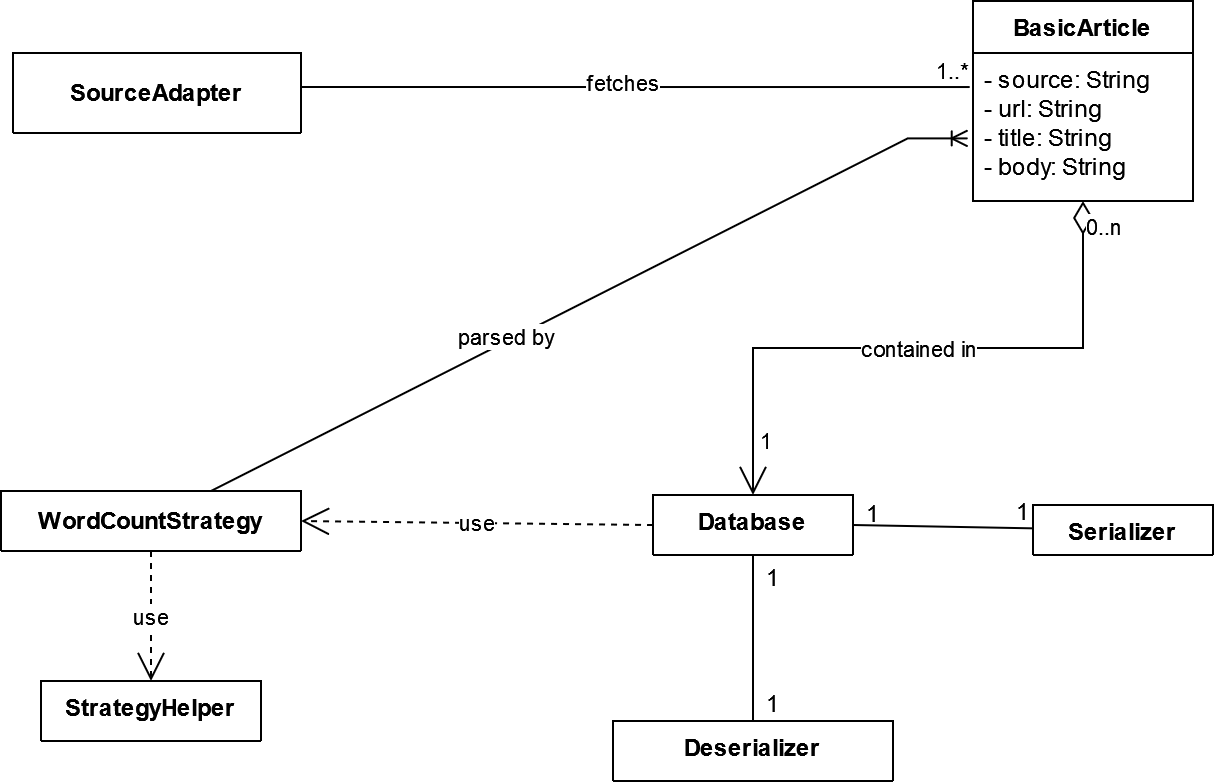
* source: fonte;
* url: link all’articolo;
* title: titolo;
* body: corpo,

e dai relativi metodi di accesso e di modifica.

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. 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 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*.

Il prodotto del software è un file di testo (.txt) che contiene una mappa <String, Integer>, ovvero una collezione di coppie chiave-valore ove la chiave è la parola e il valore è la sua frequenza.



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

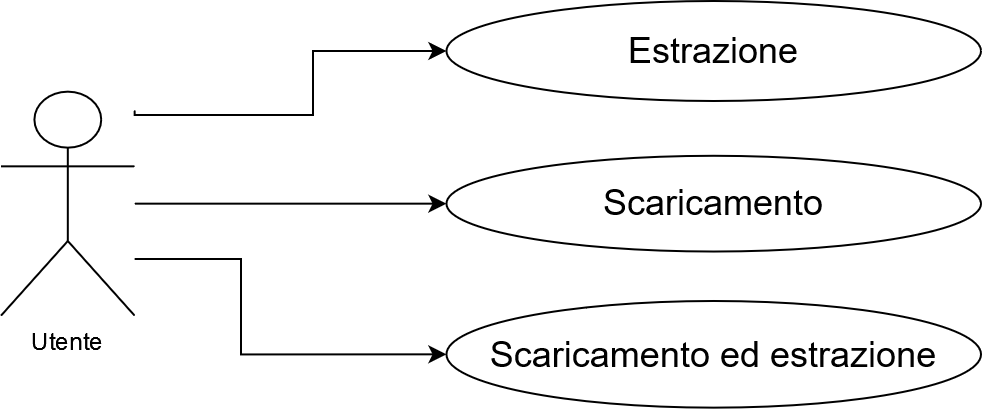
L’interfaccia utente consiste in tre comandi lanciati da terminale all’esecuzione del programma. L’utente accede agli articoli:

* in fase di scaricamento tramite il server sul quale sono ospitati oppure tramite un file memorizzato in locale;
* 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 la mappa di coppie parola-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

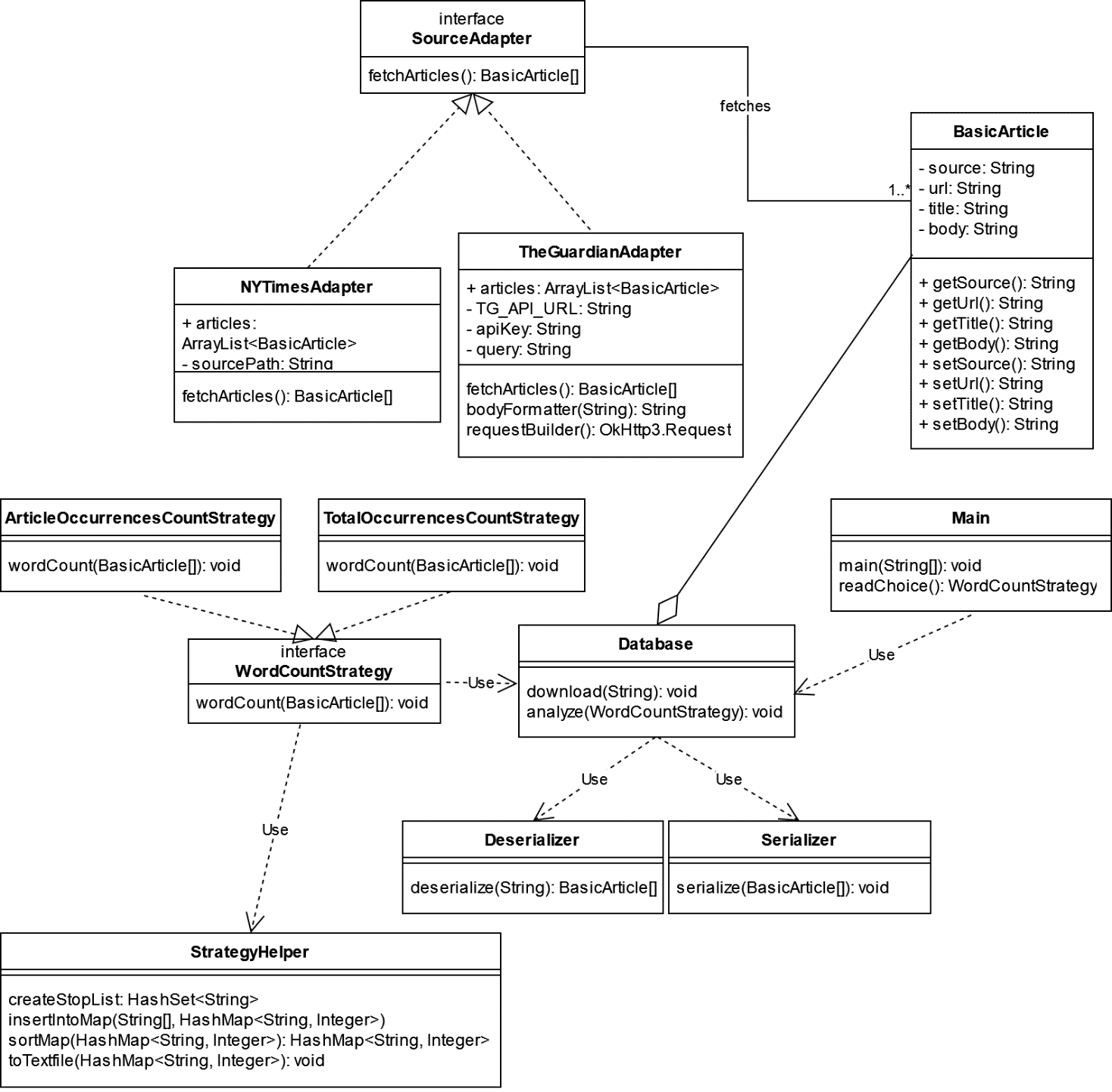
Wordparser è stato sviluppato in linguaggio Java, in particolare usando le API di Java SE 8.

Il progetto si articola in 9 classi e 2 interfacce, di cui la classe denominata Main eroga l’interfaccia utente, la classe BasicArticle modella i singoli articoli, 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: processa 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 apiKey e un termine di ricerca query, attraverso una richiesta HTTP 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 serve a deserializzare un file JSON contenente una raccolta di articoli, ovvero a mappare gli attributi JSON di ogni articolo in una cella di un array di articoli;
* **manipolazione degli articoli**: è garantita dall’interfaccia WordCountStrategy che, idealmente implementando il *pattern Strategy*, è realizzata dalle classi:
  + ArticleOccurrencesCountStrategy: implementa l’interfaccia in modo che siano enumerate tutte le occorrenze di ogni parola, escluse eventuali ripetizioni all’interno di uno stesso articolo;
  + TotalOccurrencesCountStrategy: implementa l’interfaccia in modo che siano enumerate tutte le occorrenze di ogni parola, comprese eventuali ripetizioni all’interno di uno stesso articolo,

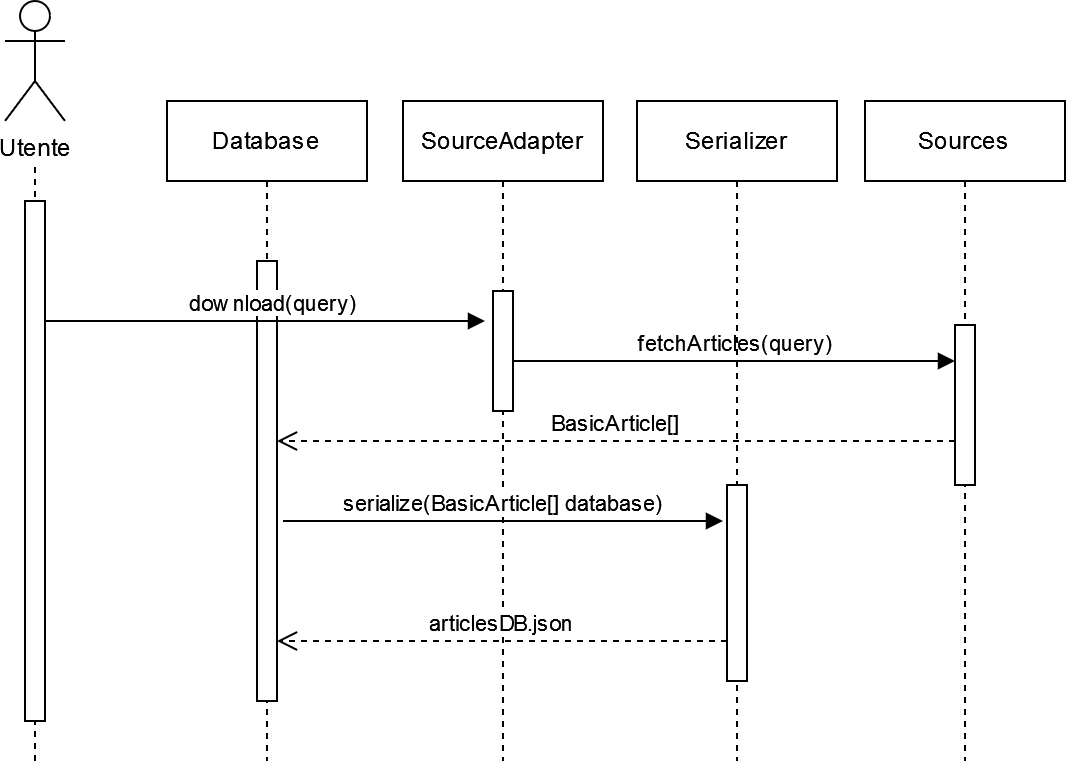
che dipendono dalla classe StrategyHelper per le seguenti operazioni:

* + **creazione di una lista di *stop words***: con *stop words* si intende un insieme di parole da escludere dall’analisi di un testo;
  + **inserimento delle coppie parola-frequenza nella mappa**;
  + **ordinamento della mappa** per valore e, in caso di pareggi, in ordine alfabetico di chiave;
  + **esportazione della mappa** su file di testo (.txt).

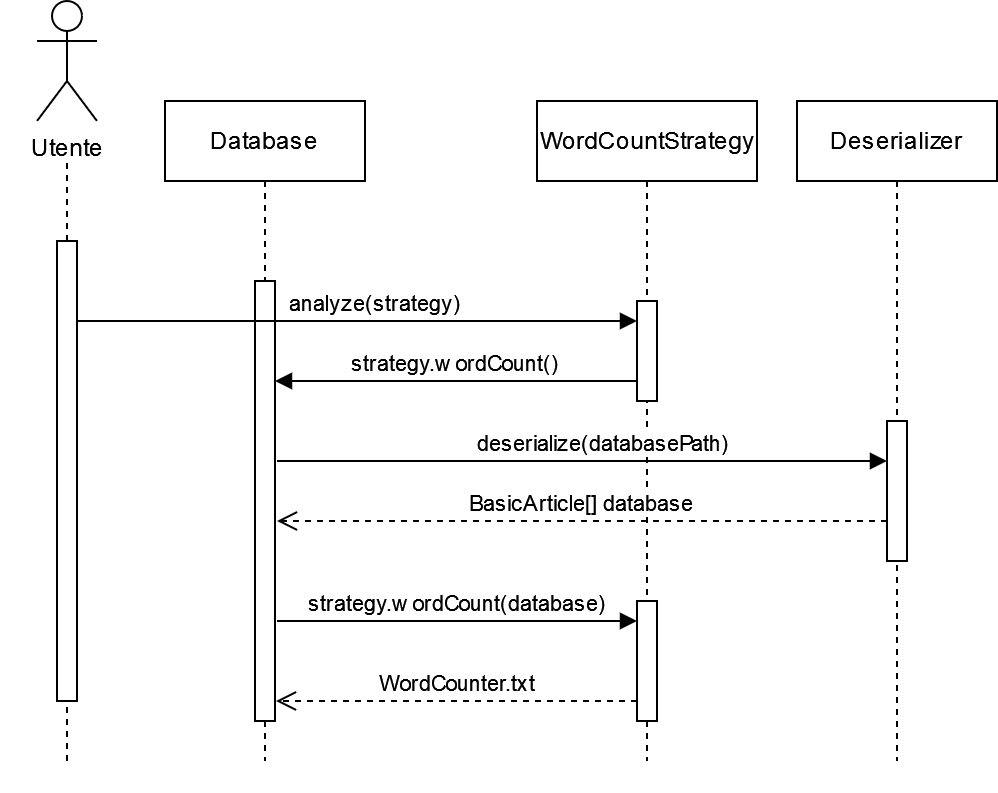


A livello di interazioni, i casi d’uso sono:

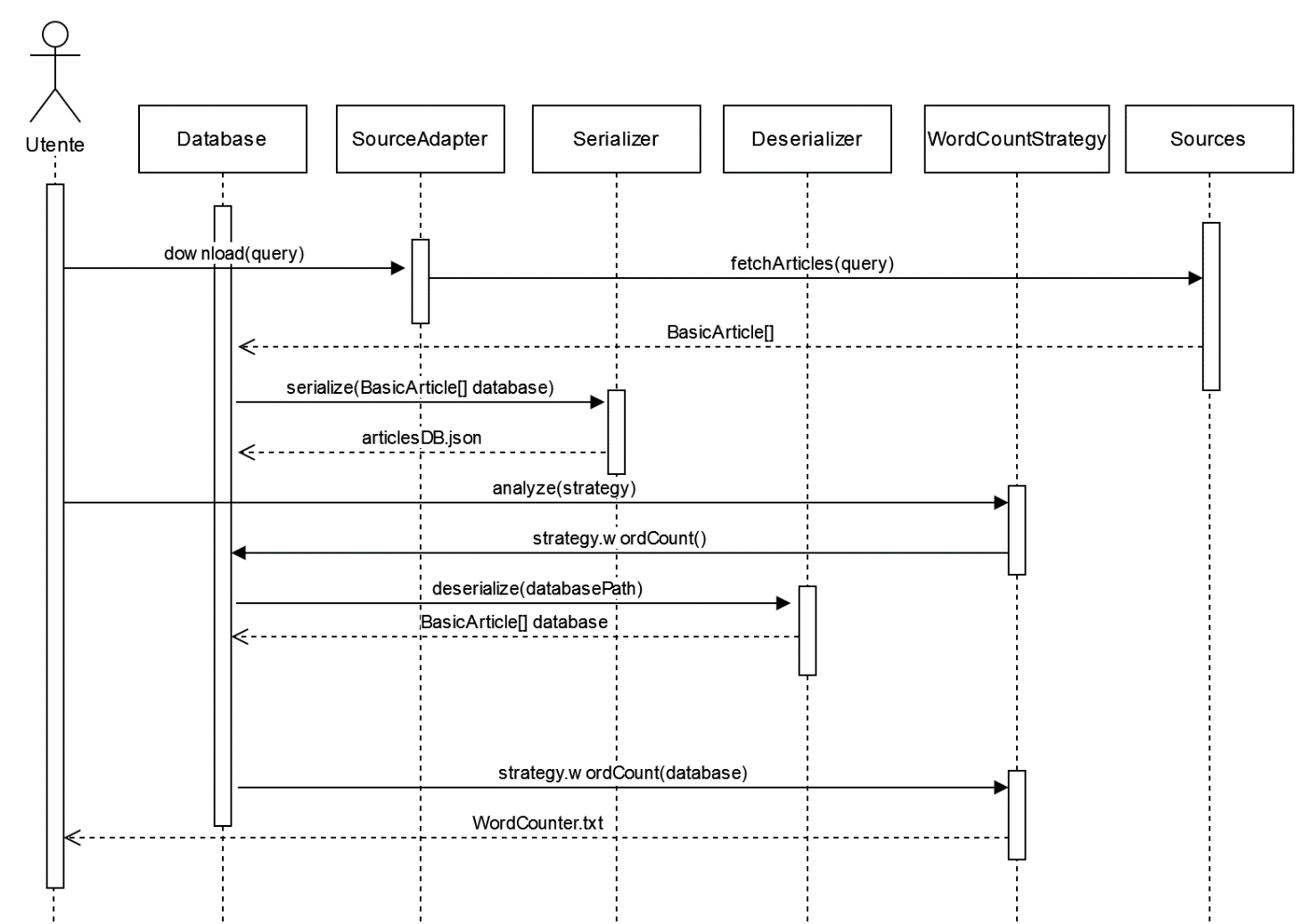
* **scaricamento**: l’utente lancia il comando di scaricamento, specificando un termine di ricerca query, all’adattatore per le sorgenti, che invia una richiesta di accesso alle fonti a cui segue l’invio di una collezione di articoli, poi serializzata in un file JSON;



* **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 strategy 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.



# Manuale

## Dipendenze

Le dipendenze, riportate nel sito costruito mediante il comando mvn site alla voce Project Information > Dependencies, sono:

* FasterXML Jackson 2.15.1: libreria Java per la manipolazione di file JSON che comprende anche strumenti per la gestione di file in altri formati semistrutturati quali CSV o XML;
* System Lambda 1.2.1: libreria di funzioni per collaudare codice che dipende da java.lang.System;
* SquareUp OkHttp 4.11.0: client HTTP in linguaggio Java;
* Apache Commons:
  + Apache Commons CLI 1.5.0: libreria per la costruzione e la gestione di interfacce utente da riga di comando;
  + Apache Commons Lang 3.12.0: libreria per la manipolazione delle classi base di Java;
* Jsoup 1.16: libreria per l’elaborazione e la manipolazione di documenti HTML.

## Istruzioni per installare ed eseguire

Il progetto consegnato su Moodle è fornito con dati di ingresso già pronti per l’esecuzione. Si precisa che il progetto è trasmesso con una chiave di accesso all’API del Guardian pre-impostata: per registrarne una nuova, occorre registrarsi all’indirizzo <https://open-platform.theguardian.com/access/>, quindi inserire la chiave così ottenuta nel file *guardian.properties*, accessibile al seguente percorso:

./src/main/resources/guardian.properties

Per compilare il progetto e generare il file .jar, utilizzare Maven, disponibile al seguente indirizzo:

<https://maven.apache.org/download.cgi>

Installato Maven nella cartella di elezione, verificata la corretta installazione lanciando da riga di comando:

mvn --version

effettuare l’installazione di Wordparser tramite il comando:

mvn package

Spostare dunque nella cartella principale del progetto il file .jar, necessario per garantire il corretto accesso alla cartella *assets* contenente sorgenti e lista di *stop words*, lanciando il comando:

mv ./target/wordparser-1.0-jar-with-dependencies.jar ./

Infine, per eseguire l’applicazione, usare il comando:

java -jar wordparser-1.0-jar-with-dependencies.jar -{d, e, de, h} <download query>

scegliendo uno solo tra i comandi -{d, de, e, h}; tali comandi sono:

* d, --download-articles <query>: scarica articoli contenenti il termine specificato nel campo <query> da tutte le sorgenti disponibili;
* e, --e extract terms: estrae i termini dalla raccolta di articoli in formato JSON;
* de, --download-and-extract <query>: è una combinazione dei comandi -d ed -e;
* h, --help: stampa a schermo una guida all’uso.

Per maggiori informazioni, invocare:

java -jar wordparser-1.0-jar-with-dependencies.jar -h

Per accedere alla documentazione Maven / JavaDocs del progetto, invocare:

mvn site