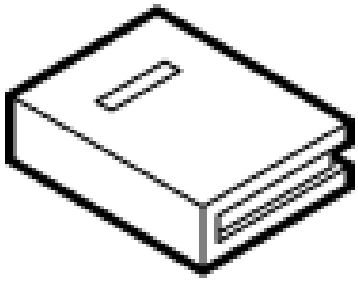

# UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN
**INGEGNERIA INFORMATICA**

**CORSO DI ELEMENTI DI INGEGNERIA DEL SOFTWARE**

# Manuale di progetto



**Gruppo di lavoro:**
Giovanni Faedo
Martino Bissiato
Marco Facco
Andrea Maluta

**Anno accademico:**
2023/2024

# Indice

# Capitolo 1

# Documentazione

## 1.1 Descrizione

Il programma che è stato sviluppato ha come obiettivo quello di analizzare degli articoli estratti da una raccolta di articoli online e non, riconoscerne le i termini più frequenti e ritornare il numero di volte che esse appaiono. Il linguaggio utilizzato è Java.

## 1.2 Scenario

Gli articoli vengono modellati con un oggetto della classe Article composto dai seguenti campi che sono tutti definiti come private:

- *title*: titolo dell'articolo
- *body*: testo dell'articolo
- *source*: fonte da cui viene estratto

All'interno della classe sono presenti le definizioni e i vari metodi di accesso e modifica dei campi.

Gli articoli vengono scaricati e successivamente vengono memorizzati in un file JSON che funziona in maniera similare ad una base di dati. Il formato JSON fornisce una struttura standard di codifica delle informazioni associate a ciascun articolo. Il download degli articoli, il salvataggio sul file JSON e la successiva elaborazione è affidata a una classe *Downloader*. Le varie conversioni degli articoli sono affidate a due oggetti di nome *Serializer* e *Deserializer*.

- Serializer: classe che effettua la conversione da oggetto Java a file JSON. Nel nostro caso converte l'oggetto Article in un file JSON.

- Deserializer: classe che effettua la conversione da file JSON (che deve già essere salvato e reperibile) a oggetto Java. Nel nostro caso serve a convertire da JSON a Article per poter conteggiare i termini.

Un ulteriore oggetto è il *WordCounter* a cui è affidato il compito di estrarre e conteggiare i termini più frequenti all'interno degli articoli.

L'output del programma è un file testuale di nome *wordscount.txt*, composto da una mappa (<String, Integer>) descritta da una coppia chiave-valore dove la chiave è la parola e il valore è il suo peso, ovvero il numero di articoli in cui compare.
Le classi appena menzionate sono descritte in Figura 1.1 nel modello di dominio, che rappresenta gli oggetti del dominio (reali e fittizi) e ne descrive le loro interazioni tralasciando l'aspetto funzionale e implementativo.

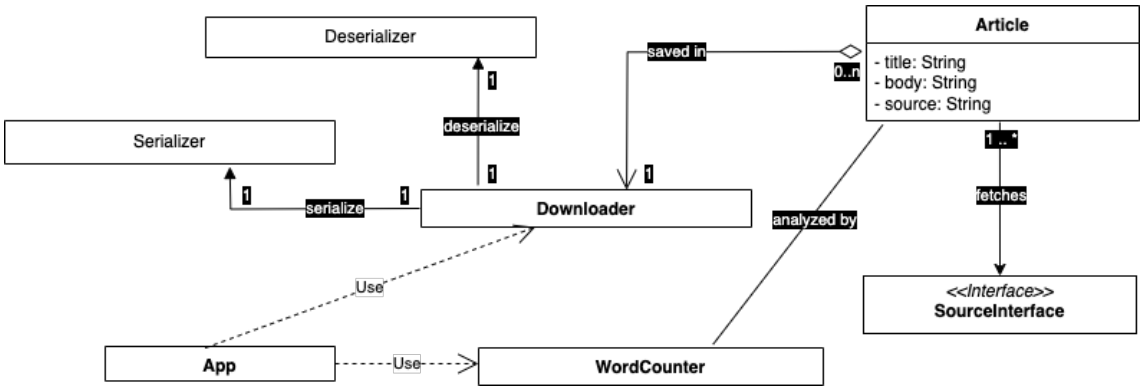



Figura 1.1: Modello di dominio

L'interfaccia utente consiste in tre comandi lanciati da terminale, con l'aggiunta di alcune opzioni nel caso dell'operazione di scaricamento.

L'utente accede agli articoli in fase di scaricamento tramite il server su cui sono stati memorizzati o tramite file locale, in fase di estrazione attraverso il file JSON oppure entrambe tramite un comando apposito.

### 1.2.1 Casi d'uso

I casi d'uso sono i seguenti:

- **Scaricamento**
  - **Attori**: Utente
  - **Descrizione**: Richiesta di una raccolta di articoli a un server o in locale
  - **Dati**: raccolta di articoli
  - **Stimolo**: Comando inviato dall'utente tramite il terminale
  - **Risposta**: La raccolta di articoli oppure fallimento dell'operazione

  - **Commento dell' errore**: L'operazione fallisce se non sono stati caricati precedentemente degli articoli (nel caso di CSV) oppure se non si riesce ad accedervi (nel caso TheGuardian)

- **Estrazione**
  - **Attori**: Utente
  - **Descrizione**: L'utente richiede al programma di analizzare gli articoli scaricati e estrarre i 50 termini con peso maggiore
  - **Dati**: Titolo e testo degli articoli e parole (termine e peso relativo)
  - **Stimolo**: Comando inviato dall'utente da riga di comando
  - **Risposta**: wordscount.txt contentente una mappa delle coppie termine - peso oppure un errore di fallimento dell'operazione
  - **Commento dell'errore**: L'operazione fallisce se non sono stati scaricati precedentemente degli articoli
- **Scaricamento ed estrazione**: combina i due casi d'uso precedenti

In fase di scaricamento l'utente avrà la possibilità di scegliere da che sorgente estrarre gli articoli, attraverso riga di comando.

Il diagramma dei casi d'uso viene mostrato in Figura 1.2

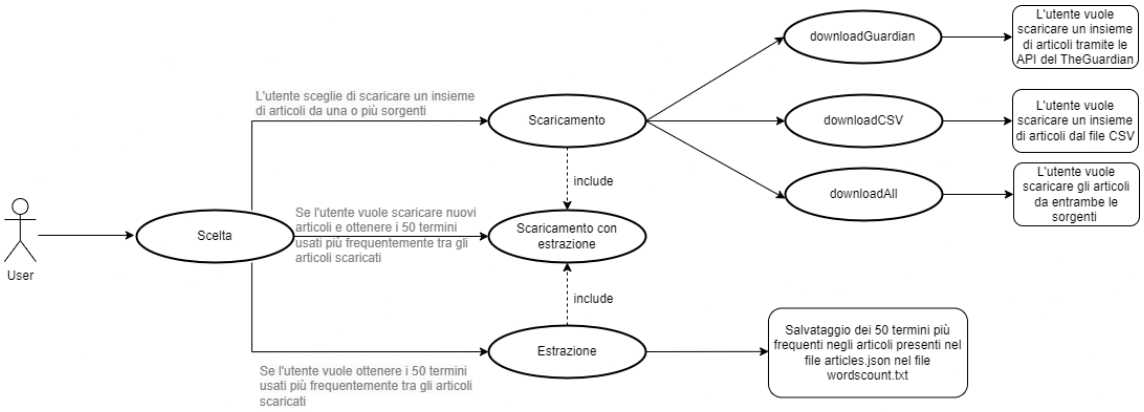



Figura 1.2: Diagramma dei casi d'uso

## 1.3 Implementazione

Il programma è stato sviluppato in linguaggio Java, utilizzando le API di Java SE 8. Il progetto si articola in 8 classi e 1 interfaccia. La classe App gestisce l'interfaccia utente mentre la classe Article modella i singoli articoli. Le altre classi forniscono altri servizi diversi.

- **Interazione con le sorgenti**: garantita e realizzata da due classi
  - **CSV**: processa una serie di file CSV, contenente una selezione di articoli della testata americana The New York Times. Per funzionare la classe preleva path del file CSV e crea un nuovo oggetto Article per salvare tutti gli articoli presenti nel file.
  - **TheGuardian**: fornisce l'interfaccia con i server della testata The Guardian. La classe fa una richiesta all'API TheGuardian, attraverso una chiave (*api key*) ottenuta dal sito del The Guardian e viene restituito un file JSON. Al file viene in seguito fatto un parsing in maniera tale da poter essere facilmente leggibile tramite funzione *setPrettyPrinting*.
- **Persistenza dei file**: gestita dalla classe *Downloader* che gestisce i download degli articoli e li salva in un file JSON. Si appoggia sulle classi:
  - **Serializer**: costituita dal solo metodo *Article_to_JSON* che converte un insieme di oggetti di tipo *Article(Article[] articles)* in un file di tipo JSON.
  - **Deserializer**: costituita dal solo metodo *JSON_to_Article(String path)* che prende una stringa JSON e fa la deserializzazione in una lista di oggetti di tipo *Article* e li stampa a video.
- **Manipolazione**: garantita dalla classe *WordCounter* che serve per ottenere i 50 termini più frequenti negli articoli insieme al loro peso. Per far questo sono presenti diverse operazioni:
  - **creazione di una stop list**: lista contenente parole da escludere dall'analisi di un testo
  - **creazione di una mappa contentente termine-peso**
  - **creazione di un file di testo**: nel file le parole sono ordinate per peso (in ordine decrescente). Nel caso di parole con stesso peso si ottiene una priorità basata sull'ordine alfabetico delle chiavi.

Quanto appena descritto è mostrato nel diagramma di classe (Figura 1.3).

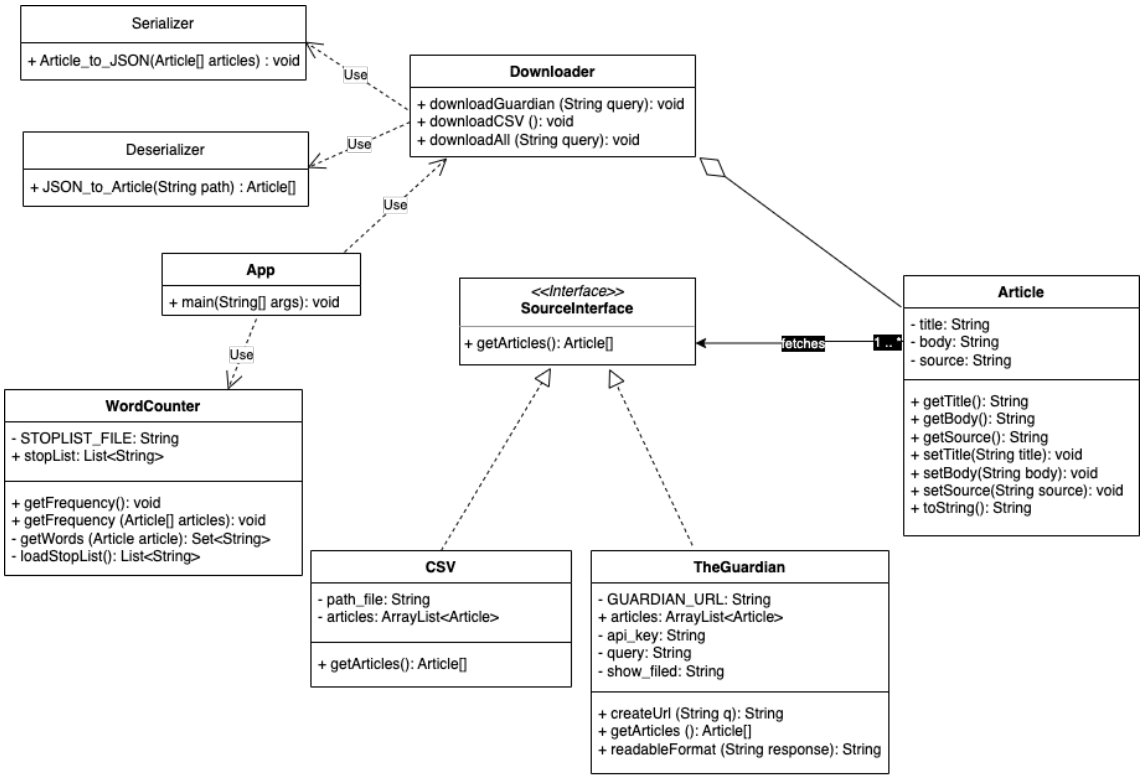



Figura 1.3: Diagramma di classe

Per quanto riguarda le interazioni i casi d'uso sono:

- **Scaricamento**: l'utente lancia il comando di scaricamento, specificando un termine di ricerca (query), all'apposita componente, che invia una richiesta di accesso alle fonti che rilasciano una collezione di articoli, che infine viene serializzata in un file JSON.

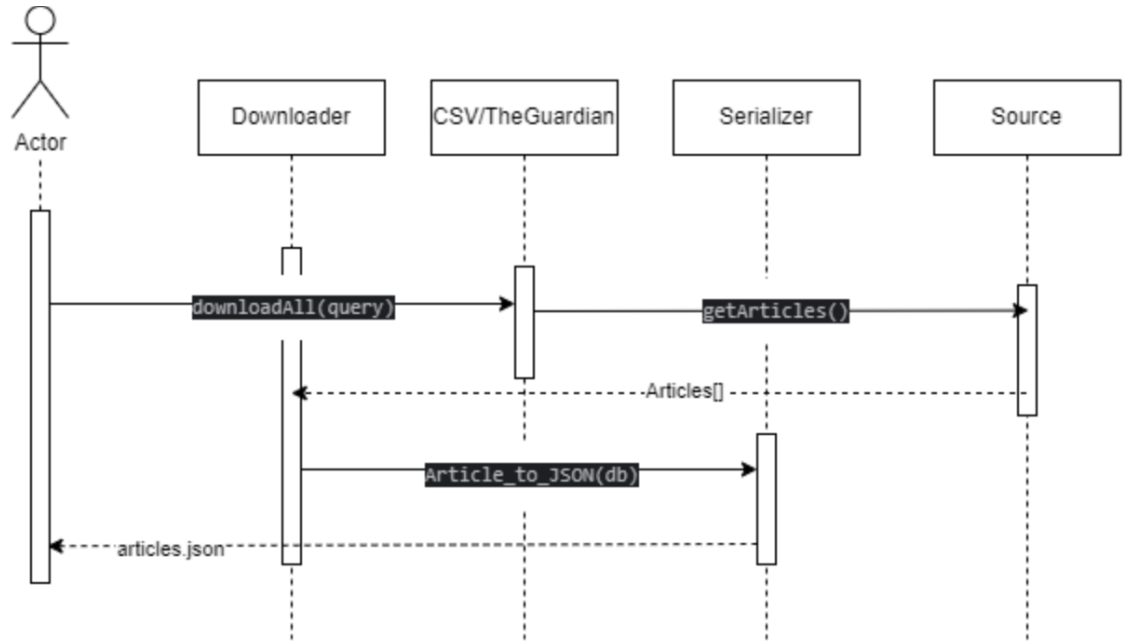



Figura 1.4: Scaricamento

- **Estrazione**: l'utente lancia il comando di analisi degli articoli all'apposita componente che deserializza il file JSON, conta le parole e restituisce all'utente un file wordscount.txt contenente una mappa di 50 parole con il relativo peso.

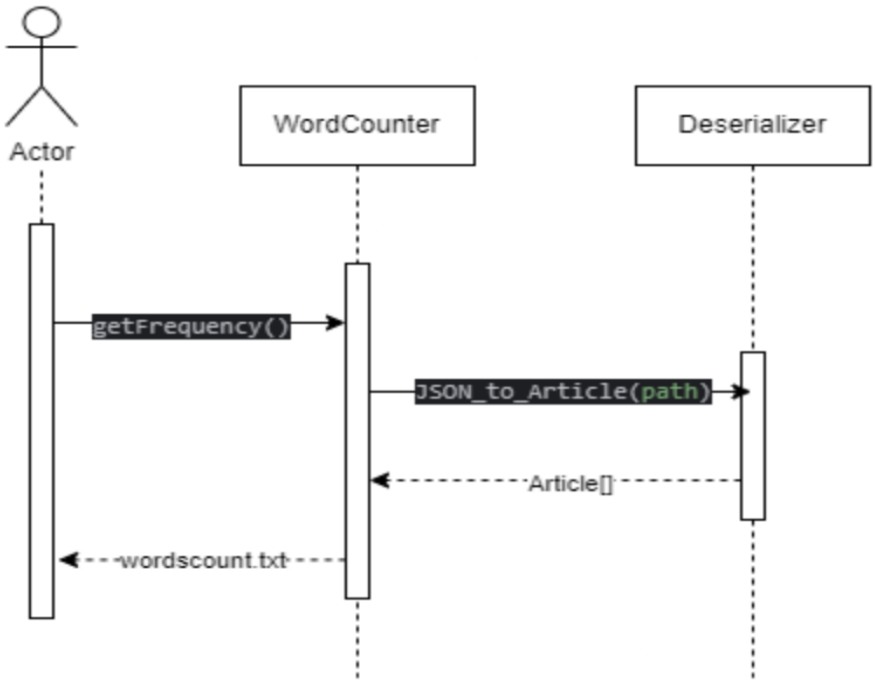



Figura 1.5: Estrazione

- **Scaricamento ed estrazione**: combina i due casi d'uso precedenti.

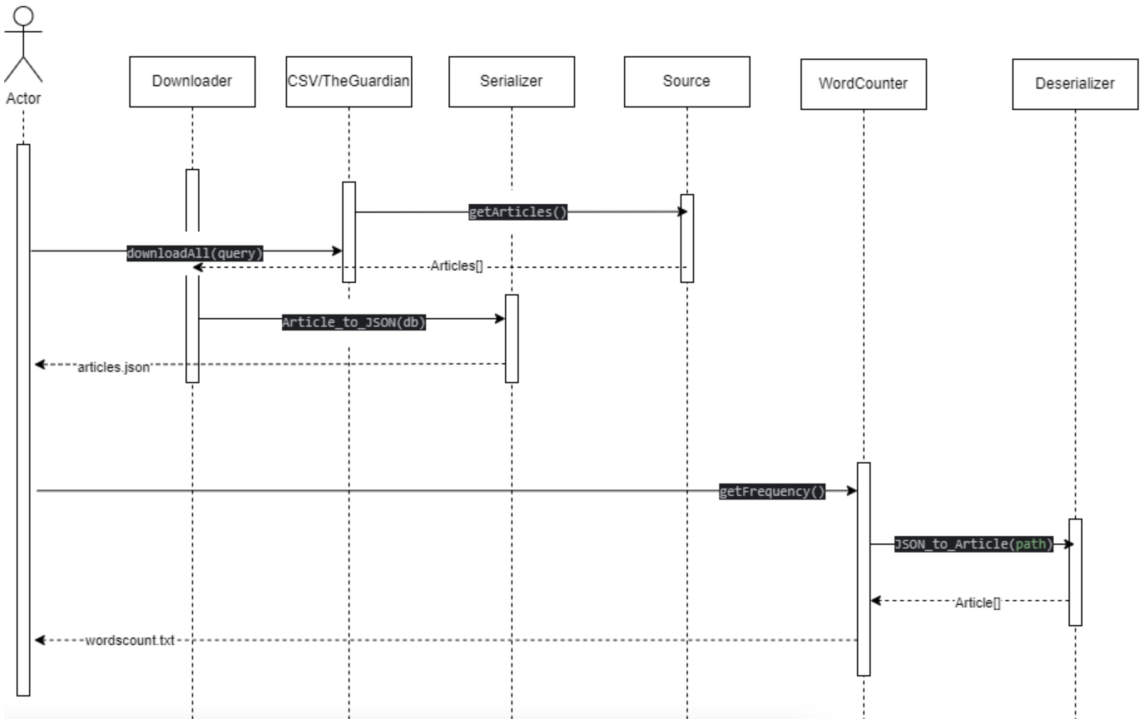



Figura 1.6: Scaricamento ed estrazione

# Capitolo 2

# Manuale

## 2.1 Dipendenze

Tutte le dipendenze, riassunte nel sito construito mediante il comando *mvn site*, sono le seguenti:

- **FasterXML Jackson (version 2.15.1)**: libreria Java per la serializzazione e la deserializzazione di oggetti Java in formato JSON e viceversa.
- **Gson (version 2.8.9)**: libreria Java creata da Google che permette di convertire un oggetto Java in un oggetto JSON e viceversa in modo veloce ed efficiente
- **openCSV (version 5.7.0)**: libreria open source in Java che fornisce un modo semplice e flessibile per leggere e scrivere dati in formato CSV (*Comma-Separated Values*)
- **Apache Commons CLI (version 1.5.0)**: libreria open source in Java che fornisce un framework per la gestione dei parametri da riga di comando. Inoltre semplifica la creazione di user interface da riga di comando permettendo agli sviluppatori di controllare e gestire i parametri passati.
- **Apache Commons CSV (version 1.6)**: Commons CSV legge e scrive file in variazioni del formato Comma Separated Value (CSV), che è un formato di file con valori separati da virgole.
- **Apache Commons Lang (version 3.14.0)**: Apache Commons Lang fornisce una serie di utilità ausiliarie per l'API java.lang, in particolare metodi di manipolazione delle stringhe, metodi numerici di base, riflessione sugli oggetti, concorrenza, creazione e serializzazione, e proprietà di sistema.
- **Apache HttpComponents (version 4.5.13)**: è un insieme di componenti open source sviluppati da Apache Software Foundation per im-

plementare il protocollo HTTP in Java. I seguenti componenti facilitano la creazione e la gestione di richieste e risposte HTTP in software creati tramite il linguaggio di programmazione Java.

- **Junit (version 4.11)**: La piattaforma JUnit funge da fondamento per avviare framework di testing sulla JVM (Java Virtual Machine). Definisce anche l'API TestEngine per lo sviluppo di un framework di testing che può essere eseguito sulla piattaforma. Inoltre, la piattaforma fornisce un Console Launcher per avviare la piattaforma da linea di comando e un Runner basato su JUnit 4 per eseguire qualsiasi TestEngine sulla piattaforma in un ambiente basato su JUnit 4.
- **Junit Jupiter (version 5.9.2)**: JUnit Jupiter è la combinazione del nuovo modello di programmazione e del modello di estensione per scrivere test ed estensioni in JUnit 5. Il sottoprogetto Jupiter fornisce un TestEngine per eseguire test basati su Jupiter sulla piattaforma.

## 2.2 Istruzioni per installare ed eseguire

### Key TheGuardian

Il programma utilizza una key per comunicare con l'API di The Guardian, ottenibile qui.
Una volta scaricato il progetto, è necessario inserire la chiave personale del TheGuardian nel file ./assets/application.properties (al posto di test).

### Come compilare ed eseguire il programma

Per compilare il progetto e creare i file jar:

```
mvn package
```

Per eseguire l'applicazione tramite il file jar:

```
java -jar .\target\EIS_Project-1.0-jar-with-dependencies.jar
    -{d, de, e, h} <query> -{C, G, A}
```

Opzioni disponibili:

```
-d,--download <arg>                 Download the articles
-de,--download & extraction <arg>   Download and extract
-e,--extraction                     Extract terms
-h,--help                           Print the help


-C                                  CSV file
-G                                  TheGuardian file
-A                                  All files
```

Ad esempio, per scaricare ed estrarre (`-de`) articoli da tutte le fonti disponibili (`-A`) usando la query `nuclear+power`:

```
java -jar .\target\EIS_Project-1.0-jar-with-dependencies.jar
    -de nuclear+power -A
```

### Generazione sito

Per generare il sito, basta eseguire il comando:

```
mvn site
```