Automatische Daten Extraktion aus Web Quellen

Tristan Nolde
Fachhochschule Südwestfalen

Schriftliche Ausarbeitung im Modul
„Projektgruppe“

im Studiengang
M.Sc. Angewandte Künstliche Intelligenz

eingereicht bei
Prof. Dr.-Ing. Doga Arinir

3. Oktober 2023

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und dabei keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Werken anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

3. Oktober 2023



Tristan Nolde

Inhaltsverzeichnis

1. Einleitung **1**
1.1. Web Mining 2
1.2. Extract, Transform, Load (ETL) 2

2. Web Scraping & Crawling **3**
2.1. Bezug von Stammdaten aus dem Unternehmensregister 4
2.2. Bezug von Finanzdaten aus dem Bundesanzeiger 5

3. Extraktion von News Artikeln **7**
3.1. RSS Feeds 8
3.2. Tagesschau REST API 10

4. Orchestrierung & Operations **11**

5. Rechtliche Rahmenbedingungen **14**

6. Zusammenfassung **15**
6.1. Fazit 15
6.2. Ausblick 15

A. Anhang **II**
A.1. Company Datenmodell II
A.2. Ablauf Web Scraping im Unternehmensregister III
A.3. Airflow DAG IV

1. Einleitung

Im Projekt *Transparenzregister* steht die Analyse sowie Visualisierung von Unternehmensdaten im Vordergrund. Zu diesen Daten gehören unter anderem Key Performance Indicators (KPIs) im Bereich Finanzen (z.B. Verlauf des Umsatzes oder Jahresüberschusses) aber auch grundlegende Stammdaten wie den Sitz des Unternehmens oder die Besetzung des Aufsichtsrates. Um diese Informationen jedoch darstellen zu können, bedarf es zunächst der Beschaffung und Aufbereitung von Daten. In der im Projekt verankerten Architektur bedeutet dies, die Extraktion von Rohdaten aus verschiedenen Quellen, die Stücke der benötigten Informationen beinhalten, un den Transport dieser in einem geeigneten Format in die sog. *Staging Datenbank (DB)*. Aus dieser heraus bedienen sich die verschiedenen Analyse-Prozesse und stellen letztendlich einen vollkommen aufbereitenden Datenbestand in der *Production DB* zur Verfügung.

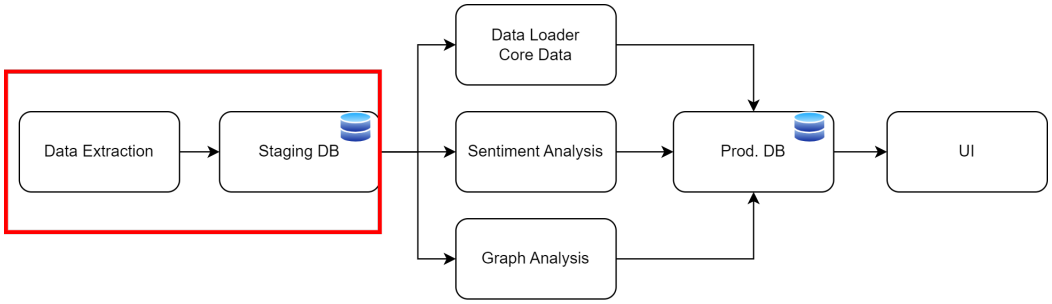


Abbildung 1: Einordnung des Praxisanteils der Hausarbeit in den Gesamtprozess der Projektgruppe

Quelle: Eigene Darstellung

Diese Hausarbeit befasst sich mit der Bereitstellung eben dieser Informationen für die *Staging DB*. Neben der initialen Vermittlung von in der Forschung und Industrie bewährten Methoden, soll primär die praktische Umsetzung der Daten Extraktion für die Projektgruppe im Vordergrund stehen. Dafür werden Verfahren vorgestellt, um die gewünschten Informationen aus verschiedenen Quellen zu beziehen. Zum Ende der Arbeit wird ebenfalls ein erster Blick auf die rechtlichen Rahmenbedingungen dieser Daten-Extraktion sowie ihrer Speicherung geworfen. Da die während der Arbeit entstehenden Applikationen auch betrieben und gewartet werden müssen, wird ein Blick auf Apache Airflow als Workflow Orchestrator geworfen.

Als Quellen für die gewünschte Datenbasis dienen unter anderem die Folgenden:

- Unternehmensregister
- Bundesanzeiger

- Tagesschau
- Handelsblatt

1.1. Web Mining

Das Web Mining als Unterdisziplin des Data Minings - "[..] einer Sammlung von Methoden und Vorgehensweisen, mit denen große Datenmengen effizient analysiert werden können"[1] - beschäftigt sich vorwiegend mit der Auswertung von Quellen aus dem Internet. Da das Internet zu großen Teilen aus textuellen Informationen in Form von Websites besteht, bildet das Web Mining Schnittmengen zum Text Mining.

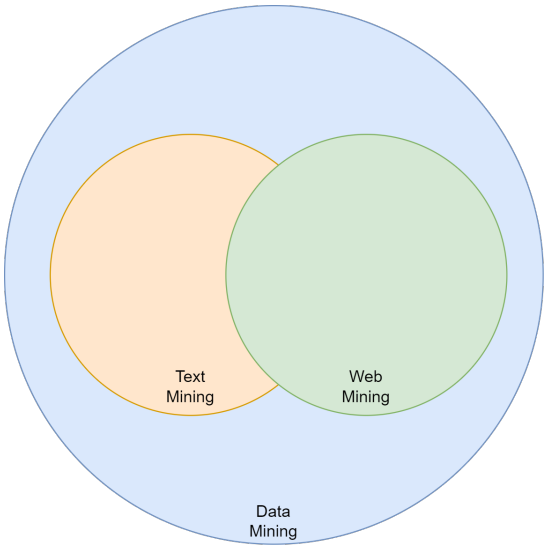


Abbildung 2: Venn Diagramm des Data-, Text- und Web Minings

Quelle: Eigene Darstellung

Besonders im Text Mining wird Wert auf die Analyse von Dokumenten in Form von HTML oder XML Dateien gelegt. Das Web Mining legt lediglich die Schicht der Beschaffung jener Dokumente dazu. [1]

1.2. ETL

Ein grundlegender Prozess, der in allen Data Mining Sub-Formen Anwendung findet ist der des ETL. Dieser beschreibt unabhängig von der ursprünglichen Datenquelle oder ihrem Ziel, den Wunsch, Daten von einem System in einer aufbereiteten Form in das Zielsystem zu transportieren. [1]

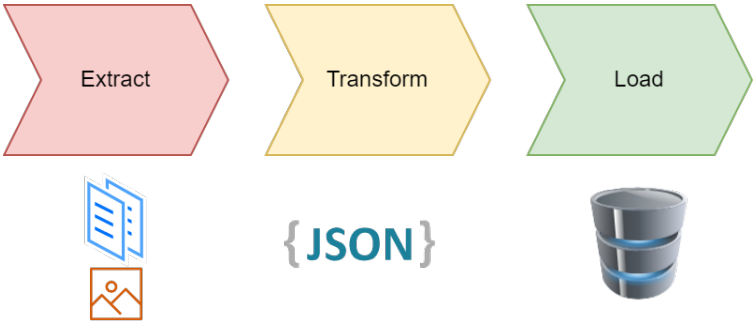


Abbildung 3: ETL Prozess am Beispiel der gegebenen Datenquellen

Quelle: Eigene Darstellung

Da die Ergebnisse dieses Prozesses die Grundlage für Folge-Schritte wie eine weitere Analyse oder bereits Darstellung legt, ist hier ein besonderes Augenmerk auf die Qualität der Rohdaten sowie Ergebnisse zu legen. Ein bekanntes Sprichwort in diesem Kontext ist *"Garbage in, Garbage out"*. Darunter wird der Zustand verstanden, dass niederwertige Rohdaten selbst mit komplexer Aufbereitung nicht in qualitativen Ergebnissen münden können. So ist im Schritte der Extraktion besonderes Augenmerk auf die Auswahl der Datenquelle bzw. Dokumente zu legen. [1]

Die Bedeutung von Qualität findet sich ebenfalls in der Transformation wieder. Hier gilt es nicht nur Daten von einen in das andere Format zu überführen (Harmonisierung), sondern auch ungeeignete Datensätze oder Passagen zu erkennen und nicht weiter zu betrachten (Filterung). [1]

Zuletzt gilt es die aufbereiteten und selektierten Daten in das Zielsystem - hier die Staging Datenbank - zu überführen. Dieser Prozess kann je nach zeitlicher Ausführung (initiales Laden, Aktualisierung) unterschiedlich ablaufen, da entweder neue Daten hinzugefügt oder Bestehende angepasst werden. [1]

2. Web Scraping & Crawling

Eine technische Anwendung des Web Minings ist in Form des *Web Scrapings* (auch: Web Crawling) zu finden. Zunächst beschreibt das Scrapen die Extraktion von Daten aus Websites. Dies kann entweder manuell oder automatisch geschehen. Sollte das zur Automatisierung entwickelte Programm nicht nur auf einer Zielseite verweilen, sondern über auf der Seite gefundene Links zu weiteren Seiten vordringen, so ist vom *Crawling* die Rede. Dieses Vorgehen ist unter anderem bei Suchmaschinen wie Google anzutreffen. Da diese Technik unter anderem eingesetzt werden kann, um sensible Daten wie Telefon-

nummern zu finden, wurden mit der Zeit einige Mechanismen wie die *robots.txt* Datei entwickelt, in jener der Betreiber einer Website festlegt, welche Zugriffe erlaubt sind. [2]

Allerdings hat dieser Mechanismus eine elementare Schachstelle: Die *robots.txt* Einträge dienen lediglich als Richtlinie für Clients, so dass Crawler in der Lage sind diese schlichtweg zu ignorieren und dennoch relevante Daten von der Seite zu scrapen. [3]

2.1. Bezug von Stammdaten aus dem Unternehmensregister

Im Unternehmensregister werden veröffentlichungspflichtige Daten deutscher Unternehmen wie etwa die Firmengründung oder Liquidation in elektronischer Art zur Verfügung gestellt. [4]

Besonders relevant, um überhaupt ein Inventar an Unternehmen aufstellen zu können, sind die dort zu findenden Registerinformationen. In diesen werden Daten aus Handels-, Genossenschafts- und Partnerschaftsregistern bereit gestellt. Das Ziel ist es aus diesen grundlegende Informationen wie den Sitz des Unternehmens, seine Gesellschaftsform sowie ggf. verfügbare Relationen zu Personen oder anderen Unternehmen zu beziehen. Das dazugehörige Ziel-Datenmodell ist im Anhang zu finden[1].

Zu Beginn der Entwicklung wird zunächst einer erster manueller Download Prozess von Daten gestartet. Die Website des Unternehmensregisters stellt eine schlagwort-basierte Suchmaske bereit über die der Nutzer auf eine Historie der gefundenen Unternehmen geleitet wird. Bereits hier stellt sich heraus, dass die Suche des Unternehmensregisters keine 1:1 Abgleich des Unternehmensnamens sondern eine ähnlichkeits-basierte Suche verwendet. So förderte die unten dargestellte Suche nach der "Bayer AGäuch Einträge wie die B̈ayer Gastronomie GmbHżu Tage:

Auch eine Anpassung des Suchbegriffes durch die Verwendung regulärer Ausdrücke (z.B. ̈̂Bayer AG$) kann die Qualität der Ergebnisse nicht verbessern. Da es sich bei diesen ähnlichen Unternehmen jedoch auch tatsächlich um Tochtergesellschaften handeln könnte, die für die Verflechtsungsanalyse besonders interessant sind, werden nicht passende Unternehmensnamen nicht entfernt sondern ebenfalls verarbeitet.

Ist ein Unternehmen ausgewählt, stellt das Unternehmensregister verschiedene Format zur Verfügung. Für eine automatische Auswertung der Daten eignet sich besonders der *SI - strukturierter Registerinhalt*, welcher ein XML-Dokument im Format der XJustiz beinhaltet. Es wird das Schema *xjustiz_0400_register_3_2.xsd* verwendet. [2]

[1] siehe A.1
[2] XJustiz 3.3.1

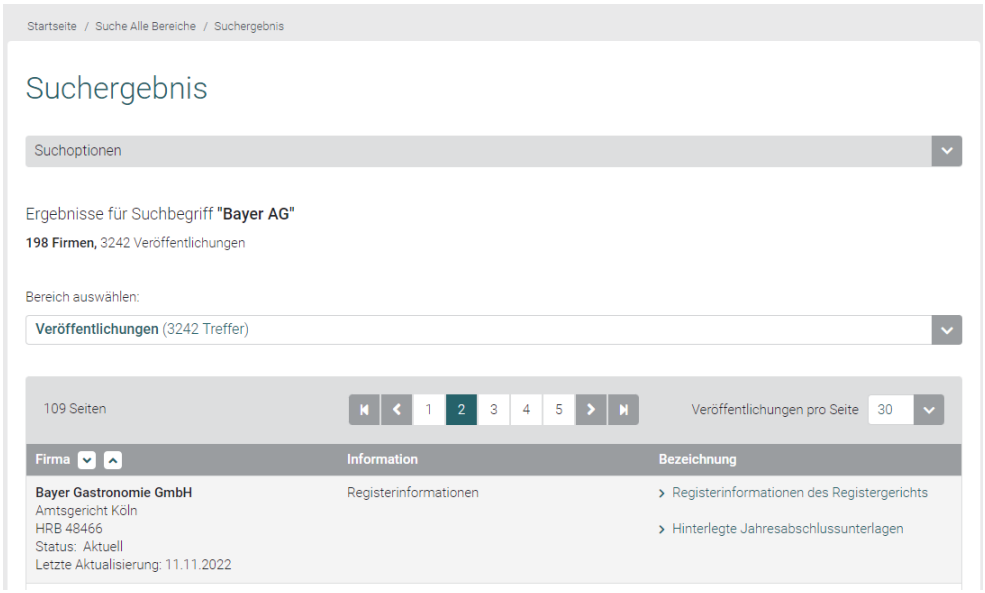


Abbildung 4: Ergebnisse des Unternehmensregister bei Schlagwortsuche

Quelle: Eigene Darstellung

Da das Unternehmensregister keine öffentliche Application Programming Interface (API) hostet, über die dieses Dokument bezogen werden kann, erfolgt eine Automatisierung des über den Nutzer in der Web-Oberfläche getätigten Aktionsfluss mithilfe von Selenium.

Eine Schritt-für-Schritt Beschreibung des Prozesses würde den Rahmen der Hausarbeit sprengen, daher wird ein High-Level Überblick in Form eines Ablaufdiagramms bereit gestellt A.2.

Im nachgelagerten Schritt werden die heruntergeladenen *.xml* Dateien in eine *.json* konvertiert und in das Zielformat überführt. Darauf werden die Daten über einen zentral gelegenen Service, der die Verbindung zur StagingDB (MongoDB) in entsprechende Methoden kapselt, in die Persistenz überführt. [3]

2.2. Bezug von Finanzdaten aus dem Bundesanzeiger

Jede Kapitalgesellschaft in Deutschland ist gemäß §242 des Handeslgesetzbuch (HGB) dazu verpflichtet zu Gründung der Gesellschaft sowie nach jedem abgeschlossen Geschäftsjahr eine Aufstellung der finanziellen Situation des Unternehmens zu veröffentlichen. Diese besteht mindestens aus der Gewinn- und Verlustrechnung, kann jedoch je nach

[3] siehe Projekt Code auf GitHub

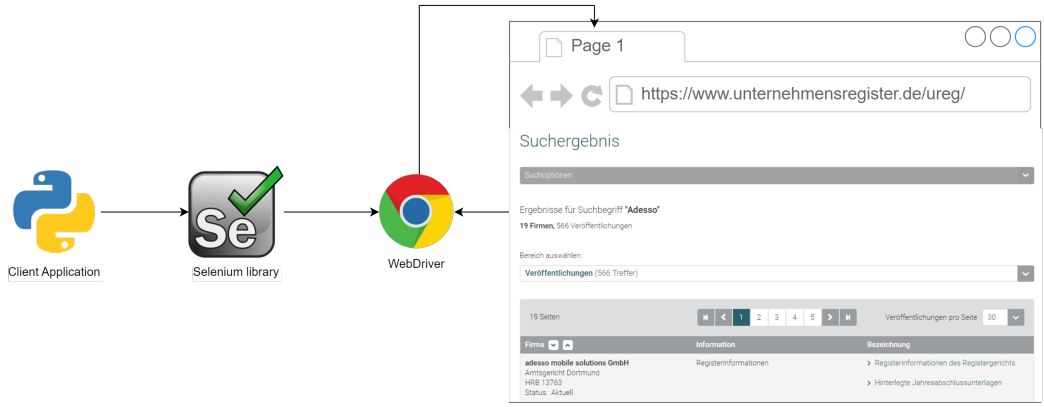


Abbildung 5: Integration von Selenium in die Applikation

Quelle: Eigene Darstellung

Gesellschaftsform weitere Details verpflichtend beinhalten. Im Bundesanzeiger, einem Amtsblatt sowie Online Plattform geleitet vom Bundesministerium der Justiz, werden diese Jahresabschlüsse der Allgemeinheit zur Verfügung gestellt.

Aus Sicht des Transparenzregisters eignet sich der Bundesanzeiger daher als besonders gute Quelle, um Finanzdaten zu bereits bekannten Unternehmen (siehe vorheriges Kapitel) zu beziehen, um den wirtschaftlichen Erfolg einer Kapitalgesellschaft quantifizieren zu können.

Auf GitHub, einem zentralen Anlaufpunkt für Open-Source Projekte, kann eine Python Bibliothek namens deutschland gefunden, welche eine Reihe deutscher Datenquellen kapselt und in einem festen Format zur Verfügung stellt. Unter anderem beinhaltet diese Bibliothek auch ein Sub-Paket namens *bundesanzeiger*, welches ein Interface zur Oberfläche des Bundesanzeigers zur Verfügung stellt. Das Projekt wird unter der Apache License 2.0 betrieben und ist folglich auch für den Einsatz im Projekt geeignet.

Die erste Integration der Bibliothek zeigt jedoch schnell eine große Schwäche auf: Im Zuge des Datenbezugs werden die Rohdaten (HTML) bereits in einfachen Text übersetzt. Dies hat zur Folge, dass wertvolle Informationen, die sich aus der Struktur des Dokumentes ergeben (z.B. der Einsatz von Tabellen, um die Aktiva wie Passiva des Jahresabschlusses aufzulisten). Es wird daher ein Fork des Repositories angelegt und der Quellcode so adaptiert, dass auch der Inhalt in seiner Ursprungsform bereitgestellt wird.

Der damit verbundene Aufwand, um sich in die Bibliothek einzuarbeiten, gibt hilfreiche Einblicke in die genaue Funktionsweise der Implementierung: Über eine Reihe gezielter HTTP-Anfragen gekoppelt mit einer Auswertung über Web Scraping navigiert die Biblio-

thek durch die Seiten des Bundesanzeigers. Wird für eine Seite (z.B. Jahresabschlüsse) ein CAPTCHA - ein Authentifizierungs-Mechanismus im Challenge-Response-Verfahren, der den Zugang durch Bots verhindern soll [5] - abgefragt, so wird dieser mittels eines Machine Learning Models des Chaos Computer Club e.V. gelöst.[4]

Auf Basis dieser Grundlage lassen sich unter anderem Informationen wie die Angabe des Wirtschaftsprüfers eines Jahresabschlusses extrahieren:

```
def extract_auditor_company(self, report: str) -> str | None:
  soup = BeautifulSoup(report, features="html.parser")
  temp = soup.find_all("b")
  for elem in temp:
    br = elem.findChildren("br")
    if len(br) > 0:
      return elem.text.split("\n")[1].strip()
    return None

def extract_auditors(self, report: str) -> list:
  auditor_company = self.extract_auditor_company(report)
  auditor_regex = r"[a-z A-Z,.'-]+, Wirtschaftsprüfer"
  hits = re.findall(auditor_regex, report)
  return [
    Auditor(hit.replace(", Wirtschaftsprüfer", "").lstrip(),
    auditor_company)
    for hit in hits
  ]
```

Diese Methoden werden auf eine anhand des Dokumententyps gefilterte Liste der bezogenen *Reports* angewandt. Eine Erweiterung auf die Extraktion von Finanzdaten steht aus.

3. Extraktion von News Artikeln

Auch wenn das Ziel des ETL-Prozesses eine MongoDB - also eine NoSQL Document Datenbank, die in einer Collection Dokumente jeglicher Struktur akzeptiert - ist, wird vor der Entwicklung ein Datenformat definiert. Dies soll gewährleisten, dass Folge-Komponenten wie die Sentiment-Analyse auf eine feste Definition der Daten zurückgreifen können. Letztlich soll so die Codequalität und Stabilität erhalten werden, da ein nicht-typisiertes Format zu Fehlern führt. Vor der Entwicklung der Komponenten, um Daten aktueller News Artikel zu beziehen, wird daher das folgende Zielformat als Python DataClass definiert und in einem zentralen *models/* Sub-Modul des Repos zur Verfügung gestellt:

[4] Näheres zum Aufbau und Training des Models unter: https://av.tib.eu/media/52366

```
@dataclass
class News:

id: str
title: str
date: str
text: str
source_url: str

def to_dict(self) -> dict:
  return asdict(self)
```

Neben dem Inhalt des Artikels in Form des Titels, sowie dem Text werden wertvolle Meta-Informationen wie das Veröffentlichungsdatum sowie der Link des Artikels mitgeführt. So wird der späteren Analyse die Möglichkeit gegeben, ihre Ergebnisse auch in den zeitlichen Kontext zu legen und im Zuge der Visualisierung Nutzer auf den ursprünglichen Artikel zu navigieren. Des Weiteren wird eine ID als eindeutiger Identifier mitgeführt, um eindeutige Referenzen zu ermöglichen und Duplikate zu verhindern. Die Werte des Feldes orientieren sich an den IDs der jeweiligen Quelle - unter anderem Universally Unique Identifier (UUID). Die Anreicherung der Datensätze mit den betroffenen Unternehmen sowie dem Sentiment erfolgt in der nachgelagerten Analyse.

3.1. RSS Feeds

Im Zuge der Vorbereitung des Projektes wurden Really Simple Syndication (RSS) Feeds als mögliche Quelle für eine regelmäßige Bereitstellung aktueller Nachrichten-Artikel und Events identifiziert. Ein RSS Feed verfolgt das Ziel Informationen regelmäßig über eine Web-Schnittstelle an Kunden zu liefern. Die Inhalte bzw. deren Änderungen werden dabei in einem standardisierten, maschinen-lesbaren Format zur Verfügung gestellt. [6]

Der genaue Aufbau eines RSS Feeds wird im von der Harvard Law School entwickelten RSS 2.0 Standard beschrieben. Er beschreibt zunächst XML als unterliegendes Dateiformat vor. Die Wurzel eines RSS Dokumentes wird von einem sog. *Channel* gebildet, der in sich mehrere *Items* kapselt. Neben dem Titel, Link des Feeds muss ein *Channel* eine Beschreibung enthalten - weitere Meta-Informationen wie die Sprache oder Informationen zum Urheberrecht sind möglich. Ein *Item* hingegen setzt sich aus mindestens dem Titel, Link, Beschreibung, Autoren, Kategorie [...] zusammen. [7] Ein simpler RSS Feed könnte wie folgt aussehen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">

<channel>
  <title>W3Schools Home Page</title>
```

```
  <link>https://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>https://www.w3schools.com/xml/xml_rss.asp</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>https://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>
```

Bezogen auf die Anbindung an das Transparenzregister fällt auf, dass diese Quelle zunächst nur Basis-Informationen des Artikels und noch nicht den eigentlichen Inhalt, welcher für die Sentiment-Analyse sowie Ermittlung betroffener Unternehmen besonders interessant sein könnte, enthält.

Als erste Quelle wird der das Handelsblatts gewählt. Es stellt eine Reihe verschiedener Feeds zur Verfügung, die sich mit unterschiedlichen Themen beschäftigen. Für eine Auswertung von Unternehmens-Daten wird daher anfangs der RSS Feed Schlagzeilen eingesetzt.

Eine Abfrage dieser Daten gestaltet sich aufgrund der standardisierten XML Struktur über eine gezielte HTTP-Anfrage und nachträgliche Auswertung des eigentlichen Artikels recht einfach:

```python
import requests
import xmltodict
from bs4 import BeautifulSoup


class HandelsblattRSS:
  def __init__(self):
    self.base_url = "https://www.handelsblatt.com/contentexport/feed"

  def get_news_for_category(self, category: str = "unternehmen") -> dict:
    url = f"{self.base_url}/{category}"
    result = requests.get(url=url)
    if result.status_code == 200:
      return xmltodict.parse(result.text)["rss"]["channel"]["item"]
    return None

  def get_news_details_text(self, url: str) -> dict:
    content = requests.get(url)
    soup = BeautifulSoup(content.text, features="html.parser")

```

```
    return " ".join(
      [elem.text.replace("\n", " ") for elem in soup.find_all("p")][:]
    )
```

Die eigentlichen Inhalte des Artikels weisen kein Standard Format mehr auf, auch wenn diese HTML basiert sind, und müssen daher mit einer Bibliothek wie BeautifulSoup verarbeitet werden. Im obigen Code werden dabei lediglich alle Textinhalte ausgewertet und konkateniert.

Mit der obigen Python-Klasse können jedoch lediglich Daten extrahiert werden, die Transformation sowie das Laden in die Staging DB erfordern weitere Entwicklung.

```
handelsblatt = HandelsblattRSS()
items = handelsblatt.get_news_for_category()

from datetime import datetime
from tqdm import tqdm
import pandas as pd


news = []
for news_article in tqdm(items):
info = {
  "id": news_article["guid"],
  "title": news_article["title"],
  "date": datetime.strptime(
    news_article["pubDate"], "%a, %d %b %Y %H:%M:%S %z"
    ).strftime("%Y-%m-%dT%H:%M:%S%z"),
  "source_url": news_article["link"],
  "text": handelsblatt.get_news_details_text(news_article["link"])
}
news.append(info)
```

3.2. Tagesschau REST API

Als weitere Quelle für News Artikel wird die Tagesschau API eingesetzt. Diese wird von bund.dev - einer Bemühung Verwaltungsdaten und Informationen über APIs zur Verfügung zu stellen[8], bereitgestellt und ermöglicht Anwendern einen maschinen-lesbaren Zugang zu Artikeln der Tagesschau.

Obwohl der Zugang zu diesen Daten ebenfalls über HTTP GET Anfragen möglich ist, so entspricht das Datenformat keinem Standard, sondern dem Produkt-eigenen Datenschema, welches vom Betreiber mit Swagger dokumentiert wurde, und erfordert daher eine andere Aufbereitungs-Routine. Der Abruf der Daten bleibt jedoch simpel und eine gezielte Suche nach Kategorien ist möglich.

```python
import json
import requests
from bs4 import BeautifulSoup


class TagesschauAPI:
  def __init__(self):
    self.base_url = "https://www.tagesschau.de/api2"

  def get_news_for_sector(self, sector: str) -> dict:
    url = f"{self.base_url}/news/"
    regions = ",".join([str(i) for i in range(1, 16)])
    result = requests.get(url=url, params={"regions": regions, "ressort": sector})
    return result.json()

  def custom_search(self, query: str) -> dict:
    url = f"{self.base_url}/search/"
    result = requests.get(url=url, params={"searchText": query})
    return result.json()

  def get_news_details_text(self, url: str) -> dict:
    content = requests.get(url)
    soup = BeautifulSoup(content.text, features="html.parser")

    return " ".join(
      [elem.text.replace("\n", " ") for elem in soup.find_all("p")][1:]
    )
```

Auch hier werden erneut Details über den ursprünglichen Artikel bezogen. Dafür wird der HTML-Inhalt der ursprünglichen Seite mit BeautifulSoup geparsed und jeglicher Text des Dokumentes extrahiert.

4. Orchestrierung & Operations

Sobald die obigen Lösungen das Entwicklungsstadium verlassen und folglich eine Ausführungsumgebung benötigen, stellen sich die folgenden Fragen:

- Wie kann die Ausführung der Anwendungen geplant und zeitbasiert gestartet werden? (Scheduling)
- Wie lässt sich ein Monitoring auf die Ausführungsumgebung anwenden? (Transparenz)
- Wie lässt sich die Ausführungsumgebung um weitere Anwendungen erweitern? (Skalierbarkeit)

Zum einen wird ein Scheduling benötigt, um die Extraktion aktueller News Artikel in festgelegten Intervallen (z.B. täglich) durchzuführen. Des Weiteren muss es möglich sein, umfassende Informationen über diese Ausführung zu erhalten. Da der Nutzer nicht vor dem Monitor sitzt, das Skript manuell startet und seine Ausführung überwacht. Folglich ist es schwierig im Nachhinein nachzuvollziehen, woran die Ausführung gescheitert ist und wann dies geschah. Zuletzt muss die Lösung in der Lage sein, auch weitere Anwendungen aufzunehmen, da auch Prozesse wie die Extraktion der Stammdaten oder der Finanzdaten in Intervallen - wenn auch größer, da Jahresabschlüsse immerhin nur jährlich anfallen - ausgeführt werden können.

Eine einfache Lösung ist der Einsatz vom in Unix integrierten Cron Dienst. Dieser ermöglicht den zeitbasierten Start von Skripten oder Anwendungen. [9]

```
0 8 * * * /var/scripts/daily_cron.sh
```

Problematisch gestaltet sich beim Cron jedoch die Bereitstellung von Fehlermeldungen sowie die Kapselung von Ausführungsumgebungen, welches vor allem bei Python Anwendungen von hoher Bedeutung ist, damit es nicht zu kollidierenden Abhängigkeiten kommt.

Eine in der Industrie verbreitete Lösung stellt Apache Airflow dar.

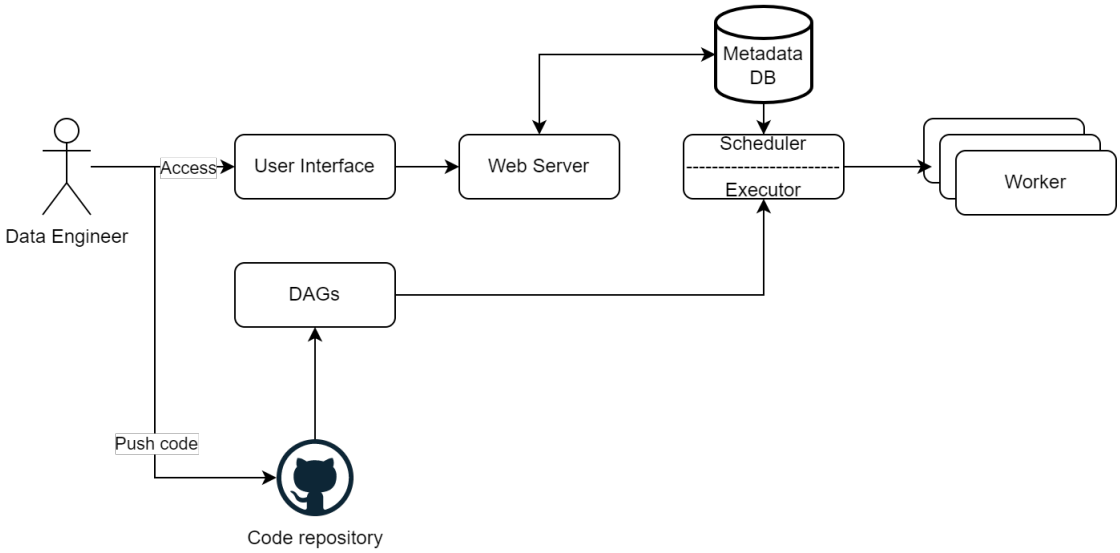


Abbildung 6: Überblick der Architektur von Apache Airflow

Quelle: Eigene Darstellung

Wie in der obigen Grafik dargestellt, setzt sich eine Airlflow Instanz aus verschiedenen Sub-Komponenten zusammen. Neben einem Web Server, der eine grafische Bedienoberfläche

sowie dazugehörige API-Endpunkte stellt, besteht der Kern aus dem Scheduler sowie Executor und der dazugehörigen Datenbank. Anwender legen in Form eines Directed Acyclic Graph (DAG) einen Anwendungsstrang an, der über den Scheduler aufgerufen und im Executor sowie seinen *Workern* - also dedizierten Ausführungsumgebungen - letztlich gestartet wird. Besonders in einer auf Kubernetes bereitgestellten Instanz, der administrative Rechte über das Cluster gegeben wurden, kann Airflow eigenständig für jede Ausführung einen temporären Pod hochfahren, der somit die jeweiligen DAGs komplett isoliert ausführt und nach Ausführung wieder terminiert, um Ressourcen zu sparen. Zusätzlich besteht die Möglichkeiten den Quellcode der DAGs automatisch über einen *Git-sync* Mechanismus mit einem Git Repo zu verbinden. So kann neuer Code automatisch auf die Instanz übertragen werden, ohne dass dieser manuell auf den Server kopiert werden muss. [10]

Zunächst wird im Rahmen des Transparenzregisters jedoch ein simples Deployment auf Basis von Docker Compose bereitgestellt.[5] Das Deployment besteht aus den Kern-Komponenten sowie einem einzelnen Worker. So gehen zwar die flexiblen Möglichkeiten des Hochfahrens individueller Pods verloren, jedoch ist die Einrichtung umso einfacher. Um die Ausführung der Extraktion der News Artikel zu automatisieren, wird der Quellcode in das Format eines DAG überführt und an den Airflow Server übertragen. [6]

Aus Sicherheitsgründen wurden die Zugangsdaten zur StagingDB in die Airflow Variables überführt, so dass die genauen Daten nicht im Quellcode ersichtlich sind sowie zentral verwaltet werden können, um schnell zwischen verschiedenen Ziel-Datenbanken wechseln zu können. Diese werden in der grafischen Oberfläche durch den Admin Account unter dem Menüpunkt *Admin/Variables* bzw. dem Link */variable* verwaltet.

Über den Einsatz des *@task.virtualenv* Decorators wird sichergestellt, dass jede Methode ihr eigenes Virtual Environment nutzt und so keine Kollision von Python Abhängigkeiten auftreten können. Über das Argument *schedule=*
"@daily¨ wird der DAG auf eine tägliche Ausführung terminiert. Der Startzeitpunkt (*start_date*) muss vor dem aktuellen Datum liegen, damit der Scheduler sofort agiert. Nach der Definition der einzelnen Schritte, wird die Hauptlogik in Form eines geordneten Aufrufs der eben definierten Methode und Übergabe der nötigen Parameter definiert.

[5] Die Basis dafür legt die offizielle Dokumentation.
[6] Siehe A.3

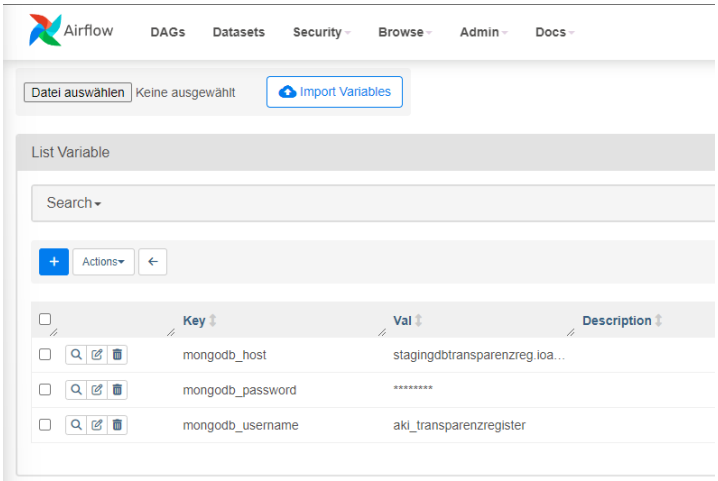


Abbildung 7: Verwaltungsoberfläche von Variablen in Airflow

Quelle: Eigene Darstellung

5. Rechtliche Rahmenbedingungen

Der Einsatz von Web Scraping / Crawling, um die Daten der vorgestellten Quellen zu beziehen, lässt sich auf verschiedene Bereiche geltenden Rechts verweisen, um zu bewerten auf welche weiteren Rahmenbedingungen Rücksicht genommen werden muss.

Zum einen ist das Urheberrecht der Daten zu berücksichtigen. Da es sich bei dem Projekt um nicht-kommerzielle Forschung handelt, ist ein Scraping grundsätzlich gemäß §60d des Urheberrechtsgesetz (UrhG) erlaubt. Jedoch ist darauf zu achten, dass Mechanismen wie die *robots.txt* sowie weitere technische Hürden - etwa IP-Sperren - nicht umgangen werden. Außerdem darf dem Betreiber der Website durch die erzeugte Last keine Schäden beigefügt werden[11]

Des Weiteren ist besonderes Augenmerk auf den Datenschutz zu legen. Da im Transparenzregister eine Reihe von personenbezogenen Daten wie Namen, Geburtsdaten und Wohnort sowie Firmenzugehörigkeiten verarbeitet werden, muss sichergestellt sein, dass Richtlinien der Datenschutzgrundverordnung (DSGVO) eingehalten werden. Auch wenn es sich bei den verarbeiteten Daten um frei zugängliches Material handelt, muss dennoch ein Rahmenwerk aufgebaut werden, welches die Ziele des Projektes klar absteckt, um seinen Forschungscharakter zu wahren, und ggf. entsprechende Maßnahmen wie etwa ein zeitlich begrenzte Speicherdauer und Zugangsbeschränkung implementiert werden. [11]

6. Zusammenfassung

Im Rahmen der Hausarbeit wurden zunächst die theoretischen Grundlagen bestehend aus dem Web Scraping und Crawling sowie dem ETL-Prozess vermittelt. Darauf folgte die Vertiefung dieser Themen anhand der praktischen Umsetzung in der Data Extraktion für das Transparenzregister aus unterschiedlichen Quellen. Zu diesen zählten unterschiedliche Provider für News Artikel wie etwa der RSS Feed des Handelsblattes sowie die Tagesschau REST API der *bund.dev* Gruppe. Darauf folgend wurden Unternehmens Stammdaten aus dem Unternehmensregister und Bewegungsdaten aus dem Bundesanzeiger bezogen. Zuletzt folgte die Ermittlung einer geeigneten Ausführungsplattform in Form von Apache Airflow sowie das Deployment der ersten DAGs auf dieser. Rechtliche Rahmenbedingungen, die die weitere Entwicklung beeinflussen werden, wurden am Schluss dargestellt.

6.1. Fazit

Beim Bezug von Daten aus staatlichen Quellen zeigte sich schnell, dass noch keine vollends in APIs überführte Lösungen vorliegen. So gestaltete sich die Exktaktion von Daten hier im Vergleich zu den News Artikeln deutlich komplizierter und vor allem fehleranfälliger. Sollte sich die Struktur der Website nur im geringsten ändern - zum Beispiel indem Buttons umbenannt oder komplett ersetzt werden - so wird die Anwendung nicht mehr korrekt agieren können. Da es sich ohnehin um öffentliche Daten für den allgemeinen Zugriff handelt, wäre hier eine anständige Lösung auf Seiten der Betreiber wünschenswert.

Der Einsatz von Apache Airflow stellte sich in dem gegebenen Aufbau als fehleranfällig heraus. Dadurch, dass die Instanz nur über einen einzigen Worker verfügt, der sich jedoch von verschiedenen Anwendungen geteilt wird, die dennoch unterschiedliche Abhängigkeiten einsetzen müssen, war das Aufsetzen der DAGs deutlich komplizierter und auch der Einsatz von Bibliotheken, um Duplikate zu vermeiden, konnte dem Worker nur schwer beigebracht werden. [7] Hier wurde deutlich, dass - trotz komplizierteren initialen Setups - ein auf Kubernetes und dem *KubernetesPodOperator* basiertes Deployment hier eleganter wäre. Dennoch ist das Monitoring von Airflow im Vergleich zu anderen Lösungen zu loben, so dass ein weiterer Einsatz geplant ist.

6.2. Ausblick

Nun gilt es die entwickelten Puzzle-Stücke zusammenzusetzen und die StagingDB mit immer mehr Daten zu versorgen. Da News Artikel nun regelmäßig eingespeist werden und solange sich an den APIs der Betreiber nichts ändert auch keine Code Änderungen erfor-

[7] Airflow setzt dafür auf sog. *plugins*, die getrennt vom eigentlichen Quellcode verwaltet werden.[10]

derlich sind, wird der Fokus klar auf dem Beziehen weiteren Stamm- und Bewegungsdaten der Unternehmen liegen, um so die weiteren Prozesse wie die Verflechtugnsanalyse und Datenvisualisierung mit Informationen zu versorgen. Sollten sich im Laufe der Entwicklung weitere News-Quellen als wertvoll erweisen, kann auf der bestehenden Code-Basis schnell eine weitere Integration geschaffen werden.

Des Weiteren mag der im vorherigen Absatz angesprochene Vorschlag, Airflow auf Kubernetes aufzusetzen, eine sinnvolle Ergänzung sein. In Zukunft werden auch Finanzdaten in regelmäßigen Abständen vom Bundesanzeiger bezogen werden müssen, so dass die Investition in eine besser skalierbare Lösung wertvoll wäre.

Ein weiterer indirekter Vorteil von Airflow ist, dass Entwickler dazu gezwungen sind, ihren Code möglichst modular zu halten, um so die Ausführungsumgebung zu abstrahieren. Der Einsatz von Airflow und das Verpacken des Codes in einen DAG sollte lediglich bestehende Klassen/Methoden verwenden und in die richtige Reihenfolge bringen bzw. verknüpfen. Der Aufbau einer klaren Interface-Struktur und guten Teststrategie ist daher sehr wichtig. Der damit verbundene Aufwand muss entsprechend eingeplant werden.

A. Anhang

A.1. Company Datenmodell

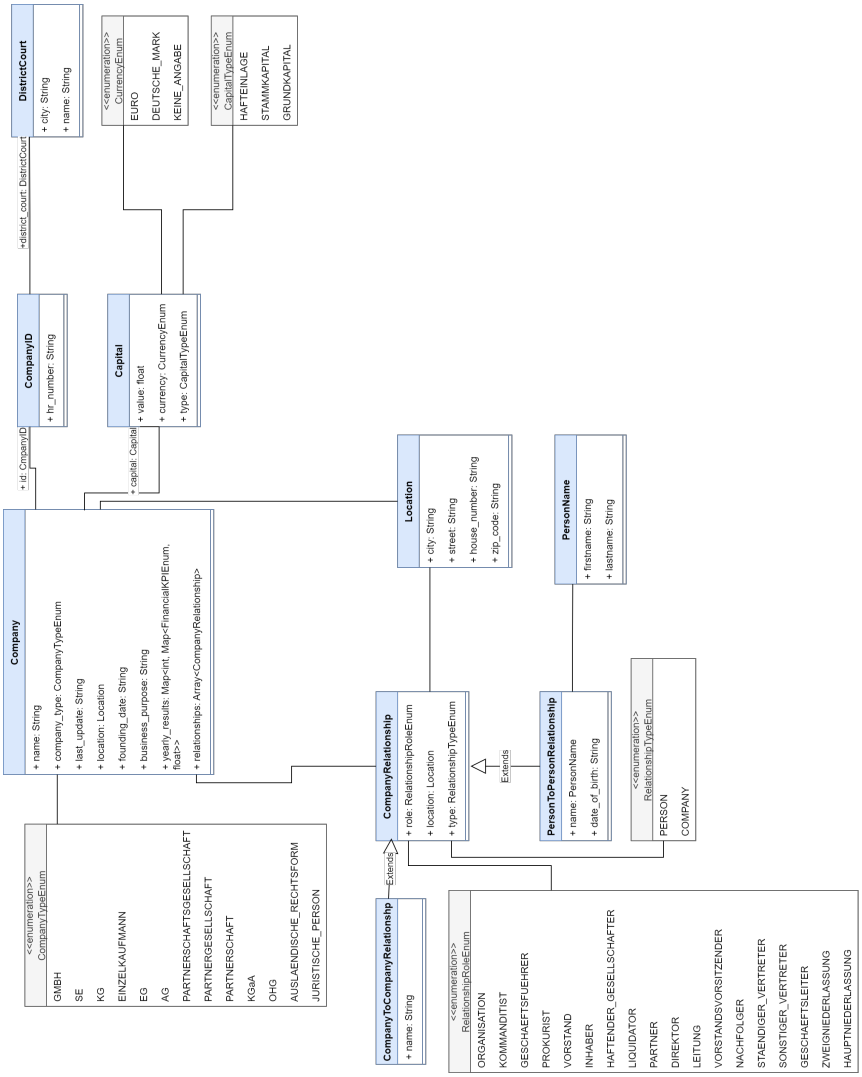


Abbildung 8: Datenstruktur der Unternehmensdaten auf der StagingDB

Quelle: Eigene Darstellung

A.2. Ablauf Web Scraping im Unternehmensregister

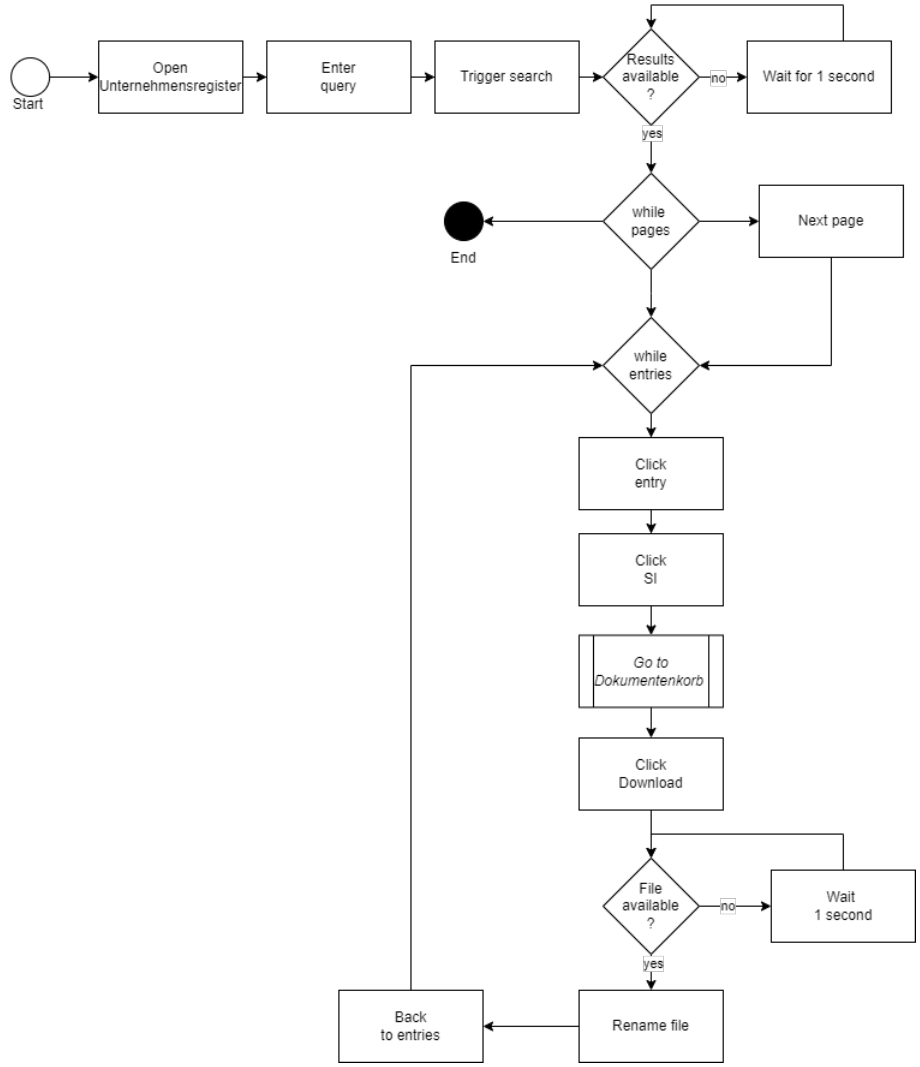


Abbildung 9: Ablaufdiagramm des Web Scrapings des Unternehmensregisters

Quelle: Eigene Darstellung

A.3. Airflow DAG

```python
import datetime

from airflow.decorators import dag, task

@dag("handelsblatt_news_dag", start_date=datetime.datetime(2000,1,1),
    schedule="@daily", catchup=False)
def main():

@task.virtualenv(system_site_packages=False, requirements=['requests', '
    bs4', 'xmltodict'])
def fetch_data():
  from datetime import datetime
  from transparenzregister.utils.handelsblatt import HandelsblattRSS

  handelsblatt = HandelsblattRSS()

  items = handelsblatt.get_news_for_category()

  data = []
  for article in items:
    info = {
      "id": article["guid"],
      "title": article["title"],
      "date": datetime.strptime(
      article["pubDate"], "%a, %d %b %Y %H:%M:%S %z"
      ).strftime("%Y-%m-%dT%H:%M:%S%z"),
      "source_url": article["link"],
      "text": handelsblatt.get_news_details_text(article["link"]),
    }
    data.append(info)
  return data

@task.virtualenv(requirements=["pymongo"], task_id="Process_data")
def process_data(list_news: list) -> list:
  from airflow.models import Variable
  from transparenzregister.utils.mongo import MongoConnector,
    MongoNewsService
  from transparenzregister.models.news import News

  connector = MongoConnector(
    hostname=Variable.get("mongodb_host"),
    database="transparenzregister",
    username=Variable.get("mongodb_username"),
    password=Variable.get("mongodb_password"),
    port=None
  )
  service = MongoNewsService(connector)
```

```
45
46    num_inserted = 0
47    for info in list_news:
48      article = News(**info)
49      if service.get_by_id(article.id) is None:
50        service.insert(article)
51      num_inserted += 1
52    return num_inserted
53
54  raw_data = fetch_data()
55  inserted_entries = process_data(raw_data)
56  print(f"Number of inserted entries: {inserted_entries}")
57  return inserted_entries
58
59  news_dag = main()
```

Literatur

[1] A. R. Samanpour, *Studienunterlagen, Business Intelligence 1*, 1. Aufl. Iserlohn: Wissenschaftliche Genossenschaft Südwestfalen eG, 2016.

[2] „Was ist Web Scraping?“ (2020), Adresse: `https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-web-scraping/` (besucht am 16. 09. 2023).

[3] „Indexierungsmanagement mit der robots.txt.“ (2016), Adresse: `https://www.ionos.de/digitalguide/hosting/hosting-technik/indexierungsmanagement-mit-der-robotstxt/` (besucht am 16. 09. 2023).

[4] „Unternehmensregister.“ (), Adresse: `https://www.unternehmensregister.de/` (besucht am 24. 09. 2023).

[5] „What is CAPTCHA?“ (), Adresse: `https://support.google.com/a/answer/1217728?hl=en` (besucht am 03. 10. 2023).

[6] P. D. D. Markgraf. „RSS-Feed.“ (2018), Adresse: `https://wirtschaftslexikon.gabler.de/definition/rss-feed-53722/version-276790` (besucht am 16. 09. 2023).

[7] „RSS 2.0.“ (2014), Adresse: `https://cyber.harvard.edu/rss/` (besucht am 16. 09. 2023).

[8] „Wir dokumentieren Deutschland.“ (), Adresse: `https://bund.dev/` (besucht am 16. 09. 2023).

[9] „Cron.“ (2023), Adresse: `https://wiki.ubuntuusers.de/Cron/` (besucht am 01. 10. 2023).

[10] „Apache Airflow.“ (), Adresse: `https://airflow.apache.org/docs/apache-airflow/2.7.1/index.html` (besucht am 01. 10. 2023).

[11] „Handreichung zur rechtskonformen Durchführungg von Web-Scraping Projekten in der nicht-kommerziellen wissenschaftlichen Forschung.“ (), Adresse: `https://www.wiso.uni-hamburg.de/forschung/forschungslabor/downloads/20200130-handreichung-web-scraping.pdf` (besucht am 02. 10. 2023).

Abbildungsverzeichnis

1. Einordnung des Praxisanteils der Hausarbeit in den Gesamtprozess der Projektgruppe 1
2. Venn Diagramm des Data-, Text- und Web Minings 2
3. ETL Prozess am Beispiel der gegebenen Datenquellen 3
4. Ergebnisse des Unternehmensregister bei Schlagwortsuche 5
5. Integration von Selenium in die Applikation 6
6. Überblick der Architektur von Apache Airflow 12
7. Verwaltungsoberfläche von Variablen in Airflow 14
8. Datenstruktur der Unternehmensdaten auf der StagingDB II
9. Ablaufdiagramm des Web Scrapings des Unternehmensregisters III

Abkürzungsverzeichnis

KPIs Key Performance Indicators

DB Datenbank

ETL Extract, Transform, Load

API Application Programming Interface

HGB Handeslgesetzbuch

UUID Universally Unique Identifier

RSS Really Simple Syndication

DAG Directed Acyclic Graph

UrhG Urheberrechtsgesetz

DSGVO Datenschutzgrundverordnung