Studienarbeit

im Zusammenhang mit einem Forschungsprojekt

an der

Hochschule München
für angewandte Wissenschaften

mit dem Titel

Extraktion unstrukturierter Daten und Integration in eine Datenbank am Beispiel TripAdvisor

Fakultät für Informatik und Mathematik
Studiengang Master Wirtschaftsinformatik

Verfasser:	Johannes Knippel, Anja Wolf, Johanna Sickendiek, Skanny Morandi
1. Prüfer:	Dr. Michael Möhring
2. Prüfer:	Barbara Keller
Betreuer an der Hochschule München:	Prof. Dr. Rainer Schmidt
Ausgabedatum:	19.03.2018
Abgabedatum:	22.07.2018

Ort, Datum	Unterschrift des 1. Prüfers
	Dr. Michael Möhring
Ort, Datum	Unterschrift des 2. Prüfers
	Barbara Keller
Ort, Datum	Unterschrift des Verfassers
	Johannes Knippel [19793313]
Ort, Datum	Unterschrift des Verfassers
	Anja Wolf [03361217]
Ort, Datum	Unterschrift des Verfassers
	Johanna Sickendiek [00381113]
Ort, Datum	Unterschrift des Verfassers
	Skanny Morandi [21857314]

Inhaltsverzeichnis

1 Problemstellung

Für nahezu Jedermann bietet das WWW im Vergleich zu damaligen Zeiten ungeahnte Möglichkeiten der Recherche und Themenfindung. Noch nie waren so viele Informationen öffentlich einsehbar wie heute. Meist sind jedoch die gewünschten Inhalte an vielen unterschiedlichen Orten im Web verteilt, auf unterschiedlichste Weisen formatiert oder aufgrund schlechter Informationsarchitektur nur mit größerem Aufwand zu erreichen. Grundsätzlich kann die Fülle an Ergebnissen überfordern und nicht immer sind alle relevanten Inhalte über gängige Suchmaschinen auffindbar. Das bedeutet dann oft die mühsame Suche nach den relevanten Informationen in nicht immer nutzerfreundlich gestalteten Datenbanken. Bei so gut wie allen Recherchearbeiten besteht die Aufgabe heutzutage darin, die sprichwörtliche „Nadel im Heuhaufen“ zu finden. In vielen Fällen wünscht man sich einen gewissen Grad an Automatisierung. Daher soll im Rahmen eines Forschungsprojektes an der Hochschule München ein Tool entworfen werden, das Informationen und Inhalte von Webseiten automatisiert gewinnt.

Programmiertechnisch bedient man sich hierfür meistens vorhandener Schnittstellen, die die jeweiligen Quellen zur Verfügung stellen. Neben anwendungsspezifischen APIs sind dies zum Beispiel RSS- oder AtomFeeds, die in eigene Datenbanken geladen und von dort aus weiterverarbeitet werden können. [1] Die Informationen liegen in diesen Fällen also bereits in einem strukturierten Format vor und enthalten wenig bis gar keinen überflüssigen Inhalt.

Hingegen wenn keine der genannten Schnittstellen angeboten werden, bleibt die ständige Beobachtung dieser Inhalte schwierig. Sofern es sich bei den Quellen um einfach strukturierte HTML-Seiten handelt, kann man sich mit verschiedenen Tools weiterhelfen. Gerade bei öffentlichen Datenbanken, deren Inhalte über Webformulare abgefragt werden („Deep Web“), ist dies jedoch meistens nicht möglich. [2] Hier können Webscraper weiterhelfen, die das automatisierte „Auslesen“ von bestimmten strukturierten oder semi-strukturierten Inhalten aus öffentlich zugänglichen Webseiten ermöglichen.

Diese Studienarbeit beleuchtet am Beispiel von TripAdvisor wie ein solcher Web-scraper implementiert werden kann. Des Weiteren umfasst diese Studienarbeit das automatisierte Abspeichern der gewonnenen Daten in einer strukturierten Datenbank.

2 Grundlagen

Das folgende Kapitel geht auf grundlegende Datenstrukturen ein, beleuchtet das Thema Web Scraping, dessen Funktionsweise und Verwendung sowie die Beschreibung des Reiseinformationsportals TripAdvisor.

2.1 Datenstrukturen

Sowohl der Kontext von Daten, als auch ihre Eigenschaften führen zur Strukturierung dieser in verschiedene Datenarten. Kontextinformationen machen genauere Angaben zu Prozessen in denen die Daten benötigt werden. Eigenschaften hingegen unterscheiden Daten nach Format, Inhalt, Stabilität, Verarbeitung oder Struktur.
Besonders relevant für dieses Projekt ist folglich letzteres – die Struktur von Daten. Hier unterscheidet man grundsätzlich zwischen drei verschiedenen Fällen, strukturierten, semistrukturierten und unstrukturierten Daten.
Bei strukturierten Daten sind Metadaten, d.h. strukturgebende Informationen zu Daten vorhanden. Diese können u.a. Auskunft zu Format, erlaubten Werten oder semantischer Bedeutung geben. Ein klassisches Beispiel für strukturierte Daten sind alle möglichen Formen von Datenbanken. Hier müssen zunächst das Schema der Datenbank und deren einzelnen Tabellen festgelegt werden bevor Daten in die definierten Felder einfügt werden können. Somit ist eine einfache Analyse dieser zu einem späteren Zeitpunkt sichergestellt.
Bei semistrukturierte Daten sind einzelne Bestandteile strukturiert, es mangelt jedoch an einer eindeutigen strukturierten Gesamtheit. Die Daten sind keinem objektorientierten oder relationalem Datenbankschema untergeordnet.
Bei unstrukturierten Daten wird keinem eindeutigen Schema gefolgt, nur einer reinen Bitfolge. Das bedeutet jedoch nicht, dass die Daten gar keine Struktur aufweisen. Diese ist jedoch nicht direkt bzw. eindeutig für den Nutzer erkennbar. Neben Multimedia Daten (Bilder, Musik) finden sich unstrukturierten Daten auch meist in Emails, auf Internetseiten oder in Text-Dokumenten [6].

Mithilfe von Web Scraping werden unstrukturierte Daten im Web zu Daten mit strukturgebenden Metadaten transformiert. Was genau Web Scraping bedeutet und wie es funktioniert, wird im nächsten Abschnitt näher erläutert.

2.2 Web Scraping

Mit Web Scraping – auch “Web Harvesting”, “Web data mining” oder “Web data extraction“ genannt – wird das automatisierte herunterladen, analysieren und organisieren von Daten mehrerer Webseiten bezeichnet. Der Vorteil ist, dass aus einer Menge von unstrukturierten Informationen nur die benötigten Daten einer Webseite gezielt extrahiert und strukturiert abgespeichert werden. [7]
Der Begriff „Scraping“ (engl.: „abkratzen, schaben“) gibt es circa so lange wie das Internet selbst. Bevor Web Scraping populär wurde war das Verfahren „Screen Scraping“ bereits allgemein bekannt. Damit wird das Auslesen von Daten aus Darstellungen am Computer bezeichnet. Genau wie heute war man damals schon daran interessiert, große Textmengen zu extrahieren, um daraus nur die benötigten Daten für den späteren Gebrauch zu speichern. Da dieses Verfahren heutzutage meist nur noch für Webseiten benötigt wird, wurde der Begriff „Web Scraping“ etabliert. [7]
Oft wird auch als Synonym der Begriff „Web Crawling“ verwendet. Beide Verfahren untersuchen Daten aus dem Internet, jedoch ist Web Crawling ein Prozess, der eine große Anzahl von Webseiten nach bestimmten Informationen durchsucht. Ein Beispiel dafür sind Suchmaschinen, wie beispielsweise Google, die mithilfe von Crawlern präzise Suchergebnisse anzeigen können. Im Gegenzug dazu ist Web Scraper ein Prozess, der nur bestimmte Informationen einer begrenzten Auswahl von Webseiten sucht. Diese werden daraufhin in ein strukturiertes Format gebracht. [8]

2.2.1 Funktionsweise

Mithilfe von HTTP-Anfragen an eine ausgewählte URL wird der Inhalt einer Webseite – meist in Form von HTML – eingelesen. Dieser HTML-Code kann mithilfe von diversen Frameworks geparst werden, um die benötigten Daten zu erhalten. Die gängigste Technik, um diese Daten zu analysieren, ist mithilfe von HTML-Parsern.

Sie konvertieren den HTML-Code in eine baumartige Struktur. [9]
Zur Extraktion der Daten ist es wichtig, dass sie eindeutig erkennbar sind. Die benötigten Informationen können beispielsweise mithilfe der Developer Tool-Ansicht einer Webseite untersucht werden und somit beispielsweise über einen eindeutigen Klassennamen oder einer ID identifiziert werden. Dabei sind die Daten auf vielen Webseiten nicht in dem benötigten Format und müssen zuerst bereinigt werden. Beispielsweise wird die durchschnittliche Anzahl an Sternen der Bewertungen eines Restaurants auf TripAdvisor (vgl. Abbildung 2.1) als Bild dargestellt und nicht als Zeichenkette. [9]

Abbildung 2.1: Beispiel unbereinigte Daten
[10]

Sobald alle Daten extrahiert und angepasst wurden, werden diese in einer dem Zweck angemessenen Form gespeichert. Diese Sicherung kann in Form von XML, JSON, CSV oder in einer Datenbank erfolgen. (vgl. Abbildung 2.2) Somit ist eine weitere Verarbeitung der Daten möglich. [9]

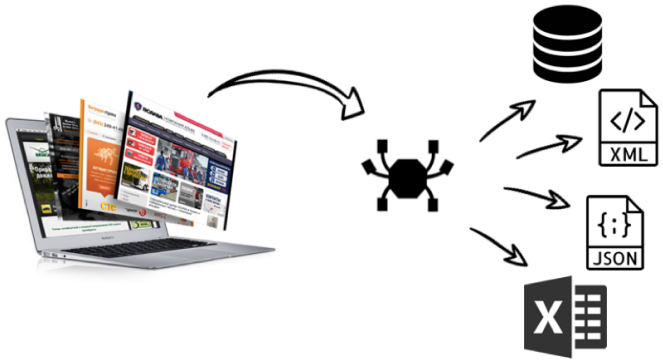

Abbildung 2.2: Funktionsweise eines Web Scrapers

2.2.2 Verwendung

Aktuelle Softwares und Tools sind in der Lage, komplette Webseiten in strukturierte Daten umzuwandeln, um diese weiter zu verarbeiten. Dies kann in vielerlei Hinsicht nützlich sein. Denn alle Vorgänge, die manuell im Internet erfolgen, können somit automatisiert werden. Beispielsweise können Tickets für ein Konzert gekauft werden, sobald diese verfügbar sind, oder einen Onlineshop regelmäßig überprüfen, ob der Preis für einen bestimmten Artikel gesunken ist. [9]
In der folgenden Studienarbeit wurden die Informationen der Seite „TripAdvisor.de“ analysiert. Dieses Unternehmen wird im folgenden Kapitel näher betrachtet.

2.3 Tripadvisor

TripAdvisor ist eines der bekanntesten Bewertungsportale in der Touristik- und Hotelbranche. Auf dieser Webseite können User über 7,5 Millionen Restaurants, Unterkünfte, Airlines und Erlebnissen bewerten und vergleichen. Diese Bewertungen können sie zusätzlich mit Bildern veranschaulichen. Somit wird den Reisenden die Planung ihrer Reise vereinfacht, da sie mithilfe von echten Bewertungen einen authentischen Eindruck erhalten. Zusätzlich bietet die Webseite im Bereich „Unterkünfte“ die Möglichkeit an, Zimmerpreise zu prüfen und diese direkt zu buchen. Mit insgesamt über 630 Millionen Bewertungen und Erfahrungsberichten der Reisenden und einem jährlichen Umsatz von mehr als 1,5 Milliarden Euro im Jahr 2017 [11] ist TripAdvisor die weltweit größte Touristikwebseite. Sie ist auf 49 Märkten vertreten. [12]

Abbildung 2.3: TripAdvisor Logo
[13]

Jedoch ist diesen Bewertungsportalen nicht immer Glauben zu schenken. Mittlerweile gibt es Dienstleister, die für gute Bewertungen bezahlt werden. Denn TripAdvisor überprüft nur stichpunktartig, ob der Verfasser der Bewertung sich auch wirklich dort aufgehalten hat. [14]

3 Anforderungen

Der Anwender der Software soll in der Lage sein, sich ein auf dem Reiseportal „Tripadvisor“ (tripadvisor.com) verzeichnetes Restaurant auszusuchen und aus dieser Repräsentanz des Restaurants hinterlegte Informationen, insbesondere über Bewertungen von Usern des Portals, zu extrahieren. Diese Informationen sollen strukturiert auf einer beliebigen Datenbank persistiert und vom Anwender einsehbar sein.
Im Folgenden werden die Anforderungen eingeteilt in funktionale und nicht–funktionale Anforderungen und werden näher erläutert.

3.1 Funktionale Anforderungen

Zu erfassende Daten Die zu extrahierenden Daten können in 2 Kategorien eingeteilt werden. Die 1. Kategorie besteht in Informationen zum konkreten Restaurant. So wie Name, Adresse, durchschnittliche Bewertungspunktzahl und weitere, wie aufgeführt in Tabelle 3.1.

Die 2. Kategorie besteht aus den Bewertungen, die User auf dem Portal für dieses konkrete Restaurant abgegeben haben. Hier bestehen die Informationen von Interesse laut Anforderung aus dem Titel der Bewertung, dem Text der Bewertung, der abgegebenen Punktzahl sowie insbesondere der mit der Bewertung mitgesendeten Bilder. Von diesen wird laut Anforderung lediglich verlangt die auf die jeweiligen Bilder verweisenden URLs als Datenpunkt zu erfassen und nicht die Bildinformation selbst.

Datenbank Anforderungen Nach der Extraktion von tripadvisor sollen die erfassten Daten persistiert werden um zu einem späteren Zeitpunkt für weitere Zwecke ohne großen Aufwand wieder verfügbar zu sein. Dies soll durch Eintragung in einer Datenbank realisiert werden. Weitere Anforderungen an diese Datenbank bestehen in:

Funktionale Anforderungen			Nicht-funktionale Anforderungen
Zu erfassende Daten	**Datenbank**	**Ausführbarkeit**	**Usability - Benutzerfreundlichkeit**
RESTAURANTDATEN:	Daten persistieren	Ausführbar über Kommandozeile	Erstellung einer Benutzeroberfläche (GUI)
Restaurantname (String)	Integritätssicherung	Eingabe einer Restaurant-URL	Angabe der URL über Eingabefeld in GUI
Durchschnittspunktzahl der Bewertung (Number)	Redundanzarmut		Funktionalität zur Vorschau auf das Restaurant hinter der URL
Anzahl der Bewertungen insgesamt (Number)	Datenunabhängigkeit		Prüfung der URL vor Starten des Scraping-vorgangs
Rang des Restaurants unter allen Restaurants der betreffenden Stadt (Number)	Zentrale Kontrolle		Abfrage ob bereits ein Datensatz für das gewählte Restaurant besteht
Preisniveau (String)			Starten der Extraktion
Küche oder Typ des Restaurants (String)			
Straße und Straßennummer des Restaurants (String)			
Postleitzahl und Ort des Restaurants (String)			
Telefonnummer (String)			
REVIEWDATEN:			
Name des Verfassers (String)			
Anzahl der Bewertungen des Verfassers (String)			
Anzahl der Likes des Verfassers(String)			
Titel des Reviews (String)			
Text des Reviews (String)			
Bewertung des Reviews (Number)			
Alle Urls der Bilder in einem Review (String)			

Abbildung 3.1: Anforderungsübersicht an das zu erstellende Programm

- Integritätssicherung – die Gewährleistung der Korrektheit der Datensätze auf der Datenbank durch entsprechende Prüfmaßnahmen bei Manipulation bzw. Eintragung von Datensätzen
- Redundanzarmut – Minimierung von doppelt geführten Datenpunkten oder -sätzen, sofern die Doppelung keine Mehrinformation beinhaltet
- Datenunabhängigkeit – das bearbeiten oder verwalten der in der Datenbank geführten Daten soll möglich sein ohne weitere Serverinfrastruktur o.ä.

- Zentrale Kontrolle – ein Administrator soll in der Lage sein, die gesamte Datenbank zu verwalten

Ausführbarkeit des Anwenders Der Empfänger bzw. Anwender der Software soll in der Lage sein die Programmfunktionalität zu starten und zu Beenden ohne weitere Abhängigkeiten. Als Voraussetzung wird hier angenommen, dass der Anwender über einen für den Stand der Zeit durchschnittlichen Rechner verfügt, auf dem es möglich ist ein Programm in der Kommandozeile des jeweiligen Betriebssystems auszuführen.

Eingabe einer Restaurant-URL Der Anwender soll in der Lage sein dem Programm einen Parameter zu übergeben, der das Restaurant, dessen Daten extrahiert werden sollen, identifiziert. Im vorliegenden Kontext soll dies über die Eingabe der Tripadvisor-URL des betreffenden Restaurants erfolgen.

3.2 Nicht funktionale Anforderungen

Usability – Benutzerfreundlichkeit
In Hinsicht darauf, dass die Anwender der Software nicht ausschließlich IT-affine Personen sein werden, soll die Ausführung der Software über eine einfach zu verstehende Graphische Benutzeroberfläche (GUI) erfolgen.
Die Eingabe der URL soll über ein Textfeld in der GUI gewährleistet werden. Bevor der Vorgang der Datenextraktion für das Restaurant und alle seine Bewertungen gestartet wird, soll es möglich sein über einen Button eine Validierung der angegebenen URL durchzuführen. Diese überprüft, ob die URL, zumindest Ihrer Form nach, eine gültige Tripadvisor-Restaurant-URL ist, und sicherstellen, dass das Programm nicht aus dem Grund fehlschlägt, dass die angegebene URL nicht die eines Tripadvisor-Restaurants ist.
Anschließend an die Validierung sollen die Basisinformationen Name, PLZ und Ort sowie durchschnittliche Bewertung des mit der URL verbundenen Restaurants angezeigt werden. So kann sich der Anwender vor Starten des bis zu mehreren Minuten langen Vorgangs versichern, dass er auch wirklich die URL des gewünschten

Restaurants angegeben hat.
Letztendlich soll es möglich sein durch einen Button die Extraktion und Persistierung der Daten zu starten. Vor Starten des eigentlichen Vorgangs soll eine Prüfung erfolgen, ob das gewählte Restaurant bereits extrahiert wurde. Falls ja wird der Anwender entsprechend informiert und der Vorgang wird abgebrochen.

4 Implementierung

Im folgenden Kapitel wird die voll umfängliche Implementierung des zu entwickelnden Web Scrapers beschrieben. Dabei wird zunächst auf die Konfiguration der Arbeitsumgebung eingegangen und die Hauptbibliothek von BeautifulSoup näher erläutert. Das Unterkapitel 'Implementierung Web Scraper' verdeutlicht anhand von Codebeispielen die Integration von Bibliotheken und die Extraktion der Rohdaten. Darauffolgend wird im Kapitel 'Datenhaltung' die Inklusion der Datenbank skizziert und im Abschluss die zum Programm entworfene Benutzeroberfläche erläutert.

4.1 Konfiguration

Zunächst wurde als Entwicklungsumgebung die aktuellste Version von Eclipse (Version 4.7.*) sowie als Programmiersprache die neueste Version von Python und PIP zur Verwaltung von Paketen installiert. Es wurde bewusst nicht die Version 2.7 verwendet, da ab der Version 3.0 Pakete, wie JSON und urllib, schon vorinstalliert sind. Es wurde Python gewählt, da dies die bekannteste Sprache für Web Scraper ist und Frameworks, wie Scrapy, Beautifulsoup, etc. in Python geschrieben sind.
Um Python für Eclipse zu nutzen, wurde das Plugin „PyDev“ in Eclipse installiert. Daraufhin muss der Python-Interpreter konfiguriert werden. Diese Einstellung wird unter Window > Preferences vorgenommen. [18]
Für den Web Scraper wurden zunächst zwei wichtige Pakete mithilfe von PIP installiert: Requests und Beautifulsoup. Um die URL zu öffnen und den HTML Code der Webseite herunterzuladen, wird das Paket „Requests“ benötigt. Um diesen Code zu analysieren und die Daten zu extrahieren, wurde das Framework „Beautifulsoup“ installiert. [19] Warum sich in dieser Studienarbeit für dieses Framework entschieden wurde, wird im nächsten Abschnitt näher erläutert.

4.2 BeautifulSoup

Für die Programmierung des Web Scrapers, um Daten von auf TripAdvisor gelisteten Restaurants zu scrapen wurde die Python Library 'BeautifulSoup' (*bs4*) verwendet. Hierbei handelt es sich um eine Library für XML und HTML parsing, die zunächst als zusätzliches Paket installiert werden muss. Beim Parsen werden XML oder HTML Dateien analysiert und auf ihre einzelnen Bestandteile geprüft. So können jegliche Informationen aus diesen Dateien gesucht bzw. extrahiert werden, vorausgesetzt es sind entsprechende Tags bzw. Attributnamen vorhanden. Ohne diese kann nicht gesucht werden [15]. Diese Einfachheit und gleichzeitig große Effizienz sind der westenliche Vorteil, warum das Projekt-Team sich für die Library BeautifulSoup und deren Methoden entschieden hat.

Um das wahrscheinlich am meisten verwendeten Objekt der BeautifulSoup Library – das BeautifulSoup Object – zu initialisieren, muss diesem ein XML bzw. HTML Dokument oder, wie in unserem Fall, der zu parsenden HTML-Code übergeben werden. BeautifulSoup kann eine URL also nicht selbstständig aufrufen, sondern erst mit dem gespeicherten Quelltext der HTML-Seite arbeiten [16]. In unserem Fall wird die TripAdvisor URL des gewünschten Restaurants daher zunächst mit einer Methode der Python Library *'request'* aufgerufen. Diese sendet einen HTTP/1.1 request und pingt die Webseite an, dass Informationen extrahiert werden möchten. Sobald der Webserver eine *response* sendet wird der Inhalt der URL, d.h. der HTML-Code in einer einfachen String Variable gespeichert. Erst jetzt kann ein neues BeautifulSoup Object initialisiert werden (vgl. Programmcode 4.1).

```
1        def get_single_data(self,url):
2        source_code = requests.get(url)
3        plain_text = source_code.text
4        soup = BeautifulSoup(plain_text, "html.parser")
```

Programmcode 4.1: Initialisierung BeautifulSoup Object

Dieses transformiert den komplexen HTML-Code in einen ebenfalls komplexen Baum mit Python Objekten. Die HTML-Tags korrespondieren mit den neuen Tag Objekten des BeautifulSoup Objects [16]. Am folgenden Beispiel kann die Struktur eines solchen beispielhaft veranschaulicht werden.

```
<html>
    <head>
        <title> ... </title>
    </head>
    <body>
        <h1> ... </h1>
        <div>
            <a> ... </a>
            <span> ... </span>
        </div>
    </body>
</html>
```

Abbildung 4.1: Baum mit Python Objekten [15]

Natürlich enthält dieser normalerweise eine weitaus größere Anzahl an Vererbungen, d.h. ineinander verschachtelten Tags, und ist daher komplizierter zu lesen. Jedoch wird anhand dieses Beispiels bereits deutlich, wie die im folgenden beschriebenen Methoden der BeautifulSoup Library damit arbeiten können.
Sobald das BeautifulSoup Object vorliegt, können die vorhandenen Methoden der bs4 Library genutzt werden. In folgender Tabelle sind die beiden bekanntesten und auch im Programm-Code am meisten genutzten Methoden aufgelistet [16].

Methode	Beschreibung	Ergebnis
Soup.find(x)	Suche nach erstem Treffer für gewähltes Tag x und gebe dieses aus	Erster Treffer Bei keinem Ergebnis: *none*
Soup.find_all(x)	Suche nach allen Treffern für gewähltes Tag x und gebe diese in einer Liste aus	Liste mit allen Treffern Bei keinem Ergebnis: Leere Liste []

Abbildung 4.2: Such-Methoden der Python Library BeautifulSoup [16]

Im folgenden Kapitel werden diese genutzt um den Web Scraper für TripAdvisor erfolgreich zu implementieren.

4.3 Implementierung Web Scraper

Im folgenden wird die Implementierung des Web Scrapers anhand von konkreten Code-Beispielen im Detail beschrieben. Besonders wird dabei auf die einfache und intuitive Verwendung der Bibliothek 'BeatifulSoup' eingegangen.

4.3.1 Initialisierung BeautifulSoup

Zunächst wird die URL des gewünschte Restaurants, für welches Daten extrahiert werden sollen, an die Methode *get_single_data()* übergeben. Diese requested zu Beginn die übergebene URL, speichert ihren Source Code und initialisiert ein BeautifulSoup Object (vgl. Kapitel 4.2). Jetzt kann die eigentliche Implementierung des Web Scrapers beginnen.

4.3.2 Extraktion Allgemeiner Daten

Zunächst sollen die in Kapitel 3 definierten allgemeinen Daten eines Restaurants extrahiert werden. Zu diesen gehören Name, Kontaktdetails, Küche, Preis-Level, Popularität, Anzahl an Bewertungen sowie die durchschnittlichen Bewertung. Dies soll in der neu definierten Methode *get_single_data()* mit den bereits beschriebene Suchfunktionen (Kapitel 4.2) umgesetzt werden.
Die Unkompliziertheit von BeautifulSoup wird auch hier wieder deutlich. Mit *find()* und *find_all()* kann nach jedem gewünschten Tag im Source Code gesucht und dessen Inhalt ausgegeben werden. So kann nach Strings, Blockelementen (z.B. -Tag, Tabellen, Listen oder Überschriften), Inline-Elementen (z.B. -Tag, Links oder Bilder) und weiteren Tags sowie deren Klassennamen gesucht werden [16]. Um diese zu bestimmen, ist es notwendig den übergebenen HTML-Code der URL genauer zu betrachten. Dies ist z.B. im Browser Chrome mit Rechtsklick „Quellcode anzeigen“ möglich. Um direkt auf ein gewünschtes Element im Quellcode zu springen, kann Rechtsklick „Element untersuchen“ genutzt werden. Diese Developer Tool-Ansicht ist deutlich strukturierter als den gesamten Text anzeigen zu lassen und erleichtert die Suche nach den gewünschten Tags um ein Vielfaches.

Der Programmcode 4.2 zeigt einige der Elemente und Klassen die gewählt wurden, um die gewünschten Informationen zu extrahieren. Um den Namen des Restaurants zu finden wird beispielsweise nach dem Tag <*h1*> und dem class-Namen *'heading-title'* gesucht. Dieser Tag steht üblicherweise für Überschriften und kommt meist nur einmal vor. Daher ist die Suche hier relativ simpel. Hingegen wird für die Durchschnittliche Bewertung des Restaurants nach einem <*div*>-Tag mit class-Namen *'rs rating'* gesucht. Da es sich dabei jedoch nur um die parent-class des gesuchten Tags handelt wird hier noch mit den Anhängen .div.span erweitert und tiefer in das BeautifulSoup Object „gezoomt" um die children dieser Klasse zu erhalten. Mit der ID *'content'* werden letztendlich die durchschnittliche Bewertung gefunden und in der Variable overallPoints gespeichert.

```
1            for self.name in soup.find('h1', {'class': 'heading_title'}):
2            print("Name:" + self.name.string)
3
4            for self.rating_amount in soup.find('span', {'property': 'count'}):
5            print("Anzahl Bewertungen:" + self.rating_amount.string)
6
7            for self.cuisine in soup.find_all('span', {'class': 'header_links
                 rating_and_popularity'}):
8            print('Kueche:' + self.cuisine.text)
```

Programmcode 4.2: Extraktion Allgemeiner Daten Object

BeautifulSoup parsed so den gesamten HTML-Code nach den definierten Tag's und gibt je nach Methode das erste bzw. alle gefundenen Ergebnisse der Suche aus. Beispielsweise wird beim Namen nur nach einem Ergebnis gesucht, da hier mehrere Ergebnisse schlichtweg nicht sinnvoll wären. Bei der Art der Küche hingegen wird die Methode *find_all()* genutzt, da hier auch mehrere Ergebnissen gefunden werden können (vgl. Programmcode 4.2). Alle Ergebnisse werden wiederum in neu definierten Variablen als einfache Strings bzw. bei mehreren Ergebnissen in Listen gespeichert.

4.3.3 Bewertungslink aller Bewertungen auslesen

Neben allgemeinen Daten zum Restaurant sollen außerdem noch Informationen zu allen Bewertungen inkl. angehängter Bilder und Daten zu deren Verfassern extrahiert und gespeichert werden. Dafür wird zunächst die URL jeder Bewertung generiert werden, um die Informationen von hier auszulesen. Da sich die Bewertungen meist nicht nur über eine, sondern eine Vielzahl an Seiten erstrecken wurde dies mit einer Verschachtelten for-Schleife gelöst.
Zunächst wurde eine for Schleife definiert, die alle verfügbaren Seiten eines Restaurants auf TripAdvisor durchläuft (vgl. Programmcode 4.3). Dafür wird die maximale Anzahl an Seiten mit *find_all()* gesucht und eine neue URL für jede der Seiten generiert. Diese unterschieden sich lediglich anhand eines Indexes (in 10er Schritten). Daher wird dieser für jeden Schleifendurchlauf, d.h. für jede Seite um 10 erhöht und mit der vorher gesplitteten URL neu zusammengesetzt.

```
1            #create a new_url for each page of the reviews
2            for index in range (10,pages*10, 10):
3
4            data = url.split("Reviews-")
5            new_url = data[0]+"Reviews-or"+str(index)+'-'+data[1]
6            #pass the new_url to the loop_trough_review_pages function
7            self.loop_through_review_pages(new_url)
```

Programmcode 4.3: For-Schleife über alle verfügbaren Bewertungsseiten

Die neu generierte URL wird jedes Mal an die Methode *loop_through_review_pages()* übergeben.
Mit dem ersten Aufruf von *find_all()* in dieser Mehtode wird zunächst nach dem <*div*>-Tag mit class-Namen *'review container'* gesucht (vgl. Programmcode 4.4).

Bei einem solchen container handelt es sich um einen Block im HTML-Code, der alle Informationen einer einzigen Bewertung umfasst und diese sozusagen „einrahmt". Für jede einzelne Bewertung wird dieser daher wiederholt (vgl. Abbildung 4.3). Da die Methode *find_all()* genutzt wurde, wird deshalb in der Variable *review_containers* nun eine Liste mit allen ausgelesenen containern gespeichert.

```
1           def loop_through_review_pages(self, loop_url):
2           source_code = requests.get(loop_url)
3           plain_text = source_code.text
4           soup = BeautifulSoup(plain_text, "html.parser")
5
6           #get all review containers but on the nuw_url
7           review_containers = soup.find_all('div', class_= 'review-container')
8           reviewsPerPage = len(review_containers)
9
10          #search trough all reviews on the page
11          for container in review_containers:
12
13          for link in container.find_all('a', href=True):
14          href = "https://www.tripadvisor.de" + str(link.get('href'))
15          #pass it to the function where single data is scraped
16          self.get_single_review_data(href)
```

Programmcode 4.4: Methode loop_through_review_pages

Mit dem erneuten Aufruf der Methode *find_all()* wird in einem container nach einem Link (<*a*>-Tag) gesucht und gleichzeitig href auf '*True*' gesetzt. So stellt man sicher, dass nur Links mit href-Attribut ausgegeben werden. Bei diesem Link handelt es sich um den Teil der URL, der an die reguläre TripAdvisor Webseite angehängt werden muss, um die Bewertungsseite aufrufen zu können.

Da jedoch nicht nur Informationen zur ersten gefunden Bewertung pro Seite, sondern zu allen Bewertungen extrahiert werden sollen, ist die Suchfunktion in eine weitere for-Schleife eingebunden. Diese loopt durch die vorher gespeicherte Liste *review_container*. So wird nicht nur der erste review container nach der Bewertungsseiten URL durchsucht sondern alle. Die Methode *loop_through_review_pages()* in

```
▶<div class="review-container" data-reviewid="507673008" data-
collapsed="true" data-deferred="true">…</div>
  <div class="info hidden" data-expansionsoftgate="false"></div>
▶<div class="review-container" data-reviewid="507416214" data-
collapsed="true" data-deferred="true">…</div>
  <div class="info hidden" data-expansionsoftgate="false"></div>
▶<div class="review-container" data-reviewid="501669962" data-
collapsed="true" data-deferred="true">…</div>
  <div class="info hidden" data-expansionsoftgate="false"></div>
```

Abbildung 4.3: Wiederholung der 'review container' [17]

Kombination mit der for-Schleife über alle Bewertungsseiten ermöglicht nun die Bewertungslinks jeder einzelnen Bewertung auszulesen.
Diese neu ausgelesene URL wird bei jedem Durchlauf an die Methode *get_single_review_data()* übergeben. Mit dieser können alle in Kapitel 3 definierten Daten zu den Bewertungen extrahiert werden.

4.3.4 Extraktion Bewertungensdaten

Die Daten der einzelnen Bewertungen werden mithilfe der oben genannten Methode *get_single_review_data()* extrahiert. Dazu gehören wie in Kapitel 3 beschrieben: der Verfasser, seine Anzahl an Bewertungen und Likes, der Titel und der Text sowie die URL der Bilder einer Bewertung, falls dieser welche hochgeladen hat. Zunächst wird wieder ein BeautifulSoup Object initalisiert, jedoch diesmal mit dem gespeicherten HTML-Code einer einzelnen Bewertungsseite (vgl. Programmcode 4.5).

```
1            def get_single_review_data(self,review_url):
2            source_code2 = requests.get(review_url)
3            plain_text2 = source_code2.text
4            soup = BeautifulSoup(plain_text2, "html.parser")
```

Programmcode 4.5: Initialisierung BeautifulSoup Object

Die allgemeine Schwierigkeit an der Extraktion der Bewertungsdaten lag daran, dass auf der Webseite jeder einzelnen Bewertung zusätzlich noch die folgenden vier Bewertungen angezeigt werden. Dadurch, dass jede Bewertung den gleichen Aufbau an Klassen besitzt, war es nicht möglich nur nach dem Klassennamen zu suchen, da sonst beispielsweise die Überschriften aller Bewertungen ausgegeben worden wären. Da der Name des Verfassers der Bewertung an erster Stelle im Quellcode angezeigt wird, kann dieses Element lediglich mithilfe der Methode *find()* und dem Klassennamen ausgegeben werden. Die Elemente „Anzahl an Bewertungen“ sowie „Anzahl an Likes“ besitzen jedoch denselben Klassennamen. Daher wurde in diesem Fall die Methode *findAll()* benutzt, allerdings mit der Kombination, dass nur ein bestimmtes Element ausgegeben werden soll (vgl. Programmcode 4.6)

```
1            for self.numberOfReviews in soup.findAll('span', {'class': 'badgetext'})[0]:
2            print("Anzahl Reviews: " + self.numberOfReviews.string)
3
4            for self.numberOfLikes in soup.findAll('span', {'class': 'badgetext'})[1]:
5            print("Anzahl Likes: " + self.numberOfLikes.string)
```

Programmcode 4.6: Anzahl an Bewertungen und Likes des Verfassers

Somit wird für die Anzahl an Bewertungen der Text des ersten gefundenen Klassennamens ausgegeben, für die Anzahl an Likes der Text des zweiten Klassennamens. Um den Titel sowie den Text der Bewertung zu extrahieren, wurde das JSON des heruntergeladenen HTML-Codes wieder in ein JSON-Format geparst (vgl. Programmcode 4.7). Denn die Problematik hierbei war, dass wenn die Bewertung in einer anderen Sprache verfasst worden ist, die Überschrift und der Text der Bewertung automatisch ins Deutsche übersetzt worden sind. Dadurch erkennt das System die übersetzte Überschrift sowie den Text anhand des Klassennamens nicht. Mithilfe dieses JSONs wurde zusätzlich die Anzahl an vergebenen Punkten herausgelesen.

```
1            self.item_name = soup.find('script', {'type': 'application/ld+json'})
2            json_string = str(self.item_name.string)
3            obj = json.loads(json_string)
```

Programmcode 4.7: Transformation in ein JSON-Format

Des Weiteren war die Anforderung, die URL der Bilder einer Bewertung abzuspeichern. Hierbei war die Schwierigkeit, dass auch Bewertungen ohne Bilder existieren. Des Weiteren können auch die anderen angezeigten Bewertungen auf der Webseite Bilder enthalten. Dadurch war es nicht möglich nur nach der Klasse *inlinePhotosWrapper* zu suchen. Somit hätte das System auch die URLs der anderen Bewertungsbilder ausgegeben. Daher sucht der Web Scraper zunächst, nach einer ID, die nur die gewünschte Bewertung besitzt. Daraufhin überprüft das System, ob es innerhalb dieser ID eine Klasse *inlinePhotosWrapper* existiert, also ob der Verfasser der Bewertung Bilder hochgeladen hat. Ist dies der Fall, werden die URLs der Bilder in der Datenbank abgespeichert. Andernfalls wird eine textuelle Nachricht ausgegeben, dass für diese Bewertung keine Bilder hochgeladen wurden.

```
1           for link in soup.findAll('div', {'id': '
                taplc_location_reviews_list_sur_callout_0'}):
2           if link.find('div', {'class': 'inlinePhotosWrapper'}):
3           for pic in link.findAll('img', {'class': 'centeredImg'}):
4           self.src = pic.get('src')
5           print("Quelle Bild " + self.src)
6           else:
7           print("No pictures have been uploaded for this review!")
```

Programmcode 4.8: Extraktion der URL der Bewertungsbilder

Alle Ergebnisse werden wiederum in neu definierten Variablen als einfache Strings bzw. bei mehreren Ergebnissen in Listen gespeichert.

4.4 Datenhaltung

Eine Anforderung an das System besteht auch darin die gewonnenen Daten in einer Datenbank ablegen zu können. Zunächst wurden vom Product Owner die nötigen Ansprüche an die Datenbank bereitgestellt, nach der sich dann aus zwei Datenbanktypen eine vorteilhaftere herauskristallisieren konnte.
Da die gewonnen Rohdaten sich noch meist in einem encodierten Zustand befinden, müssen diese zunächst einen Bereinigungsprozess durchlaufen bevor sie leserlich in der Datenbank abgelegt werden können. Nicht nur das einfache Lesen der Daten sondern auch die Suche nach bestimmten Rechercheergebnissen soll möglichst intuitiv für den Nutzer ablaufen. Die Datenbankstruktur wurde daher Zusammen mit dem Auftraggeber abgestimmt und daraufhin die Architektur implementiert.

4.4.1 Definition der Datenbankanforderungen

Die vom Abnehmer vordefinierten Anforderungen an die Datenbank sind wie folgt:

- Integritätssicherung

  Daten werden auf Korrektheit (bereits während der Eingabe) überprüft und Fehlmanipulationen verhindert

- Redundanzarmut

  es gibt keine ungeordnete Mehrfachspeicherung von Datenwerten

- Datenunabhängigkeit
  die Datenbank kann ohne Server verwaltet und weiterentwickelt werden
- zentrale Kontrolle
  ein Administrator ist in der Lage, das gesamte System von einem Rechner aus zu verwalten

4.4.1.1 Vergleich: SQLLite und TinyDB

Im Folgenden werden zwei NoSQOL Datenbanktypen miteinander verglichen, die dem Entwicklerteam als am sinnvollsten erschienen.

TinyDB:
TinyDB ist eine leichtgewichtige dokumentenorientierte Datenbank, die für eine einfache Bedienung optimiert ist. Dabei ist sie in reinem Python geschrieben und hat keine externen Abhängigkeiten. Das Ziel sind kleine Apps, die von einer SQL-DB oder einem externen Datenbankserver entkoppelt funktionieren. [4]

SQLite:
SQLite ist eine gemeinfreie Programmbibliothek, die ein relationales Datenbanksystem enthält. Sie unterstützt einen Großteil der im SQL-92-Standard festgelegten SQL-Sprachbefehle. Die SQLite-Bibliothek lässt sich direkt in entsprechende Anwendungen integrieren, sodass keine weitere Server-Software benötigt wird. Dies ist der entscheidende Unterschied zu anderen Datenbanksystemen. Durch das Einbinden der Bibliothek wird die Anwendung um Datenbankfunktionen erweitert, ohne auf externe Softwarepakete angewiesen zu sein. [5]

Der wesentliche Vorteil von TinyDB liegt darin, dass diese Datenbank bereits in Python geschrieben ist und sich somit ideal in unseren Source-Code integrieren lässt. Es entstehen dabei keine Inkonsistenzen oder fehlerhafte Bibliotheksaufrufe. Ein weiteres Plus ist die Möglichkeit das Dateiformat JSON als Datenbank zu verwenden. Dieses kann im Vergleich zur SQLite Datei in sehr vielen Anwendungen integriert werden. Fortwährend ist man zu der Entscheidung gekommen, dass

das Programmieren in SQL-Sprache in einem Python-Code zu Verwirrungen und unnötigen Programmier-Anforderungen beiträgt, was bei SQLite der Fall ist.
Für das Projekt wurde sich einstimmig für die Verwendung von TinyDB als Datenbanksystem entschieden.

4.4.2 Datenbankstruktur

Die extrahierten Daten können in grob vier Cluster eingeteilt werden, dabei bildet der erste Cluster die allgemeinen Daten des Restaurants, wie dessen Namen und Adresse. Den zweiten und dritten Cluster kann zusammenfassend als Benutzerdatenbank und Bewertungstext sehen. Der vierte Cluster bildet die Tabelle in der die Bilder abgespeichert werden, die zu einem jeweiligen Kommentar hinterlegt wurden.

4.4.2.1 Tabellen mit Spalten

Für die bessere Übersicht wurden in der Datenbank vier Tabellen angelegt, die wie folgt in ihre jeweiligen Spalten zerlegt sind:
Cluster eins, Restaurant-Daten:

R_ID	RESTAURANTNAME	PUNKTESKALA	ANZAHL_BEWERTUNGEN	POPULARITAET
atoincrement ID des Restaurants	Name des Restaurants	Gesamtbewertung	Anzahl abgegebener Bewertungen	Beliebtheitsgrad

Tabelle 4.1: Tabelle der Restaurant-Daten Teil 1

Fortsetzung:

PREIS_LEVEL	KUECHE	STRASSE	PLZ_ORT	TELEFONNUMMER
Preislevel zw. 1 und 4	Küchennationalität	Straßenname	Postleitzahl und Ort	Festnetznummer

Tabelle 4.2: Tabelle der Restaurant-Daten Teil 2

Der zweite Cluster enthält sämtliche Daten von Benutzern, die eine Bewertung zum Restaurant abgegeben haben. Dieser teilt sich in folgende Tabellenstruktur auf:
Cluster drei enthält den Bewertungstext und die dazugehörigen Attribute wie Titel und Rating:

U_ID	REV_ID	BENUTZERNAME	ANZAHL_REVIEWS
autoincrement ID des Users	autoincrement ID des Bewertungstextes	Benutzername	Anzahl der abgegebenen Bewertungen

Tabelle 4.3: Tabelle der Benutzer-Daten

R_ID	U_ID	REV_ID	TITEL	BEWERTUNG	RATING
autoincrement ID des Restaurants	autoincrement ID des Benutzers	autoincrement ID der Bewertung	Titel	Bewertungstext	Rating des Textes

Tabelle 4.4: Tabelle der Bewertungs-Daten

Der vierte Cluster enthält die direkten Links zu den hinterlegten Kommentar-Bildern:

REV_ID	QUELLE
autoincrement ID der Bewertung	Link des Bildes

Tabelle 4.5: Tabelle der Kommentar-Bilder

4.4.2.2 Zeichnung und Zusammenhang

Die Datenbank kann im angelegten Schema, wie es in Kapitel 4.4.2.1 beschrieben ist, über die autoincrement ID's der jeweiligen Tabellen durchsucht und verknüpft werden. Beim Anlegen der autoincrement ID's wurde darauf geachtet, dass keine ID mehrfach vorkommt. So kann dem Anwender eine verwechslungsfreie Suche garantiert werden.
Ein Bewertungstext, der zu einem bestimmten Restaurant abgegeben wurde, kann beispielsweise mit folgendem Python-Code aufgerufen werden:

```
1 result = db.search((where('RESTAURANTS')['R_ID']) == (where('REVIEWS')['R_ID
      ']))
2 print(result)
```

Programmcode 4.9: Datenbanksuche: Reviews passend zum Restaurant

4.4.3 Daten bereinigen und integrieren

Meist sind die Daten auf den Webseiten bereits kodiert im UTF-8 Muster hinterlegt. UTF-8 (Abk. für 8-Bit UCS Transformation Format, wobei UCS wiederum Universal

Character Set abkürzt) ist die am weitesten verbreitete Kodierung für Unicode-Zeichen. Es ist in den ersten 128 Zeichen (Indizes 0–127) deckungsgleich mit ASCII und eignet sich, mit in der Regel nur einem Byte Speicherbedarf, für Zeichen vieler westlicher Sprachen besonders für die Kodierung englischsprachiger Texte. [3] Die bei TripAdvisor angewendete Kodierung ist in dem Fall auch UTF-8 und muss neu decodiert werden, um die deutschen Umlaute korrekt anzeigen zu können.

```
1            #HANDLE RESTAURANTS
2            #check if there already exists an entry in the Restaurans-table with the same
                 name and the same address (these two attributes don't change/are not
                 variable so there is a more constant way to check for duplicates)
3            if tableRestaurants.contains((where('RESTAURANTNAME') == self.name.string) & (
                 where('PLZ_ORT') == self.locality.string)):
4            #pop up a message box
5            msg = "Dieses Hotel hast du bereits gesucht und ist in der Datenbank hinterlegt
                 !"
6            popup = tk.Tk()
7            popup.wm_title("!")
8            label = ttk.Label(popup, text=msg)
9            label.pack(side="top", fill="x", pady=10)
10           B1 = ttk.Button(popup, text="Okay und Beenden", command = self.endProgram)
11           B1.pack()
12           popup.mainloop()
13           # if there is no such entry, parse the data into the corresponding table of the
                  database and define the data to insert into database including an auto
                 increment ID for each Table
14           else:
15           self.popularity2 = unicodedata.normalize('NFD', self.pupularity.text)
16           print("Preis_Level: " + self.price_level)
17           dataRestaurants = {'R_ID':self.idRestaurant, 'RESTAURANTNAME':self.name.string,
                  'PUNKTESKALA':self.overallPoints.div.span['content'], 'ANZAHL_BEWERTUNGEN'
                 :self.rating_amount.string, 'POPULARITAET':self.popularity2, 'PREIS_LEVEL':
                 self.price_level, 'KUECHE':self.cuisine.text, 'STRASSE':self.address.string
                 , 'PLZ_ORT':self.locality.string, 'TELEFONNUMMER':self.phonenumber.text}
18           tableRestaurants.insert(dataRestaurants)
```

Programmcode 4.10: Rohdaten decodieren und in Datenbank integrieren

Nachdem die Rohdaten in Variablen abgelegt wurden, können diese in einer neuen Methode für den Reinigungsprozess weiterverwendet werden. Zunächst wird eine Datenbankabfrage gestartet, bei der überprüft wird, ob der neu-gefundene Eintrag bereits in der Datenbank existiert. Ist dies der Fall wird ein Pop-Up-Fenster geöffnet,

das beispielsweise bei der Restaurantabfrage einen Beenden-Button enthält, sodass nicht alle Daten zum Restaurant erneut eingefügt werden. Wird ein doppelter Eintrag hingegen bei einer Bewertung gefunden, kann das Pop-Up-Fenster einfach weg geklickt werden und das Programm läuft weiter. (vgl. Programmcode 4.7, Zeilen 3-12)
Ist kein Eintrag vorhanden, werden die Daten zunächst in einem Array gespeichert und dann in der Datenbank abgelegt. Zuvor findet noch eine Decodierung der Rohdaten statt (vgl. Programmcode 4.7, Zeile 15).

4.5 Graphische Benutzeroberfläche - GUI

Um der nicht-funktionalen Anforderung einer gewissen Usability über eine Graphische Benutzeroberfläche (GUI) nachzukommen wurde die Library „Tkinter“ genutzt. Tkinter ist ein integraler Bestandteil von Python3, ist „robust, schnell und plattformunabhänigig lauffähig“ [20]. Es bringt alle benötigten Elemente mit, wie Textfelder, Texteingabefelder und klickbare Buttons, die mit Programmfunktionalität verbunden werden können

4.5.1 Konzept

Abbildung 4.4 zeigt ein Mock-up der geplanten graphischen Benutzeroberfläche. Die einzelnen Elemente und das erwartete Verhalten seien im Folgenden beschrieben:

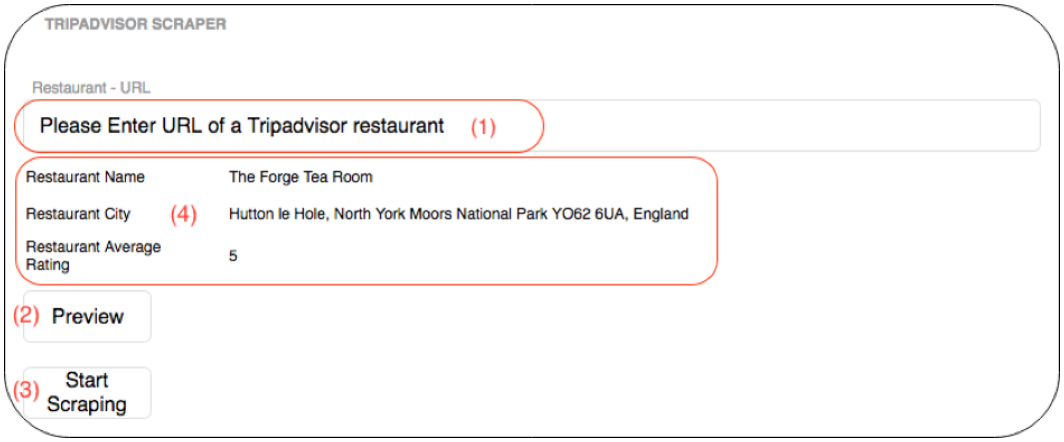

Abbildung 4.4: Mock-Up der geplanten Benutzeroberfläche

(1) URL-Eingabefeld

In dieses Feld fügt der Anwender die URL des Restaurants ein, dessen Informationen von Interesse sind.

(2) Preview-Button

Durch Klick auf diesen Button wird zunächst geprüft ob im Eingabefeld eine valide Tripadvisor-Restaurant-URL eingegeben wurde. Falls nein, soll eine entsprechende Fehlermeldung dem Nutzer angezeigt werden. Falls ja, soll im mit (4) markierten Bereich die Informationen Restaurantname, Ort und Durchschnittsbewertung angezeigt werden.

(3) Start Scraping-Button

Durch Klick auf diesen Button soll zunächst überprüft werden, ob das betreffende Restaurant bereits in der Datenbank erfasst wurde. Falls ja soll der Vorgang abgebrochen werden und der User erhält eine entsprechende Fehlermeldung. Falls nein wird die Extraktion und anschließende Persistierung in der Datenbank angestoßen.

4.5.2 GUI – Umsetzung

In Abbildung 4.5 ist die konkret implementierte GUI des im Rahmen dieser Arbeit erstellten Programms zu sehen. Sie wird bei Ausführung des Programms angezeigt. Das Layout des Mock-Ups konnte weitestgehend umgesetzt werden, das Verhalten wurde wie oben beschrieben umgesetzt. Im Folgenden soll nun die Implementierung der Funktionalitäten näher beschrieben werden.

Abbildung 4.5: Benutzeroberfläche des Tripadvisor Scrapers unter MacOS

Funktionalität - URL-Validierung Bei betätigen des Preview-Buttons wird der im Textfeld stehende String mit Hilfe der Library "validators" über RegEx darauf

überprüft, ob er dem Format einer URL im Allgemeinen entspricht. Als nächsten Schritt wird der String darauf geprüft ob er mit
"https://www.tripadvisor.de/Restaurant_Review" beginnt. Schlägt eine der beiden Prüfungen fehl wird ein Pop-Up mit entsprechender Fehlermeldung angezeigt. Ist die Prüfung erfolgreich erfolgt werden die Preview-Daten extrahiert

Funktionalität - Preview Hier wird ein Teilabschnitt des in Kapitel 4 beschriebenen Scraping Vorgangs über Beautiful-Soup durchgeführt, allerdings wird nicht der gesamte Vorgang durchgeführt sondern es werden aus dem Quellcode der URL lediglich Restaurantname, Ort und Postleitzahl sowie durchschnittliche Bewertungspunktzahl extrahiert und an entsprechender Stelle in der GUI angezeigt.

Funktionalität – Start Scraping Es erfolgt eine Prüfung ob es in der Datenbank einen Eintrag mit dem aus der URL-extrahierten Restaurantname und dessen Ort und Postleitzahl gibt. Falls ja wird ein Pop-up angezeigt mit entsprechender Nachricht und der Vorgang wird abgebrochen. Ist es noch nicht vorhanden wird die Extraktion der Restaurant- und Reviewdaten wie in Kapitel 4 angestoßen.

5 Web Scraper anhand eines Beispiels

Im Folgenden wird anhand eines Beispiels der Web Scraper näher erläutert. Es werden die Daten eines Restaurants in München extrahiert. Als Beispiel wurde dazu das Restaurant *Sollner Hof* gewählt.

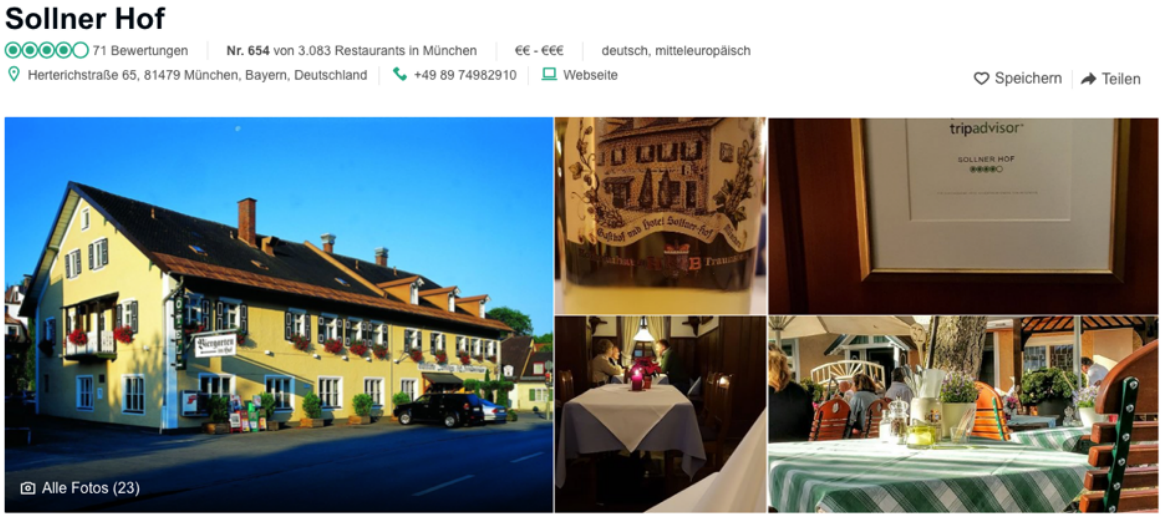

Abbildung 5.1: Screenshot des Restaurants Sollner Hof auf TripAdvisor.de [10]

Das Restaurant wurde insgesamt 71 Mal bewertet - davon sind 56 Kommentare verfasst worden. Die Durchschnittsbewertung liegt bei vier von fünf Sternen. Um die Daten des Restaurants zu extrahieren, muss zunächst die URL des Restaurants in die GUI eingetragen werden:

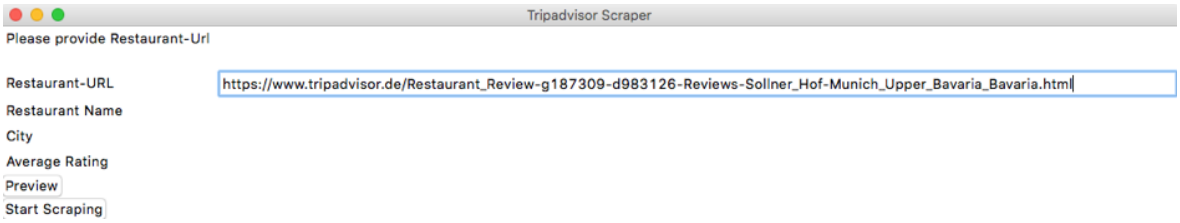

Abbildung 5.2: Benutzeroberfläche des Web Scrapers am Beispiel Sollner Hof

Mithilfe des Preview-Buttons kann zum einen überprüft werden, ob es sich um eine syntaktische korrekte URL der Webseite *TripAdvisor* handelt. Zum anderen kann der User sehen, ob es sich um das richtige Restaurant handelt. Dazu wird der

Name, die Stadt und die Anzahl an durchschnittlichen Bewertungspunkten auf der Benutzeroberfläche angezeigt:

Restaurant Name	Sollner Hof
City	81479 München, Bayern
Average Rating	4

Abbildung 5.3: Vorschau am Beispiel Sollner Hof

Wenn der User die Daten des Restaurants extrahieren möchte, betätigt er den Button *Start Scraping*. Wurden die Informationen bereits in der Datenbank abgespeichert, wird die folgende Information ausgegeben: *Dieses Restaurant ist bereits in der Datenbank hinterlegt!* Andernsfalls werden die Daten automatisch in der Datenbank abgespeichert.
Dabei wird ein Restaurant in die Tabelle *Restaurants* hinzugefügt:


```
"RESTAURANTS": {
    "1": {
        "ANZAHL_BEWERTUNGEN": "71",
        "KUECHE": "deutsch, mitteleurop\u00e4isch",
        "PLZ_ORT": "81479 M\u00fcnchen, Bayern, ",
        "POPULARITAET": "Nr.\u00a0654 von 3.083 Restaurants in M\u00fcnchen",
        "PREIS_LEVEL": "\u20ac\u20ac - \u20ac\u20ac\u20ac",
        "PUNKTESKALA": "4",
        "RESTAURANTNAME": "Sollner Hof",
        "R_ID": 1,
        "STRASSE": "Herterichstra\u00dfe 65",
        "TELEFONNUMMER": "+49 89 74982910"
    }
},
```


Abbildung 5.4: Tabelle 'Restaurants' am Beispiel Sollner Hof

Zusätzlich werden insgesamt 56 Bewertungen und User sowie drei Bewertungsbilder in der Datenbank abgespeichert:


```
"56": {
    "BEWERTUNG": "Nette Atmosph\\u00e4re und guter Service. Das Essen war auch gut und es hat uns
    "RATING": "4",
    "REV_ID": 56,
    "R_ID": 1,
    "TITEL": "Gem\\u00fctliches kleines Restaurant",
    "U_ID": 56
}
```


Abbildung 5.5: Tabelle 'Reviews' am Beispiel Sollner Hof

```
"56": {
    "ANZAHL_LIKES": "19",
    "ANZAHL_REVIEWS": "47",
    "BENUTZERNAME": "tilfyre",
    "REV_ID": 56,
    "U_ID": 56
},
```


Abbildung 5.6: Tabelle 'Users' am Beispiel Sollner Hof


```
"PICTURES": {
    "1": {
        "QUELLE": "https://media-cdn.tripadvisor.com/media/photo-f/11/6b/45/61/img-20171130-wa0037-largejpg.jpg",
        "REV_ID": 1
    }
},
```


Abbildung 5.7: Tabelle 'Pictures' am Beispiel Sollner Hof

6 Erfüllung der Anforderungen

Funktionale Anforderungen	erfüllt durch	Nichtfunktionale Anforderungen	erfüllt durch
zu erfassende Daten	WebScraper (Beautiful Soup)	**Usability - Graphische Benuzeroberfläche (GUI)**	Tkinter
Datenbank anforderungen	TinyDB		
Ausführbarkeit ders Anwenders	Python3 CLI		

Abbildung 6.1: Tabelle der erfüllten Anforderungen

6.1 Funktionale Anforderungen

Zu erfassende Daten alle angeforderten Daten der Restaurants sowie deren Bewertungen werden, wie im Detail in Kapitel 4 anschaulich erklärt, mit der Python Library „Beautiful Soup“ erfasst und zur weiteren Verarbeitung verfügbar gemacht.

Datenbank Anforderungen Wie in Kapitel 4.4 beschrieben, haben sich die Ersteller der Arbeit für den mit Python gut vereinbaren und leichtgewichtigen Datenbank-Typen „TinyDB“ entschieden. Eben da wurde geschildert wie und in welcher Form die extrahierten Daten in einer JSON-Datei presistiert werden und die angestrebten Datenbank-Qualitätsmerkmale bewerkstelligt wurden.

Ausführbarkeit des Anwenders Der Anwender des Programms kann, gegeben, dass die Infrastruktur für die Laufbarkeit von Python-Code vorinstalliert ist, das Programm ausführen, über die Kommandozeile eines Terminals in Unix- oder Mac-Systemen oder die Dos-Eingabeaufforderung eines Windowssystems.

Eingabe einer URL als Input-Parameter Die Eingabe des Input-Parameters wurde über die in Kapitel 4.5 behandelte GUI und dessen entsprechendes Texteingabefeld bewerkstelligt. So ist es dem User möglich dem Programm den Input-Parameter zu übergeben.

6.2 Nicht-funktionale Anforderungen

In Kapitel 4.5 wurde die Implementierung und Konzeption der graphischen Benutzeroberfläche dokumentiert. Über sie ist es möglich die Funktionalitäten des erstellten Programms zu steuern. Der Anwender der Software kann in das dafür vorgesehene Textfeld die URL des zu extrahierenden Restaurants eingeben. Über den Button „Preview“ wird zunächst die Validierung der URL vorgenommen und anschließend werden in der GUI die drei Grundinformationen Name, Ort und Postleitzahl und Durchschnittsbewertung angezeigt. Bei Klick auf „Start Scraping“ wird zunächst geprüft ob das zu extrahierende Restaurant schon in der Datenbank vorhanden ist und beginnt, falls dies nicht der Fall ist, Restaurantdaten und die seiner Reviews zu extrahieren und in der Datenbank zu persistieren.

7 Fazit und Ausblick

Zusammenfassend lässt sich sagen, dass die vom Projektteam aufgestellten Anforderungen an die Implementierung eines Webscrapers für die Webseite Tripadvisor erfüllt werden konnten. Mit diesem hat der Anwender nun die Möglichkeit, alle gespeicherten Daten zu einem Restaurant inklusive dessen Bewertungen in kürzester Zeit in einer eigenen Datenbank abzulegen und diese beliebig weiter zu verwenden. Mit dem Einsatz der Programmiersprache Python und verschiedenster Bibliotheken konnte sowohl das Ziehen als auch das Speichern der Daten in einer TinyDB Datenbank sehr effizient umgesetzt werden.
Die Anforderung an den Eingabeparameter für das zu scrapende Restaurant wurde nach eingehender Analyse im Laufe des Projektes überarbeitet. Ursprünglich sollte der Nutzer alleine über den Namen und dessen Ort das zu scrapende Restaurant auswählen können. Allerdings enthält die individuelle URL eines Restaurants eine Restaurant-ID die keiner einfachen Stringerweiterung (Restaurant Name und Ort oder einer anderen von außen erkennbaren Logik) folgt. Diese URL ist allerdings nötig als Input für die Extraktion des zu scrapenden Sourcecodes einer Restaurantseite. Nach einiger Recherche zeigte sich, dass nur über die Tripadvisor-eigene API Abfrage diese ID in der erarbeiteten Software intern generiert werden könnte. Keys zu dieser vergibt Tripadvisor nur unter starken Auflagen, die das Projektteam nicht erfüllt. Daher hat man sich in Absprache mit dem Auftraggeber darauf geeinigt, dass es hinsichtlich der Benutzerfreundlichkeit ausreicht, die komplette URL des zu scrapenden Restaurants als Input für das Programm bereitstellen zu müssen, anstatt lediglch Name und Ort.
Ein mögliche Erweiterung des Programms wäre außerdem die Ausweitung auf die gesamte Tripadvisor Plattform. Hier könnten neben Restaurant Daten auch Informationen zu Hotels, Ferienwohnungen, Flügen oder anderen Aktivitäten geparsed und in einer Datenbank gesichert werden. Die Implementierung müsste hier lediglich im Bereich der zu suchenden Attributnamen bzw. Klassen und in der Datenbankstruktur angepasst werden.

Abbildungsverzeichnis

Programmcodeverzeichnis

Literaturverzeichnis

[1] Atom Feed
Ryte Wiki: Atom Feed,
https://de.ryte.com/wiki/Atom_Feed, aufgerufen am 2. Juli 2018

[2] Deep Web
Advidera: Deep Web,
https://www.advidera.com/glossar/deep-web, aufgerufen am 2. Juli 2018

[3] Kodierung in UTF-8
UTF-8, a transformation format of ISO 10646,
https://tools.ietf.org/html/rfc3629, aufgerufen am 2. Juli 2018

[4] TinyDB
Project Description TinyDB,
https://pypi.org/project/tinydb/, aufgerufen am 2. Juli 2018

[5] SQLite
About SQLite,
https://www.sqlite.org/about.html, aufgerufen am 2. Juli 2018

[6] Definition von Datenarten zur konsistenten Kommunikation im Unternehmen
Piro A., Gebauer M. Hrsg: „Vieweg und Teubner in Daten- und Informationsqualität.“, S.143 - 156

[7] Practical Web Scraping for Data Science: best practices and examples with Python
vanden Broucke S., Baesens B. Hrsg: ØApress Media LLC: Welmoed Spahr.“

[8] Jan Seipel
Webscraping als Methode der Informationsbeschaffung,
http://www.vfm-online.de/weblog/wp-content/uploads/2014/01/info7-2016-2_S15-17.pdf, aufgerufen am 2. Juli 2018

[9] ScrapeHero
What is web scraping – Part 1 – Beginner's guide, https://www.scrapehero.com/a-beginners-guide-to-web-scraping-part-1-the-basics/, aufgerufen am 10. Juni 2018

[10] TripAdvisor
Restaurant Sollner Hof, https://www.tripadvisor.de/Restaurant_Review-g187309-d983126-Reviews-Sollner_Hof-Munich_Upper_Bavaria_Bavaria.html, aufgerufen am 9. Juni 2018

[11] Umsatz von Tripadvisor weltweit,
https://de.statista.com/statistik/daten/studie/543148/umfrage/umsatz-von-tripadvisor-weltweit-nach-quartalen/, aufgerufen am 2. Juni 2018

[12] Tripadvisor Mediacenter,
https://tripadvisor.mediaroom.com/DE-about-us, aufgerufen am 2. Juni 2018

[13] Our Logos: TripAdvisor Media Group Logos,
https://tripadvisor.mediaroom.com/our-brand, aufgerufen am 2. Juni 2018

[14] Süddeutsche Zeitung,
Bei Kundenbewertungen wird gelogen, was das Zeug hält
https://bit.ly/2w1oZqQ, aufgerufen am 10. Juni 2018

[15] Web scraping with Python: collecting data from the modern web
Mitchel, R. Hrsg: Ø'Reilly Media, Inc.

[16] Beautiful Soup Documentation,
http://www.crummy.com/software/BeautifulSoup/bs4/doc/ aufgerufen am 2. Juli 2018

[17] Screenshot Review Container,
https://bit.ly/2JC9HAo, aufgerufen am 2. Juli 2018

[18] Python Pipenv & Virtual Environments,
https://bit.ly/2sPKwnw, aufgerufen am 2. Juli 2018

[19] ScrapeHero,

Beginners guide to Web Scraping: Part 2 – Build a web scraper for Reddit using Python and BeautifulSoup

`https://bit.ly/2JHoVnT`, aufgerufen am 2. Juli 2018

[20] Tkinter,

Graphical User Interfaces with Tk

`https://docs.python.org/3/library/tk.html`, aufgerufen am aufgerufen am 28.Juni 2018

Eidesstattliche Erklärung

Hiermit erklären wir, Johannes Knippel, Anja Wolf, Johanna Sickendiek und Skanny Morandi, dass wir die vorliegende Arbeit mit dem Titel *Extraktion unstrukturierter Daten und Integration in eine Datenbank am Beispiel TripAdvisor* selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet haben.

München, den 2. Juli 2018

Unterschrift

Unterschrift

Unterschrift

Unterschrift