

WEBSCRAPER

IPA 2023

MONICA DE BLASIIS

1 Umfeld und Ablauf

1.1 Aufgabenstellung

1.1.1 Erläuterungen

1.2 Projektorganisation

1.3 Vorkenntnisse der Praktikantin

1.4 Arbeitsjournal

2 Projektbeschreibung

2.1 Was ist das Ziel des Projekts?

2.2 Was ist Web Scraping?

2.3 Ist Web Scraping legal?

2.4 Herausforderungen des Web Scraping

2.5 Eine Alternative zum Web Scraping: APIs

2.6 Wie funktionieren Web Scraper?

2.7 Unterschied zwischen Web Scraping und Web Crawling

2.8 Web-Scraping-Technologien

2.9 Python und seine Bibliotheken

2.10 Realisierung des Projekts

2.11 Testfall-Protokolle

2.12 Fazit

3. Eine effektivere Lösung - Octoparse

3.1. Was ist Octoparse?

3.2 Verwendung von Octoparse

3.3 Weitere interessante Funktionen

4. Glossar

1 Umfeld und Ablauf

1.1 Aufgabenstellung

Das Projekt Web Scraper ist entstanden im Rahmen einer IPA, einer Praktikumsarbeit. Es soll mit Hilfe von BeautifulSoup in Python entwickelt werden.

Der Web Scraper sollte eine benutzerfreundliche, sauber aussehende GUI haben. Es sollte ein Eingabefeld für den Wert geben, nachdem der Benutzer die Ziel-Webseiten durchsuchen möchte, nämlich den Namen der Stadt oder des Ortes. Dazu sollte eine Schaltfläche "Scrape" vorhanden sein, die den Scraping Prozess startet.

Wenn Ergebnisse gefunden werden, sollten diese in einem neuen Fenster in einer Tabelle angezeigt werden. Die Schaltfläche "Scrape" wird dann durch eine Reihe von Schaltflächen ersetzt, die jeweils eine andere Option zum Exportieren der Daten bieten.

Ein Label zeigt auch die Anzahl der vom Scraper gefundenen Ergebnisse an. Die folgenden Daten werden vom Scraper gesucht und extrahiert:

- ID (automatisch)
- Name des Hotels, Hostels, B&B
- Strasse und Nummer
- Ort
- Land
- URL
- Zimmerpreis
- Bild

1.1.1 Erläuterungen

- Es werden nur die URLs von offiziellen und anerkannten Buchungsseiten verwendet. Soziale Medien werden nicht berücksichtigt.
- Duplikate werden nicht gesucht und entfernt.
- Beim Preis wird ein Durchschnittspreis genommen.
- Die Anzahl der gesammelten Bilder muss mindestens 1 und höchstens 5 betragen.
- Eine Liste von URLs wird als Seed verwendet. Der Scraper soll nicht zu tief in die Suche nach Informationen einsteigen. Er sollte wirklich nur nach den bereits angegebenen Werten suchen und diese sammeln.
- Das Datenbankschema wird von mir erstellt.
- Der Web Scraper wird sich hauptsächlich darauf konzentrieren, Übernachtungsangebote (Hotels, Hostels, Motels, Bed & Breakfasts, usw.) zu finden.
- Der Web Scraper soll in 3 Sprachen verfügbar sein: Deutsch, Französisch und Englisch.
- Der Web Scraper bietet dem Benutzer die folgenden Exportoptionen: CSV-Datei, SQL-Skript und Import in eine Datenbank.
- Der Benutzer muss in der Lage sein, den Web Scraper für WordPress zu verwenden. Dies kann durch das Schreiben eines Plugins in PHP für WordPress geschehen, das den Python-Code aufruft, oder durch ein bestehendes Plugin, das durch das Lesen einer CSV-Datei eine Tabelle zur Anzeige der Daten erstellen kann.
- Es sollte ein Handbuch geschrieben werden, wie man den Scraper mit Wordpress benutzt und wie man den Code der Anwendung erweitern kann.

1.2 Projektorganisation

Das Projekt wird so weit wie möglich der Struktur der IPERKA-Methode folgen.

I - Informieren

Im ersten Teil der Dokumentation werden alle Informationen, die ich gesammelt habe, um an dem Projekt arbeiten zu können, zusammengefasst. Diese Informationen betreffen zum Beispiel eine breite Erklärung dessen, was ein Web Scraper genau ist oder tut, wie Informationen von ausgewählten Websites gesammelt werden können und so weiter.

P - Planen

Im zweiten Teil werde ich alle möglichen Lösungen erläutern, die ich für jede der Projektaufgaben gefunden habe.

E - Entscheiden

Im dritten Teil werde ich erklären, welche Option ich aus den im Planungsteil der Dokumentation beschriebenen ausgewählt haben werde, und auch dokumentieren, warum sie ausgewählt wurde und warum die anderen letztendlich nicht verwendet wurden.

R - Realisieren

Im vierten Teil beschreibe ich ausführlich Schritt für Schritt den Prozess, durch den ich die Anwendung zusammengestellt habe, begleitet von Code-Snippets.

K - Kontrollieren

Im fünften Teil stelle ich die Testprotokolle der Anwendungen und deren Ergebnisse vor. Sollten einige Elemente nicht wie beabsichtigt funktioniert haben, werde ich erklären, was genau nicht funktioniert hat und ob ich eine Lösung oder Lösungen dafür gefunden habe. Wenn eine Lösung gefunden wurde, muss ich diese auch dokumentieren. Wie im 2. und 3. Teil müssen die gefundenen Lösungen dokumentiert und die gewählte Lösung begründet werden. Der Grund, warum es hier dokumentiert wird, ist, dass es kein geplantes Feature des Projekts ist, sondern ein Nebenprodukt, wenn entweder die ursprüngliche Lösung nicht funktioniert oder eine bereits etablierte Lösung weiterentwickelt werden muss.

A - Auswerten

Im sechsten und letzten Teil des Projekts schreibe ich ein Fazit zu meinem Projekt, gehe kritisch auf die Ergebnisse des Projekts ein und drücke aus, was beim nächsten Mal besser gemacht werden könnten.

Bei der vorliegenden Dokumentation haben wir unnötige Details entfernt. Schliesslich geht es vor allem darum, dass die Arbeit weiterentwickelt wird. Der aktuelle Stand ermöglicht heute wenig Anwendungsmöglichkeiten.

Das Webscraping ist wichtig für unsere touristischen Webseiten. Auch glauben wir, dass ein Plugin für WordPress sinnvoll ist.

1.3 Vorkenntnisse der Praktikantin

- **Python:** Selbstlernkurse zu OpenHPI und Udemy. Erstellung kleiner Anwendungen mit Pygame Zero.
- **Datenbankverwaltung:** In der Schule durchgeführte Übungen zum Import und Export sowie zum Erstellen von Anwendungen, die aus der Ferne mit einer Tabelle interagieren können. Kenntnisse über den Aufbau eines SQL-Skripts und dessen Erstellung.
-

- **WordPress:** Erstellung, Migration und Verwaltung von WordPress-basierten Websites. Fehlerbehebung bei Fehlern, die durch fehlerhafte oder widersprüchliche Plugins verursacht wurden. Manuelles Bearbeiten von Plugins, genauer gesagt PHP-Dateien, um eventuellen Anforderungen gerecht zu werden. Lernen, wie man ein benutzerdefiniertes Plugin erstellt; Das Wissen wurde jedoch noch nicht vollständig in die Praxis umgesetzt.
- **PHP:** Selbstlernkurse auf Udemy. Erstellung eines kleinen Sandbox-CMS zum Üben.

1.4 Arbeitsjournal

Datum	Tätigkeiten	Bemerkungen
27.03.2023	• Beginn der Erstellung der Dokumentation (Teil I von IPERKA) • Informations- und Datenerhebung	Unter "Informations- und Datenerhebung" ist das Sammeln von Informationen zu verstehen, die als Referenz für das Projekt verwendet werden, sowie von Daten, die in das Projekt einfließen (z. B. die Verbindung(en) des/der Seed(s)).
28.03.2023	• Fortführung der Dokumentation (I) • Erstellung einer Vorlage für ein Arbeitsjournal • Vorbereitung des Arbeitsbereichs (Installation von IDE und benötigten Bibliotheken/Paketen) • Überprüfung der Installationen	Ich bin darauf aufmerksam geworden, dass BeautifulSoup möglicherweise nicht die beste Lösung für diese Art von Projekt ist. Ich fand eine andere mögliche Lösung in einer anderen Bibliothek, Selenium.
29.03.2023	• Fortführung der Dokumentation (P) • Fortführung der Dokumentation (E) • Erstellung der GUI-Skizze und des Flussdiagramms • Beginn mit der Entwicklung des Scrapers	Überprüft, dass BeautifulSoup tatsächlich nicht die richtige Lösung zum Scrapen dynamischer Websites ist. Wechsel der Bibliothek zu Selenium.
30.03.2023	• Fortführung der Entwicklung von Scraper	Selenium erweist sich als besser als BeautifulSoup, allerdings wird Selenium von einigen Websites blockiert, wenn es im Headless-Modus ausgeführt wird. Dies ist auf den Bot-Schutz der Website zurückzuführen.
31.03.2023	• Fortführung der Entwicklung von Scraper • Test Scraper-Anwendung	Ich habe versucht, eine Reihe von Lösungen zu finden und zu implementieren, um zu verhindern, dass meine eigene Anwendung als Bot erkannt wird, aber alle schlugen fehl. Infolgedessen mussten einige der ursprünglich geplanten Seeds entfernt werden. Auch wenn dies selten vorkommt, kann es vorkommen, dass die Seed-Seite geladen wird und ihre Elemente ein wenig anders als gewöhnlich angezeigt werden, was dazu

		führt, dass der Scraper stecken bleibt. Ich habe versucht, das Problem zu beheben, konnte aber die Ursache dafür nicht finden. Ich vermute jedoch, dass es mit einem Bot-Schutzsystem zusammenhängen könnte.
03.03.2023	• Scraper dazu bringen, eine CSV-Datei zu erstellen	Obwohl die Funktion .strip() verwendet wurde, werden einige der Namen mit zusätzlichen Apostrophen gespeichert. Die Ursache dafür ist unbekannt, da die Konsole keinen Fehler anzeigt.
04.03.2023	• Web-Scraper SQL-Skript generieren lassen • Web Scraper mit der Datenbank verbinden und gesammelte Daten importieren	Hatte einige Probleme, bei denen die Schleife nicht richtig funktionierte und die allerletzte Zeile des Datenrahmens nicht in die im Skript enthaltenen Werte eingefügt wurde. Das Problem wurde gelöst, indem einige Bedingungen innerhalb der Schleife geändert wurden. Direkter Import fehlgeschlagen. Momentan bin ich noch auf der Fehlersuche.
05.03.2023	• Überarbeitung der Dokumentation an die IPA-Standards des Dokuments „FArbeit_2023.pdf“	Die Dokumentation war bis zu diesem Zeitpunkt nicht korrekt durchgeführt worden, also habe ich alle nicht konformen Teile neu geschrieben und hinzugefügt, was fehlte.
06.03.2023	• Hinzufügen von Sortieroptionen zur Anwendung • Behebung des Problems mit dem Datenbankimport • Suchen von Möglichkeiten, generierte CSVs in WordPress zu importieren und den Inhalt in Form einer Tabelle anzuzeigen • Überarbeitung der Dokumentation	Problem mit dem Datenbankimport behoben, Wechsel von der PyMySQL-Bibliothek zur MySQL-Connector-Python-Bibliothek erforderlich, damit das Programm ordnungsgemäß funktioniert. Nicht genug Zeit, um ein eigenes WordPress-Plugin zu entwickeln, das den Scraper enthält. Auf der Suche nach einer alternativen Lösung, um zumindest die gesammelten Daten auf der Website anzeigen zu können.
11.03.2023	• Fazit und Beobachtungen hinzugefügt • Erstellung, Zusammenstellung und Fertigstellung des Handbuchs • Glossar und Quellenverzeichnis hinzugefügt • Abstände im IPA-Bericht korrigiert	Nichts Nennenswertes zu berichten

2 Projektbeschreibung

2.1 Was ist das Ziel des Projekts?

Ziel des Projekts ist die Entwicklung eines Web Scrapers, dessen Zielseiten renommierte Buchungsseiten für verschiedene Übernachtungsmöglichkeiten sind.

Der Benutzer sollte in der Lage sein, direkt nach dem Ausführen der Anwendung die Sprache auszuwählen, in der sie angezeigt werden soll. Die Sprachen, aus denen der Benutzer wählen kann, sind Englisch, Deutsch und Französisch. Die Standardsprache ist Englisch. Englisch wurde als Option gewählt, da es sich um eine pseudo-internationale Sprache handelt. Deutsch und Französisch wurden als Option gewählt, weil sie die beiden meistgesprochenen Landessprachen der Schweiz sind.

Nachdem man den Namen einer Stadt seiner Wahl eingegeben und auf eine Schaltfläche geklickt hat, sucht der Scraper nach den besten Angeboten auf jeder der Zielseiten, die entweder aus der eingegebenen Stadt oder aus der Nähe stammen.

Aus diesen Angeboten extrahiert der Scraper dann alle relevanten Informationen, die unter dem Abschnitt Aufgabenstellung erwähnt werden.

Sobald der Scraper Informationen extrahiert hat, erhält der Benutzer die Möglichkeit, die gesammelten Daten entweder als CSV-Datei herunterzuladen, als SQL-Skript herunterzuladen oder die Daten direkt in eine Datenbank seiner Wahl hochzuladen.

Die von diesem Web Scraper gesammelten Daten sollen auf einer WordPress-basierten Seite in schöner und lesbarer Form dargestellt werden können. Dies kann entweder durch Einbetten der Anwendung in eine Seite oder durch Verwendung eines Plugins erfolgen, das die generierte CSV-Datei in eine Tabelle umwandeln kann.

Als letztes sollte eine Anleitung erstellt werden, wie man dieses Programm mit WordPress verwendet und wie man den Code weiter ausbaut.

2.2 Was ist Web Scraping?¹

Unter Web Scraping versteht man das Sammeln von Daten von einer oder mehreren Websites. Diese Informationen werden dann in ein Format exportiert, das für den Benutzer nützlicher ist, wie eine Excel-Tabelle oder eine API.

Obwohl Web Scraping theoretisch manuell durchgeführt werden kann, greifen viele in der Praxis lieber auf automatisierte Hilfsmittel für diesen Prozess zurück, da diese nicht nur kostengünstiger sind, sondern die Arbeit auch viel schneller erledigen.

Schließlich ist Web Scraping alles andere als eine einfache Aufgabe. Websites gibt es in vielen Formen, sie sind auf verschiedene Art und Weise strukturiert und ihre Tags sind alle unterschiedlich benannt, auch je nachdem, in welcher Sprache die Website angezeigt wird. Aus diesem Grund unterscheiden sich Web Scraper in ihrer Funktionalität und ihren Merkmalen. Es wäre nicht falsch zu sagen, dass einige Web Scraper speziell dafür entwickelt wurden, Informationen von einer bestimmten Website zu extrahieren.

Ein weiteres Element, das einem Web Scraper in die Quere kommen kann, ist, dass einige Websites durch Captchas geschützt sind. Es gibt jedoch Möglichkeiten, diese zu umgehen.

¹ What is Web Scraping and What is it used for?, Martin Perez,
<https://www.parsehub.com/blog/what-is-web-scraping/>

2.3 Ist Web Scraping legal?²

Die kurze Antwort lautet: Ja. Der Vorgang des Web Scraping selbst ist technisch gesehen nicht illegal. Es müssen jedoch einige Regeln befolgt werden, damit es als solches betrachtet werden kann. Web Scraping kann illegal werden, wenn zum Beispiel nicht öffentlich zugängliche Daten extrahiert werden.

Bei der Diskussion über die Rechtmäßigkeit von Web Scraping muss eine klare Unterscheidung hinsichtlich der Art der Daten getroffen werden. Es muss sich um Daten handeln, auf die jeder, der eine Internetverbindung hat, zugreifen kann.

Es gibt einige einfache Möglichkeiten, um festzustellen, ob Daten im Internet als öffentlich zugänglich gelten:

- Der Benutzer, der die Daten veröffentlicht hat, hat beschlossen, sie öffentlich zugänglich zu machen
- Ein Benutzer muss kein Konto erstellen oder sich anmelden, um auf die Daten zuzugreifen
- Die robots.txt der Website blockiert keine Web Scraper oder Spider

Andererseits gibt es Fälle des Sammelns und Auslesens privater Daten, die sich in einem völlig anderen Bereich der Legalität bewegen. Beachten Sie dazu die Präsentation, die dazu Frau De Blasiis verfasst hat.

2.4 Herausforderungen des Web Scraping³

Das Web ist aus vielen Quellen organisch gewachsen. Es vereint viele verschiedene Technologien, Stile und Persönlichkeiten und wächst bis heute weiter. Aus diesem Grund kann man beim Scrapen des Webs auf einige Herausforderungen stoßen:

- **Vielfalt:** Jede Website ist anders. Während man auf allgemeine Strukturen stößt, die sich wiederholen, ist jede Website einzigartig und erfordert eine persönliche Behandlung, wenn man die relevanten Informationen extrahieren möchte.
- **Langlebigkeit:** Websites ändern sich ständig. Zum Beispiel hat man einen neuen Web Scraper, der automatisch aus seiner interessanten Ressource herauspickt, was er will. Das erste Mal, wenn man sein Skript ausführt, funktioniert es einwandfrei. Aber wenn man das gleiche Skript nur kurze Zeit später ausführt, stößt man auf einen langen Stapel von Tracebacks.

Instabile Skripte sind ein realistisches Szenario, da sich viele Websites in aktiver Entwicklung befinden. Sobald sich die Struktur der Website geändert hat, kann der zuvor einwandfrei funktionierende Scraper möglicherweise nicht mehr richtig in der Sitemap navigieren oder die relevanten Informationen finden. Die gute Nachricht ist, dass viele Änderungen an Websites klein und inkrementell sind, sodass man seinen Scraper wahrscheinlich nur mit minimalen Anpassungen aktualisieren kann.

Es ist jedoch wichtig zu bedenken, dass die Scraper, die man baut, wahrscheinlich ständig gewartet werden müssen, da das Internet dynamisch ist. Man kann Continuous Integration einrichten, um regelmäßig Scraping-Tests durchzuführen, um sicherzustellen, dass das Hauptskript nicht ohne sein Wissen kaputt geht.

² Is Web Scraping Legal?: Can You Get in Trouble for Web Scraping Publicly Available Data?, Martin Perez, <https://www.parsehub.com/blog/web-scraping-legal/>

³ Beautiful Soup: Build a Web Scraper With Python, Martin Breuss, <https://realpython.com/beautiful-soup-web-scraper-python/>

2.5 Eine Alternative zum Web Scraping: APIs⁴

Einige Website-Anbieter bieten Application Programming Interfaces (APIs) an, die einen vordefinierten Zugriff auf ihre Daten ermöglichen. Mit APIs kann das Parsen von HTML vermieden werden. Stattdessen kann man über Formate wie JSON und XML direkt auf die Daten zugreifen. HTML ist in erster Linie eine Möglichkeit, Benutzern Inhalte visuell zu präsentieren.

Wenn eine API verwendet wird, ist der Prozess im Allgemeinen stabiler als das Sammeln der Daten durch Web Scraping. Das liegt daran, dass Entwickler APIs erstellen, die von Programmen und nicht von menschlichen Augen verwendet werden.

Die Front-End-Präsentation einer Website kann sich häufig ändern, aber eine solche Änderung im Design der Website hat keinen Einfluss auf die API-Struktur. Die Struktur einer API ist normalerweise dauerhafter, was bedeutet, dass sie eine zuverlässige Quelle für die Daten der Website ist.

APIs können sich jedoch ebenfalls ändern. Die Herausforderungen sowohl der Vielfalt als auch der Dauerhaftigkeit gelten für APIs genauso wie für Websites. Darüber hinaus ist es viel schwieriger, die Struktur einer API selbst zu überprüfen, wenn die bereitgestellte Dokumentation von mangelnder Qualität ist.

2.6 Wie funktionieren Web Scraper?⁵

Automatisierte Web Scraper arbeiten auf eine recht einfache, aber auch komplexe Weise. Schließlich sind Webseiten für Menschen gemacht, die sie verstehen sollen, und nicht für Maschinen.

Als erstes erhält der Web Scraper eine oder mehrere URLs, die er vor dem Scraper laden muss. Der Scraper lädt dann den gesamten HTML-Code der gewünschten Seite. Fortgeschrittene Scraper rendern die gesamte Webseite, inklusive CSS- und Javascript-Elemente.

Dann extrahiert der Scraper entweder alle Daten auf der Seite oder bestimmte Daten, die der Benutzer vor der Ausführung des Projekts ausgewählt hat.

Im Idealfall durchläuft der Benutzer den Prozess der Auswahl der spezifischen Daten, die er von der Seite haben möchte. Zum Beispiel könnte man die Produktseite einer Online-Shopping-Website nach Preisen und Modellen durchsuchen, ist aber nicht unbedingt an Produktbewertungen interessiert.

Zum Abschluss gibt der Web Scraper alle gesammelten Daten in einem Format aus, das für den Benutzer nützlicher ist. Die meisten Web Scraper geben die Daten in eine CSV- oder Excel-Tabelle aus, während fortgeschrittenere Scraper andere Formate wie JSON unterstützen, die für eine API verwendet werden können.

2.7 Unterschied zwischen Web Scraping und Web Crawling⁶

Ein anderer Begriff, den man oft hört, wenn man über Web Scraping spricht, ist Web Crawling. Manche Leute verwenden diese Begriffe zwar synonym, aber sie sind keineswegs dasselbe.

⁴ Beautiful Soup: Build a Web Scraper With Python, Martin Breuss,
<https://realpython.com/beautiful-soup-web-scraper-python/>

⁵ What is Web Scraping and What is it used for?, Martin Perez,
<https://www.parsehub.com/blog/what-is-web-scraping/>

⁶ Web crawling vs web scraping, Colm Kenny,
<https://www.zyte.com/learn/difference-between-web-scraping-and-web-crawling/#:~:text=The%20short%20answer,to%20combine%20crawling%20and%20scraping>

Web Scraping ist eine Technik, die verwendet wird, um eine große Menge an Daten von Websites zu extrahieren und sie dann in Form von XML, Excel oder SQL auf dem lokalen Rechner zu speichern. Auf der Grundlage der gegebenen Anforderungen können sie die Daten von jeder Website in einem Bruchteil der Zeit extrahieren. Diese Automatisierung von Aufgaben ist sehr hilfreich für die Entwicklung von Daten für maschinelles Lernen und andere Zwecke. Sie arbeiten normalerweise in vier Schritten:

- Senden der Anfrage an die Zielseite
- Abrufen der Antwort von der Zielseite
- Parsen und Extrahieren der Antwort
- Herunterladen der Daten

Beim Web Crawling hingegen werden eine oder mehrere Websites besucht und deren Seiten gelesen, um Einträge für den Suchmaschinenindex zu erstellen. Die Werkzeuge, die für das Web Crawling verwendet werden, sind als Web Crawler oder Spider bekannt. Eine Reihe von Webseiten wird analysiert, und die Links zu den Seiten werden verfolgt, um noch mehr Links zu finden, so dass eine Tiefensuche zur Extraktion von Informationen durchgeführt wird. Bekannte Suchmaschinen wie Google, Yahoo und Bing führen Web Crawling durch und verwenden diese Informationen zur Indizierung von Webseiten.

2.8 Web-Scraping-Technologien⁷

Die Wahrheit ist: Fast jede Programmiersprache kann zum Scrapen von Websites verwendet werden. Aber einige von ihnen haben mehr Tools, Bibliotheken oder Frameworks als andere. Die Wahl der besten Codierung-Sprache für Web Scraping sollte von der Flexibilität der Sprache, der Einfachheit der Codierung, der operativen Fähigkeit, Datenbanken zu füttern, der Scraping-Effektivität, der Skalierbarkeit und der Vermeidung von Blockierungs- und Erkennungsmechanismen abhängen.

Python

Python ist einfach zu schreiben, zu lesen und zu verstehen. Im Gegensatz zu anderen Programmiersprachen wie Java oder C++ hat Python eine relativ niedrige Einstiegshürde und eine hohe Lerngeschwindigkeit. Außerdem wird durch die Interpretation der Sprache (die Ausführung des Programmcodes erfolgt zeilenweise ohne vorherige Kompilierung) die Geschwindigkeit des Programms deutlich erhöht.

Auch Python entwickelt sich schnell und intensiv. Mit jeder Version verbessert sich die Leistung der Sprache und damit auch die Syntax. Bei Sprachen wie C++ oder Java ist die Änderungsrate merklich langsamer – sie werden von einer speziellen Kommission genehmigt, die sich alle paar Jahre trifft.

Python hat viele Bibliotheken, Frameworks und Tools, um mit Web Scraping zu arbeiten: Requests-Bibliothek (ist eine eingebaute Bibliothek), BeautifulSoup, Selenium-Bibliothek, Scrapy-Framework, Puppeteer, URLLib, lxml und so weiter. Dank einer Vielzahl von Tools ermöglicht Python die Durchführung aller erforderlichen Aufgaben: sei es das Parsen dynamischer Daten, das Einrichten eines Proxys oder das Arbeiten mit einer einfachen HTTP-Anfrage.

Ruby

Aufgrund seiner Einfachheit und Leistung ist Ruby ideal zum Erstellen von Scraper-Bots. Im Gegensatz zu anderen Programmiersprachen bietet Ruby die Möglichkeit, Bots zu erstellen, die mithilfe von CSS-Selektoren nach HTML-Dokumenten suchen können.

⁷ The Best Programming Languages for Web Scraping, Valentina Skakun, <https://scrape-it.cloud/blog/web-scraping-languages>

Ruby kombiniert mehrere Programmiersprachen - Perl, Smalltalk, Eiffel, Ada und Lip. Ruby ist eine der einfachsten Web-Scraping-Sprachen, es erfordert weniger Schreiben für eine solche Sprache und es treten keine Anzeichen einer Code-Wiederholung auf. Ruby wird von einer Community von Benutzern unterstützt.

Es gibt auch Verpackungsmanager wie RubyGems, HTTParty oder NokoGiri, die beim Einrichten von Web Scrapern helfen können.

Node.js

Node.JS basiert auf Javascript und ist eine gute Codierung Option für das Scraping von Javascript-Seiten. Node.JS ist für Streaming, Socket-basierte Implementierung und API geeignet und wird uneingeschränkt empfohlen.

Viele Leute verwenden Node.JS für mehrere Instanzen für dasselbe Scraping-Projekt, da Node.JS nur einen Kern der CPU benötigt. Node.JS verfügt über eine Reihe von Bibliotheken, mit denen Daten gescraped werden können: Puppeteer, Cheerio, Node-Fetch, JSDOM usw.

Golang

In letzter Zeit ist die Golang-Programmiersprache sehr beliebt geworden, mit der sich leicht ein Golang-Web-Scraper erstellen lässt. Die Wahl eines flexiblen und leicht skalierbaren Scrapers wie dem Golang Web Scraper kann die Datenerfassung kurz- und langfristig vereinfachen.

Golang ist die beste Sprache für diejenigen, die schnell mit dem Scraping beginnen wollen - es wird einen einfachen Code geben, der ausreicht, um HTML zu parsen. Um Web Scraper auf Golang zu machen, kann man Bibliotheken von Drittanbietern wie Goquery oder Colly verwenden.

Perl

Perl eignet sich hervorragend zum Analysieren von Text und bietet eine gute Unterstützung für reguläre Ausdrücke, sodass es sich perfekt für das Web-Scraping eignet. Es hat auch eine starke Online-Community (CPAN), die viele Bibliotheken entwickelt hat, die für das Parsen nützlich sind.

Das beliebteste Web-Scraping-Modul von Perl ist WWW::Mechanize, was großartig ist, wenn man nicht nur die Zielseite erhält, sondern auch über Links oder Formulare dorthin navigieren möchte, z. B. für die Anmeldung. Natürlich hat Perl andere, weniger populäre Bibliotheken für Web Scraping, zum Beispiel HTML::TreeBuilder, Mojo oder Jada.

Perl kann auch sehr prägnant geschrieben werden, was einen schnellen Einstieg ermöglicht.

PHP

PHP ist eine Programmiersprache, die verwendet wird, um mit Webinhalten zu arbeiten. Um mit Data Scraping zu arbeiten, hat PHP mehrere Bibliotheken: libcurl, Nokogiri, Zend_DOM_Query, htmlSQL, Fluent DOM und Ganon.

PHP ist auch sehr kompatibel mit HTML und unterstützt reguläre Ausdrücke, durch die der Parser Informationen verarbeitet.

Aufgrund der Tatsache, dass PHP die Implementierung von Skripten erlaubt, funktionieren die meisten der darauf geschriebenen Parser auf ähnliche Weise. Der Ausführung-Algorithmus ist der folgende:

1. Eine Anfrage per URL wird erstellt
2. Eine Antwort vom Server als HTML wird empfangen
3. Die empfangenen Daten werden analysiert
4. Die benötigten Elemente werden extrahiert
5. Die Ergebnisse werden gebildet und angezeigt

Die Ergebnisse können in Dateien und Datenbanken geschrieben und direkt auf dem Gerätedisplay angezeigt werden. Im Allgemeinen ist es eine nicht zu komplexe, aber sehr mächtige Sprache.

C#

C# ist eine moderne, einfache, objektorientierte Programmiersprache auf hoher Ebene, die in Common Runtime Language (CRL) kompiliert und in ASP.NET JIT-interpretiert werden kann. Neben Web Scraping wird C# hauptsächlich für die Anwendungs- und Spieleentwicklung verwendet.

Im Fall von C#-Parsing macht es diese Sprache viel einfacher, die gesammelten Daten mit APIs, externen Schnittstellen und Datenbanken zu verknüpfen. Es ermöglicht Ihnen auch, Daten von mehreren Websites zu sammeln und unterstützt API-Scraping und Web-Scraping.

C und C++

Die Verwendung von C und C++ ist eine gute Wahl, wenn man einen leistungsfähigen Parser mit dynamischer Codierung schreiben muss. Es ermöglicht das Schreiben einer eigenen HTML-Parsing-Bibliothek gemäß spezifischen Anforderungen und Aufgaben.

C++ ermöglicht die Parallelisierung beliebiger Parser ohne Aufwand. Der Hauptnachteil dieser Programmiersprachen besteht jedoch darin, dass das Einrichten von Parsern mit ihnen ressourcenintensiv sein kann.

Java

Im Vergleich zu anderen Programmiersprachen ist Java besser netzwerkfähig und flexibler in der Skalierbarkeit. Dank der vielen Bibliotheken zum Parsen von XML und HTML ist Java zu einem bequemen Werkzeug zum Erstellen eines Web Scrapers geworden. Es gibt drei am häufigsten verwendete Bibliotheken und Frameworks für Web Scraping mit Java – JSoup, Jaunt und HTMLUnit.

Ab Java 9 wurde es möglich, Skripte zu erstellen. Man kann auch jede der über 20 JVM-Sprachen für Web-Scraping verwenden. Diese Sprachen ermöglichen es, jede der Java-Bibliotheken zu verwenden und können entweder als Skriptsprache oder als kompilierter Java-Bytecode verwendet werden. Daher ist es möglich, Javascript-Skripte mit Java-Bibliotheken zu schreiben.

Rust

Rust ist eine statisch typisierte Programmiersprache, die auf Leistung und Sicherheit ausgelegt ist, insbesondere auf sichere Parallelität und Speicherverwaltung.

Die Verwendung von Rust ist eine gute Idee, um einfache Dinge zu analysieren. Außerdem hat Rust gute Parser-Generator-Bibliotheken. Wenn das Scraping IO-gebunden ist, wird Rust übermäßig sein. Tatsächlich erfordert es ein wenig Sorgfalt, um mit seiner asynchronen Laufzeit die gleiche IO-Leistung wie Node.JS zu erreichen.

Die am häufigsten verwendete Bibliothek für Web Scraping in Rust ist select.rs.

2.9 Python und seine Bibliotheken

Aus den bereits im vorigen Absatz genannten Gründen, sowie bereits gesammelter Erfahrung mit der Programmiersprache, habe ich mich für Python für dieses Projekt entschieden. Unten sind die relevanten Bibliotheken, die ich darin verwenden möchte.

BeautifulSoup⁸

BeautifulSoup ist eine Python-Bibliothek, die es einfach macht, Informationen von Webseiten zu sammeln. Es sitzt auf einem HTML- oder XML-Parser und bietet pythonische Idiome zum Iterieren, Suchen und Modifizieren des Parse-Baums.

Die BeautifulSoup-Bibliothek hilft beim Isolieren von Titeln und Links von Webseiten. Es kann den gesamten Text aus HTML-Tags extrahieren und den HTML-Code in dem Dokument ändern, mit dem wir arbeiten. Einige Hauptmerkmale, die BeautifulSoup einzigartig machen, sind:

- BeautifulSoup bietet einige einfache Methoden und pythonische Redewendungen zum Navigieren, Suchen und Ändern eines Analysebaums
- BeautifulSoup wandelt eingehende Dokumente automatisch in Unicode und ausgehende Dokumente in UTF-8 um
- BeautifulSoup setzt auf beliebten Python-Parsern wie lxml und html5lib auf, was es uns ermöglicht, verschiedene Parsing-Strategien auszuprobieren oder Geschwindigkeit gegen Flexibilität eintauschen

BeautifulSoup wird in diesem Projekt verwendet, um den HTML-Code von statischen Seiten zu analysieren und Daten daraus zu extrahieren.

Selenium⁹

In einer agilen Umgebung legen Entwickler Wert darauf, Änderungen schnell voranzutreiben. Bei jeder Änderung, die eine Anpassung des Frontends erfordert, müssen entsprechende Cross-Browser-Tests durchgeführt werden. Während sich ein kleines Projekt für manuelles Testen entscheiden kann, sprechen immer mehr Browser für automatisiertes Testen.

Mit dem ständig wachsenden Umfang von Webanwendungen, sowohl in Bezug auf Technologie als auch auf Funktionalität, sind die Erwartungen der Benutzer um ein Vielfaches gestiegen. Alle paar Wochen werden den Webanwendungen neue Funktionen hinzugefügt, um die Nutzerbindung zu erhöhen. Um diese Funktionen zu testen und sicherzustellen, dass die Benutzeroberfläche gut funktioniert, sind automatisierte Tests erforderlich. Für Tester auf der ganzen Welt ist Selenium die erste Wahl für die Durchführung automatisierter Tests.

Selenium ist ein Open-Source-Automatisierungstool, das je nach zu testender Anwendung eine Reihe von Skriptsprachen wie Python, C#, Java, Perl, Ruby, JavaScript usw. unterstützt. Man kann das Skript entsprechend wählen.

Selenium ermöglicht es, Tests zu definieren und die Ergebnisse dieser Tests automatisch in einem vorher festgelegten Browser zu erkennen. Eine Reihe von Selenium-Funktionen ermöglicht es, schrittweise Interaktionen mit einer Webseite zu erstellen und die Reaktion eines Browsers auf verschiedene Änderungen zu bewerten. Man kann dann entscheiden, ob die Reaktion des Browsers mit dem übereinstimmt, was man erwartet.

Selenium wird in diesem Projekt verwendet, um mit der Website zu interagieren. Selenium navigiert zur Ziel-Website und gibt die vom Benutzer gewünschte Stadt in die Suchleiste ein, um dann alle relevanten Ergebnisse zu erhalten. Da Selenium dynamische Seiten lesen kann, ist es nicht nur in der Lage, die

⁸ What is BeautifulSoup?, Educative Answers Team, <https://www.educative.io/answers/what-is-beautiful-soup>

⁹ Selenium with Python Tutorial: Getting started with Test Automation, Shaumik Daityari und Pradeep Krishnakumar, <https://www.browserstack.com/guide/python-selenium-to-run-web-automation-test#:~:text=Selenium%20is%20an%20open%20source,can%20choose%20the%20script%20accordingly>

relevanten Daten aus den Ergebnissen fehlerfrei zu erfassen, sondern kann auch, wenn mehr als eine Ergebnisseite vorhanden ist, diese durchlaufen und Daten von ihnen erfassen.

Requests¹⁰

Die Requests-Bibliothek ist der De-facto-Standard für HTTP-Requests in Python. Es abstrahiert die Komplexität von Anfragen hinter einer schönen, einfachen API, sodass man sich auf die Interaktion mit Diensten und die Nutzung von Daten in seiner Anwendung konzentrieren kann.

Requests werden in Kombination mit BeautifulSoup verwendet. Anfragen sind für das Abrufen des HTML-Codes der Seite verantwortlich, während BeautifulSoup ihn zergliedert.

Time¹¹

Die Python Time Bibliothek bietet viele Möglichkeiten, Zeit im Code darzustellen, z. B. Objekte, Zahlen und Zeichenfolgen. Es bietet auch andere Funktionen als die Darstellung der Zeit, wie das Warten während der Codeausführung und das Messen der Effizienz des Codes.

Time wird hauptsächlich verwendet, um das Programm an bestimmten Punkten vorübergehend zu unterbrechen. Denn damit Selenium richtig mit einem Seitenelement interagieren kann, muss dieses Element vollständig geladen sein. Wenn Selenium versucht, mit einem nicht vollständig geladenen Objekt zu interagieren, führt dies zu einem Fehler und die gesamte Anwendung funktioniert nicht mehr.

Pandas¹²

Pandas ist ein Open-Source-Python-Paket, das am häufigsten für Datenwissenschaft/Datenanalyse und maschinelles Lernen verwendet wird. Es baut auf einem anderen Paket namens Numpy auf, das Unterstützung für mehrdimensionale Arrays bietet. Als eines der beliebtesten Data-Wrangling-Pakete funktioniert Pandas gut mit vielen anderen Data-Science-Modulen innerhalb des Python-Ökosystems und ist normalerweise in jeder Python-Distribution enthalten, von denen, die mit Betriebssystemen geliefert werden, bis hin zu Distributionen kommerzieller Anbieter.

Pandas macht es einfach, viele der zeitaufwändigen, sich wiederholenden Aufgaben zu erledigen, die mit der Arbeit mit Daten verbunden sind, darunter:

- Datenbereinigung
- Daten füllen
- Datennormalisierung
- Datenzusammenführungen und -verknüpfungen
- Datenvisualisierung
- statistische Analyse
- Datenprüfung
- Laden und Speichern von Daten
- Und vieles mehr

Tatsächlich kann man mit Pandas alles tun, was weltweit führende Datenwissenschaftler dazu bringt, Pandas zum besten verfügbaren Datenanalyse- und -manipulationstool zu küren.

¹⁰ Python's Requests Library (Guide), Alex Ronquillo, <https://realpython.com/python-requests/>

¹¹ A Beginner's Guide to the Python time Module, Alex Ronquillo, <https://realpython.com/python-time-module/#:~:text=The%20Python%20time%20module%20provides,the%20efficiency%20of%20your%20code.>

¹² What is Pandas in Python? Everything You Need to Know, ActiveState Team, <https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/>

In diesem Projekt werden Pandas als Mittel zum Speichern der vom Programm gesammelten Daten verwendet. Pandas verfügt auch über Funktionen, die es einfach machen, seinen Datenrahmen in eine CSV-Datei umzuwandeln.

PyMySQL¹³

PyMySQL ist eine reine Python-MySQL-Client-Bibliothek, das heißt, es ist ein Python-Paket, das eine API-Schnittstelle für den Zugriff auf relationale MySQL-Datenbanken bietet.

Dies ist die Bibliothek, die notwendig ist, damit der Python-Code auf die Datenbank zugreift, Tabellen darin erstellen und Werte einfügen kann.

Tkinter¹⁴

Tkinter ist der De-Facto-Standard in Python, um grafische Benutzeroberflächen (GUIs) zu erstellen, und ist in allen Standard-Python-Distributionen enthalten. Tatsächlich ist es das einzige Framework, das in die Python-Standardbibliothek integriert ist.

Dieses Python-Framework bietet eine Schnittstelle zum TK-Toolkit und arbeitet als dünne objektorientierte Schicht auf Tk. Das TK-Toolkit ist eine plattformübergreifende Sammlung von grafischen Steuerelementen, auch bekannt als Widgets, zum Erstellen von Anwendungsschnittstellen.

In diesem Projekt wird Tkinter verwendet, um die Schnittstelle für den Scraper zu erstellen.

Implementierung von mehreren Sprachen

Nach dem Start wird der Benutzer als erstes aufgefordert, eine Sprache auszuwählen, wobei die verfügbaren Sprachen diejenigen sind, die bereits im vorherigen Abschnitt erwähnt wurden. Die standardmäßig ausgewählte Sprache ist Englisch. Die Sprache kann durch Klicken auf das entsprechende Optionsfeld ausgewählt werden, und die Sprachänderung wird sofort sichtbar. Dies wird durch getrennte Python-Dateien ermöglicht, die ich im folgenden Format benennen werde: lp_(Initialen der Sprache). „lp“ steht für „language pack“ (Sprachpaket). Der Dateiname des englischen Sprachpakets lautet beispielsweise „lp_en“. Diese Sprachpakete enthalten alle gleichnamigen String-Variablen, denen jedoch unterschiedliche Werte zugeordnet sind, die dann von der GUI aufgerufen werden.

Datenbankschema

Das Datenbankschema für diese Anwendung ist eigentlich recht einfach, denn obwohl die Datenbank am Ende mehrere Tabellen enthält, sind sie alle unabhängig voneinander. Jede der generierten Tabellen hat insgesamt 8 Spalten, nämlich:

ID: Der einzige Wert, der für jede Zeile eindeutig ist. Es wird ein Int-Wert mit dem zugewiesenen Attribut AUTO_INCREMENT sein, sodass jeder Zeile automatisch eine ID zugewiesen wird, ohne dass sie manuell eingegeben werden muss. Damit AUTO_INCREMENT ordnungsgemäß funktioniert, wird die ID auch der Rolle des Primärschlüssels der Tabelle zugewiesen.

Name: Diese Spalte enthält die Namen aller Unterkünfte. Die Werte in dieser Spalte werden als VARCHAR mit einer Längenbegrenzung von 255 Zeichen gespeichert.

¹³ PyMySQL — Connecting Python and SQL for Data Science, Kenneth Leung,
<https://towardsdatascience.com/pymysql-connecting-python-and-sql-for-data-science-91e7582d21d7>

¹⁴ What is Tkinter used for and how to install it?, ActiveState Team,
<https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/>

Adresse: Diese Spalte enthält den Straßennamen sowie die Hausnummer, in der sich die Unterkunft befindet. Die Werte in dieser Spalte werden als VARCHAR mit einer Längenbegrenzung von 255 Zeichen gespeichert.

Stadt: Diese Spalte enthält den Namen der Stadt sowie die Postleitzahl, in der sich die Unterkunft befindet. Die Werte in dieser Spalte werden als VARCHAR mit einer Längenbegrenzung von 255 Zeichen gespeichert.

Land: Diese Spalte enthält einfach den Namen des Landes, in dem sich die Unterkunft befindet. Genau wie in den vorherigen Spalten werden die Werte in dieser Spalte als VARCHAR mit einer Längenbegrenzung von 255 Zeichen gespeichert.

Link: Diese Spalte enthält die Links, die zum Unterkunftsangebot führen. Aufgrund der sehr großen Anzahl von Zeichen, die einige Links enthalten können, werden die Werte in dieser Spalte als TEXT statt als VARCHAR gespeichert.

Preis: Diese Spalte enthält den Durchschnittspreis für 1 Zimmer für 2 Erwachsene für einen Zeitraum von 2 bis 8 Nächten in der Unterkunft. Da es sich um einen Preis handelt, wäre es möglicherweise intuitiver gewesen, die Werte in dieser Spalte als INT zu speichern, aber ich habe mich entschieden, sie weiterhin als VARCHAR mit einer Längenbegrenzung von 255 Zeichen zu speichern. Das liegt daran, dass ich das Währungssymbol als Teil des Werts behalten wollte.

Bild: Diese Spalte enthält alle Bilder, die die Unterkunftsangebote als Symbole verwenden, oder genauer gesagt ihre Quellenlinks. Während es andere Möglichkeiten gibt, Bilder in einer Datenbank zu speichern, war die Verwendung ihres Quelllinks der einfachste Weg für dieses Projekt. Genau wie in der Links-Spalte werden in dieser Spalte aufgrund der großen Anzahl von Zeichen, die ein Quell-Link haben kann, Werte als TEXT statt als VARCHAR gespeichert.

city_scraped_data	
ID (Primary Key)	INT(7) AUTO_ INCREMENT NOT NULL
name	VARCHAR(255) NOT NULL
address	VARCHAR(255) NOT NULL
city	VARCHAR(255) NOT NULL
country	VARCHAR(255) NOT NULL
link	TEXT NOT NULL
price	VARCHAR(255) NOT NULL
image	TEXT NOT NULL

Programmablauf

Nach dem Start wird dem Benutzer als erstes das Fenster mit dem Sprachauswahlmenü angezeigt. Die Sprache kann durch Anklicken des entsprechenden Optionsfeldes ausgewählt werden. Wenn Sie auf die Schaltfläche klicken, ändert sich die Sprache der Benutzeroberfläche entsprechend. Anschließend kann der Benutzer auf "Weiter" klicken, um zum zweiten Fenster zu gelangen.

Im zweiten Fenster kann der Benutzer einen Wert eingeben, genauer gesagt den Namen der Stadt, von der er eine Hotelliste erhalten möchte. Unter dem Eingabefeld befinden sich zwei Schaltflächen:

- Die erste ist einfach eine Schaltfläche, die den Benutzer zurück zum Sprachauswahlmenü bringt.
- die zweite ist diejenige, die den Scraping-Prozess tatsächlich startet

Die zweite Schaltfläche prüft zunächst die vom Benutzer eingegebenen Daten, bevor sie mit dem nächsten Schritt fortfährt. Wenn die Eingabe entweder...

- leer ist
- nur Leerzeichen enthält
- Zahlen enthält
- Sonderzeichen enthält

... dann wird die Anwendung nicht zum nächsten Fenster weitergehen. Stattdessen wird unter dem Eingabefeld eine Warnung angezeigt, die dem Benutzer erklärt, warum seine Eingabe nicht akzeptiert wurde. Sobald die Eingabe korrigiert wurde und die Schaltfläche erneut angeklickt wird, beginnt der Scraping-Prozess.

In diesem Fall öffnet die Python-Anwendung Chrome und ruft die Zielseite auf, die in diesem Projekt Booking heißt. Hier braucht der Benutzer nichts weiter zu tun, als zuzusehen und zu warten, bis der Scraper fertig ist.

Zunächst wird der Scraper die vom Benutzer angegebene Stadt eingeben. Dann wählt er einen zufälligen Aufenthaltszeitraum von 2 bis 8 Nächten im laufenden oder, wenn er fast vorbei ist, im kommenden Monat. Dann drückt er auf die Schaltfläche "Suchen" und gelangt so auf die Seite mit den Suchergebnissen.

Hier sieht es zunächst so aus, als würde der Scraper nichts tun, aber in Wirklichkeit sammelt er die relevanten Daten aus allen Ergebnissen. Von dort aus kann der Scraper die folgenden Daten sammeln:

- Der Name des Hotels oder der Unterkunftsart, um die es sich handelt
- Den Link, der zum Angebot führt
- den Durchschnittspreis
- Das als Icon verwendete Bild

Um die Adresse, die Stadt und das Land zu erhalten, wird ein weiterer Schritt im Hintergrund ausgeführt, der für den Benutzer nicht sichtbar ist, selbst wenn er auf den geöffneten Browser starrt. Der Scraper besucht nämlich auch die Seite des jeweiligen Angebots. Dort findet er die vollständigen Informationen über den Standort der Unterkunft.

Sollte mehr als eine Ergebnisseite vorhanden sein, kann der Benutzer sehen, wie der Scraper langsam durch alle Seiten navigiert und den Datenerfassung-Prozess auf jeder Seite wiederholt. Sobald alle Ergebnisseiten abgearbeitet sind, schließt sich der Browser, und die Python-Anwendung ist zum nächsten Fenster übergegangen.

Auf der dritten Seite wird der Benutzer gefragt, ob er die gesammelten Daten sortieren möchte oder nicht. Als Optionen stehen die folgenden Schaltflächen zur Verfügung:

- alphabetisch sortieren
- nach Preis sortieren (aufsteigend)
- nicht sortieren

Unabhängig davon, für welche Option sich der Nutzer entschieden hat, wird er nach Anklicken einer der Schaltflächen zum nächsten Fenster weitergeleitet.

Im vierten Fenster wird der Benutzer gefragt, wie er die gesammelten Daten speichern möchte. In Form von Schaltflächen werden ihnen die folgenden Optionen gegeben:

- CSV generieren
- SQL generieren
- beide Dateien generieren
- in eine Datenbank hochladen

Wenn die erste Option ausgewählt wird, wird eine CSV-Datei mit dem Namen (gesuchte Stadt)_scraped_data.csv im CSVs-Ordner generiert.

Wenn die zweite Option gewählt wird, wird eine SQL-Skriptdatei mit dem Namen (gesuchte Stadt)_scraped_data.sql im SQLs-Ordner generiert.

Wenn die dritte Option gewählt wird, werden die beiden oben genannten Dateien in ihren jeweiligen vordefinierten Ordnern generiert.

Wenn eine dieser drei ausgewählt wird, fährt die Python-App mit dem letzten Fenster fort, sobald sie mit dem Generieren der Datei(en) fertig ist.

Anders verhält es sich jedoch, wenn die vierte Option gewählt wird. Dort wird es ein weiteres Fenster geben, durch das der Benutzer gehen muss, bevor er das letzte erreicht. In diesem Fenster gibt es insgesamt 4 Eingabefelder für jeweils:

- Hostname
- Nutzername
- Passwort
- Name der Datenbank

Wenn alle eingegebenen Eingaben korrekt sind, lädt die Python-App alle gesammelten Daten in Form einer neu erstellten Tabelle in die ausgewählte Datenbank hoch, die nach dem Format (gesuchte Stadt)_scraped_data benannt ist, und geht dann einfach auf die letzte Seite über.

Das letzte Fenster enthält einfach zwei Elemente. Ein Etikett, das das Ende des Vorgangs ankündigt, und ein Neustart-Button, der den Benutzer beim Drücken zum ersten Fenster zurückbringt.

2.10 Realisierung des Projekts

Vorbereitung der Arbeitsumgebung

Das Wichtigste zuerst: Um dieses Projekt richtig umsetzen zu können, musste ich eine effiziente Arbeitsumgebung einrichten. Die Python-Basisinstallation verfügt zwar über eine eigene Shell, von der aus man Code schreiben und ausführen kann, aber sie ist nicht gerade die beste, wenn es um sehr langen Code oder die Zusammenarbeit mehrerer Python-Dateien geht. Deshalb habe ich beschlossen, Visual Studio Code für dieses Projekt zu verwenden.

Es ist nicht nur einfacher zu benutzen und man kann mit dem eingebauten Explorer zwischen den Dateien navigieren, sondern es kann auch Tippfehler im Code erkennen und hat einen Debugging-Modus.

Die zweite Sache, die ich in Angriff nehmen musste, war das Herunterladen aller notwendigen zusätzlichen Bibliotheken, die bereits im obigen Abschnitt Python und seine Bibliotheken erwähnt wurden. Um dies zu erreichen, musste ich die Eingabeaufforderung im Administratormodus öffnen und den folgenden Befehl eingeben:

- `path/to/folder/that/contains/python.exe -m pip install [Name der Bibliothek]`

Nach dem Drücken der Eingabetaste wurde die Bibliothek installiert und steht nun zur Verwendung bereit.

Als letztes habe ich die XAMPP-Anwendung genutzt, um über PHPMyAdmin auf die localhost-Schnittstelle zugreifen zu können, um eine Testumgebung für die Erstellung von Tabellen über die Python-Anwendung zu erstellen.

Erstellung der GUI - Hauptkonzept

Wie bereits in einem früheren Abschnitt erläutert, habe ich Tkinter verwendet, um eine einfach zu bedienende GUI für die Anwendung zu erstellen. Also habe ich zunächst ein Fenster erstellt und es der Variablen `root` zugewiesen. Ein Fenster ist einfach eine Instanz von TkinTERS TK-Klasse; mit anderen Worten, es ist der Rahmen, in dem die anderen Elemente angezeigt werden.

```
root = Tk()
```

Der nächste Teil bestand darin, die restlichen Seiten der Bewerbungen zu erstellen. Die Art und Weise, wie ich das gemacht habe, ist, dass jede Seite eigentlich eine Funktion ist, die entweder ein bestimmtes Element aus dem Stammfenster eingefügt oder entfernt. Die Hauptfunktion wird `start()` heißen und nicht nur die Funktion sein, die die erste Seite der Anwendung erzeugt, sondern sie wird auch alle anderen Funktionen enthalten, um den Übergang zwischen den Seiten zu erleichtern. Die Funktion `start()` selbst wird dann im Code aufgerufen, direkt vor der `root.mainloop()`. Diese Funktion weist Python an, die Tkinter-Ereignisschleife zu starten. Diese Methode lauscht auf Ereignisse, wie z.B. Tastenklicks oder Tastendrücke, und blockiert jeden nachfolgenden Code, bis man das Fenster schließt, in dem man die Methode aufgerufen hat.

```
start()  
  
root.mainloop()
```

Außerdem wird die Anwendung einige Variablen enthalten, denen zu Beginn ein Standardwert zugewiesen wird. Dies betrifft vor allem das gesamte Textmaterial, das sich ändern kann, je nachdem, ob der Nutzer beschließt, zu einer anderen Sprache als Englisch, der Standardsprache, zu wechseln.

```
language_pack = lp_en

error_input_empty = ""
error_input_contains_numbers = ""
error_input_contains_special_characters = ""
booking_url = language_pack.url_booking
scraping_in_progress = ""

order_question = language_pack.order_question
sort_alphabet = language_pack.sort_alphabet
sort_price = language_pack.sort_price
no_sort = language_pack.no_sort

storage_question = language_pack.storage_question
generate_file = language_pack.generate_file
generate_script = language_pack.generate_script
generate_all = language_pack.generate_all
upload_to_db = language_pack.db_upload

db_upload_question = language_pack.db_upload_question
db_host_input = language_pack.db_host_input_label
db_user_input = language_pack.db_user_input_label
db_password_input = language_pack.db_password_input_label
db_name_input = language_pack.db_name_input_label
db_send_data = language_pack.send_data

process_done = language_pack.process_done
```

```
restart = language_pack.restart
```

Erstellung der GUI - Das erste Fenster

Die erste Seite enthält die folgenden Elemente:

- ein Label, das den Benutzer auffordert, eine Sprache auszuwählen
- eine Liste mit 3 Optionsfeldern, die jeweils mit einer Sprache versehen sind
- eine Schaltfläche, um zum nächsten Fenster zu gelangen

Mit Ausnahme des Textes für die Optionsschaltflächen, der fest kodiert ist, beziehen alle anderen Elemente ihren Text von einer der oben genannten Variablen. Um ein Element zum Root-Fenster hinzuzufügen, muss man, wie im folgenden Codebeispiel zu sehen ist, einfach die Funktion pack() auf das Element der Wahl anwenden.

```
sel_lang = Label(text = language_pack.selector)

sel_lang.pack()


R1 = Radiobutton(root, text="English",
                  command= lambda: select_language("EN"))

R1.pack()

R2 = Radiobutton(root, text="Deutsch",
                  command= lambda: select_language("DE"))

R2.pack()

R3 = Radiobutton(root, text="Français",
                  command= lambda: select_language("FR"))

R3.pack()

R1.select()


label = Label(text=language_pack.selection)

label.pack()


go_next_to_second_page_button = Button(root, text=language_pack.next,
command=second_page)

go_next_to_second_page_button.pack()
```

Wie im früheren Codebeispiel zu sehen ist, ruft jede der Optionsschaltflächen dieselbe Methode auf, wobei ein anderer Zeichenfolgenwert als Parameter übergeben wird. Dies ist die Methode, die die Sprache der Anwendung ändern wird.

So funktioniert es: Für den Anfang habe ich einen Ordner im selben Verzeichnis erstellt, in dem sich die Python-Anwendung befindet, und ihn `language_packages` genannt. Innerhalb dieses Ordners habe ich drei weitere Python-Dateien erstellt, deren Benennung ich bereits in einem früheren Abschnitt erläutert habe. Diese Dateien haben alle die gleichen Variablen, denen jedoch unterschiedliche Textwerte zugewiesen sind.

Beispiel: Dies sind die ersten zwei Variablen aus jedem Sprachpaket.

```
selection = "Language selected: English"
selector = "Choose a language"
```

```
selection = "Gewählte Sprache: Deutsch"
selector = "Wählen Sie eine Sprache"
```

```
selection = "Langue sélectionnée: Français"
selector = "Choisissez une langue"
```

Die Funktion nimmt dann den von der Optionsschaltfläche übergebenen Parameter und durchläuft eine Reihe von ifs. Sobald die richtige Übereinstimmung gefunden ist, ersetzt die Methode den Wert der globalen Variable `language_pack` durch das entsprechende Sprachpaket.

```
def select_language(lp):
    global language_pack
    if lp == "EN":
        language_pack = lp_en
    elif lp == "DE":
        language_pack = lp_de
    elif lp == "FR":
        language_pack = lp_fr
```

Die Funktion ersetzt dann auch alle Werte für die anderen globalen Variablen; andernfalls würden sie in der Standardsprache bleiben. Das Startfenster hat auch eine Schaltfläche, die die Funktion `second_page()` aufruft und so zum nächsten Fenster führt.

Erstellung der GUI - Das zweite Fenster

Wie bereits erwähnt, wird die Anwendung nicht jedes Mal einen neuen Rahmen erstellen, sondern einfach Elemente aus demselben Rahmen entfernen und hinzufügen. Die Funktion `second_page()` entfernt also zunächst alle Elemente der Ausgangsseite und fügt dann ihre eigenen hinzu. Das Entfernen von Elementen kann mit der Methode `destroy()` erreicht werden. Dieser Schritt wird für alle Funktionen, die als Fensterlader arbeiten, wiederholt.

```
def second_page():  
    sel_lang.destroy()  
    R1.destroy()  
    R2.destroy()  
    R3.destroy()  
    label.destroy()  
    go_next_to_second_page_button.destroy()  
    location_question.pack()  
    location_input.pack()  
    error_label.pack()  
    scrape_button.pack()  
    go_back_to_start_page_button.pack()
```

Wie im obigen Code ersichtlich, wird das zweite Fenster 5 Elemente enthalten, nämlich:

- ein Label, das den Benutzer fragt, in welcher Stadt er eine Liste von Hotels haben möchte
- ein Eingabefeld, in das der Benutzer seine Antwort eingeben kann
- ein zweites Label, die standardmäßig leer ist, deren Text sich aber je nach der eingegebenen Antwort ändert
- eine Schaltfläche, die den Scraping-Prozess durch den Aufruf der `scrape()`-Funktion startet, die als Parameter die URL aus dem Sprachpaket sowie die vom Nutzer übermittelte Antwort erhält
- eine zweite Schaltfläche, mit der der Benutzer zur Startseite zurückkehren kann

```
# Elemente der zweite Seite  
go_back_to_start_page_button = Button(root, text = language_pack.back,  
command=start_page)  
  
location_question = Label(text = language_pack.location_question)  
location_input = Entry()  
  
scrape_button = Button(text = language_pack.start_scraping,  
command=check_input)  
  
error_label = Label()
```

Die Schaltfläche Scrape ruft die Funktion `check_input()` auf. Bevor der eigentliche Scraping-Prozess beginnt, wird die vom Benutzer eingegebene Eingabe überprüft. Die Methode prüft zunächst, ob tatsächlich etwas eingegeben wurde; gleichzeitig wird geprüft, ob die Eingabe nur aus Leerzeichen besteht. Sollte dies der Fall sein, wird der Scrape-Vorgang nicht gestartet, und die zuvor leere Fehlermeldung informiert den Benutzer darüber, warum seine Eingabe nicht akzeptiert wurde.

Die Funktion wiederholt diesen Vorgang zwei weitere Male, um zu prüfen, ob die Eingabe Zahlen oder Sonderzeichen enthält. Wird eines der beiden Zeichen gefunden, wird die Eingabe verweigert und eine Fehlermeldung angezeigt. Erst wenn die Eingabe alle diese Prüfungen bestanden hat, wird sie akzeptiert und der Scraping-Prozess beginnt. Diese Funktion lädt auch das dritte Fenster, indem sie die Methode `third_page()` aufruft, sobald das Scraping abgeschlossen ist.

```
def check_input():

    global error_input_empty

    global error_input_contains_numbers

    global error_input_contains_special_characters

    def containsNumber(value):

        if True in [char.isdigit() for char in value]:

            return True

        else:

            return False

    def containsSpecialCharacter(value):

        if all(char.isalpha() or char.isspace() for char in value):

            return False

        else:

            return True

    search_term = location_input.get()

    if search_term == "" or search_term.isspace():

        error_label.config(text = error_input_empty)

    elif containsNumber(search_term) == True:

        error_label.config(text = error_input_contains_numbers)

    elif containsSpecialCharacter(search_term) == True:

        error_label.config(text = error_input_contains_special_characters)

    else:
```

```
error_label.config(text = scraping_in_progress)

scrape_booking(booking_url, search_term)

third_page()
```

Der Scraping-Prozess¹⁵

Die Funktion `scrape(url, input)` gehört eigentlich zu einer anderen Datei. Ich habe den Code, der die GUI kompiliert, und den, der den Scraper ausführt, aufgeteilt, um zu verhindern, dass eine Datei zu lang und damit schwer zu lesen ist. Der erste Teil des Scraping-Prozesses macht von den folgenden Bibliotheken Gebrauch: Selenium und sein WebDriver, Time, BeautifulSoup und Requests.

Der WebDriver von Selenium ist einfach eine Schnittstelle, über die das Programm mit einem Webbrowser, in diesem Fall Chrome, interagieren kann. Dieser WebDriver hat auch verschiedene anpassbare Optionen. Im Fall dieses Projekts habe ich die Option hinzugefügt, um zu verhindern, dass irrelevante Meldungen die Konsole überfluten, und die Automatisierungs-Erweiterungen deaktiviert, eine Funktion, die, wenn sie aktiviert ist, die Wahrscheinlichkeit erhöht, dass der Scraper als Bot erkannt und somit blockiert wird. Die Methode `get(url)` bewirkt, dass die Anwendung eine aktuelle Seite in Chrome öffnet, die der als Parameter angegebenen URL folgt.

```
options = webdriver.ChromeOptions()

options.add_argument("start-maximized")

options.add_experimental_option('excludeSwitches', ['enable-logging'])

options.add_experimental_option('useAutomationExtension', False)

driver = webdriver.Chrome(options=options)

driver.get(url)
```

Damit die Anwendung mit einem Element auf der Seite interagieren kann, muss dieses Element zunächst gefunden werden. Dies kann auf zwei Arten geschehen, indem entweder über den XPath oder über den CSS-Selektor nach dem Element gesucht wird.

XPath steht für XML Path. Es ist eine Abfragesprache, die dabei hilft, Elemente in einem XML-Dokument zu identifizieren. Sie verwendet Ausdrücke, die in einem XML-Dokument auf eine Weise navigieren, die von Anfang an bis zum gewünschten Element zurückverfolgt werden kann - so, als würde man einen Pfad von Anfang an bilden. Die Vorteile der Verwendung von XPATHs sind:

¹⁵ Ein Teil des Codes wurde teilweise aus dem Code des folgenden Tutorials übernommen. Der Code wurde jedoch in einigen Teilen angepasst, damit er zum Rest des Projekts passt. How to Scrape Booking.com (2023 Update), ScrapFly Team, <https://scrapfly.io/blog/how-to-scrape-bookingcom/>

- XPATH ermöglicht die Navigation im DOM bei der Suche nach Elementen zum Testen oder Scrapen
- Es ist kompatibel mit alten Browsern
- Die Erstellung in XPath ist flexibler als im CSS Selector
- Wenn man den Namen eines Elements nicht kennt, kann man contains verwenden, um nach möglichen Übereinstimmungen zu suchen

Die meisten HTML-Seiten werden mithilfe von Cascading-Style-Sheet-Klassen, auch CSS genannt, gestaltet. Um die verschiedenen Elemente auf einer Seite anhand von Stilen zu identifizieren, muss man die Klasse auswählen, in die sie fallen. Ein CSS-Selektor kann als der Teil des Stylesheets betrachtet werden, der es ermöglicht, die Art des Inhalts auszuwählen, den man entweder testen, bearbeiten oder kopieren möchte. Die CSS-Selektoren identifizieren die verschiedenen Elemente im DOM und wirken sich auf diese Teile der Benutzeroberfläche aus oder stellen eine Verbindung zu ihnen her.

CSS-Selektoren gibt es in vielen Varianten. Das liegt vor allem daran, dass Selektoren im Gegensatz zum Baum- oder Map-Aufbau der XPath-Option echte Namen und Kategorien haben. Einige der Typen sind:

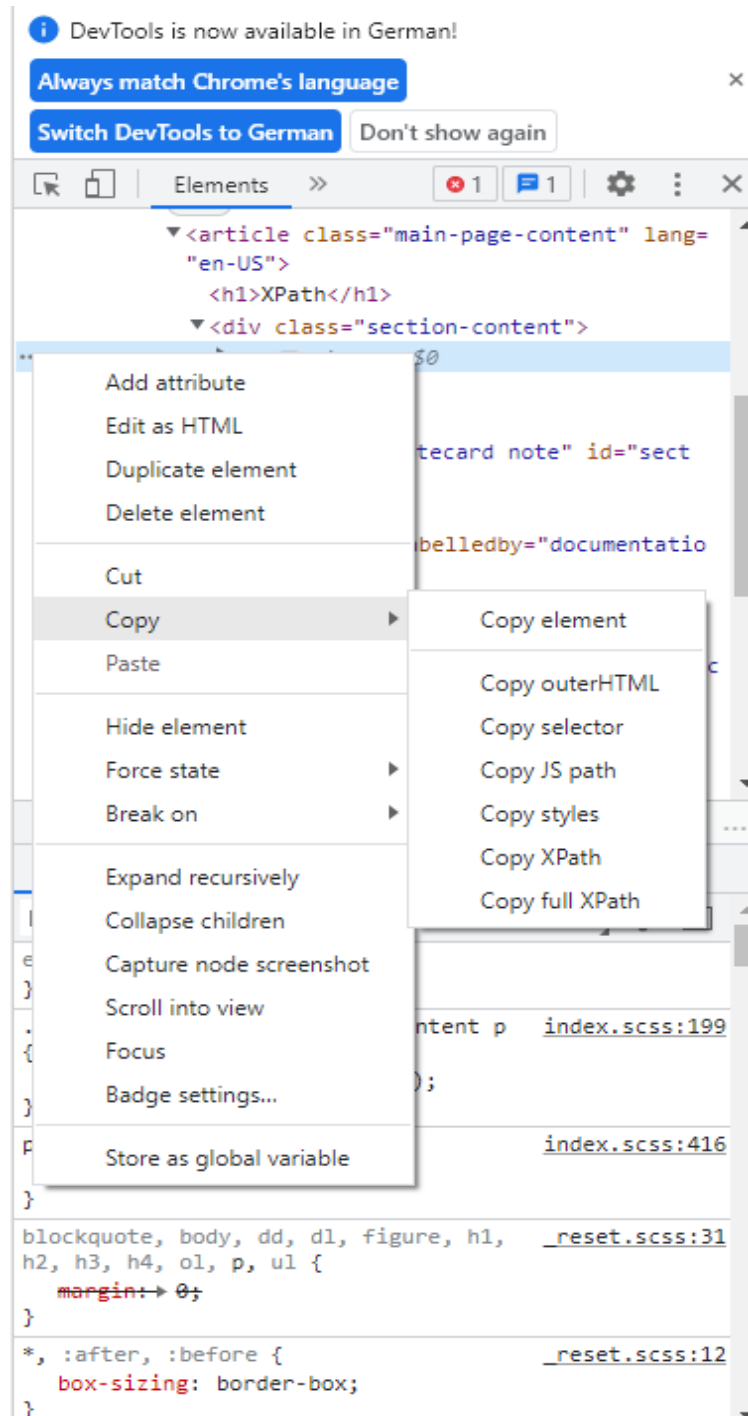
- Einfache Selektoren: Diese suchen nach Elementen auf der Grundlage ihrer Klasse oder ID
- Attribut-Selektoren: Sie suchen nach Elementen auf der Grundlage der ihnen zugewiesenen Werte
- Pseudo-Selektoren: In Situationen, in denen der Zustand von Elementen mit CSS deklariert wird, wie z. B. bei Kontrollkästchen oder On-Hover-Attributen, kommen diese zum Einsatz

Die Vorteile der Verwendung von CSS-Selektoren sind:

- Es ist schneller als XPath
- Es ist viel einfacher zu lernen und zu implementieren
- Man hat eine hohe Chance, die gewünschten Elemente zu finden
- Es ist mit den meisten bisherigen Browsern kompatibel

Aufgrund dieser oben genannten Vorteile habe ich mich entschieden, den CSS-Selektor als Referenz zu verwenden, um die Elemente zu finden, mit denen ich interagieren möchte. Es ist sehr einfach, den CSS-Selektor eines Elementes zu erhalten. In Chrome muss man nur auf die Seite gehen, die man scrapen will, mit der rechten Maustaste auf das Element klicken, mit dem die App interagieren soll, und dann auf Untersuchen klicken.

Dadurch öffnen sich auf der rechten Seite des Browserfensters die DevTools. Dort ist das Element, das man untersuchen möchte, grau hinterlegt. Nun muss man nur noch auf die drei Punkte links neben dem HTML-Code des Elements klicken, den Mauszeiger auf die Option Kopieren setzen und dann auf Selektor kopieren klicken. Dadurch wird der Selektor des Elementes in der Zwischenablage gespeichert. Dann kann er einfach in den Code eingefügt werden.



Sobald die `find_element()`-Methode ausgefüllt ist, muss man nur noch die Aktion hinzufügen, die die App ausführen soll. Zum Beispiel wird meine Anwendung beim ersten gefundenen Element die Aktion `send_keys(input)` ausführen. `Send_keys()`, wie der Name schon andeutet, lässt die Anwendung in das gewählte Element, in diesem Fall ein Suchfeld, den Text eingeben, der der Methode als Parameter übergeben wurde, in diesem Fall den Namen der vom Benutzer angegebenen Stadt. Da das Laden von Websites je nach Verbindung eine Weile dauern kann, habe ich beschlossen, die Methode `time.sleep(5)` zu verwenden, damit die Anwendung 5 Sekunden wartet, bevor sie den nächsten Schritt ausführt. Denn wenn die Anwendung versucht, ein Element zu finden, das noch nicht geladen wurde, tritt ein Fehler auf und die Anwendung friert ein.

```
driver.find_element(by=By.CSS_SELECTOR, value='#\:Ra9\:').send_keys(input)

time.sleep(5)
```

Der gleiche Vorgang wurde auch für die anderen Elemente wiederholt, mit dem einzigen Unterschied, dass die Funktion click() anstelle der Funktion send_keys() verwendet wurde, bis der Browser zur Ergebnisseite gelangt.

```
driver.find_element(by=By.CSS_SELECTOR, value='#b2indexPage >
div.c85f9f100b.cb6c8dd99f > div > div > div > div.dabce2e809 >
div.b8ef7618ca.b2ed5869b1.b78b2b2275.e8efa318be > div > button').click()

time.sleep(5)

driver.find_element(by=By.CSS_SELECTOR, value='#indexsearch >
div.hero-banner-searchbox > div > div > form >
div.ffa9856b86.db27349d3a.cc9bf48a25 > div:nth-child(2) > div > div >
button:nth-child(2)').click()

time.sleep(5)

driver.find_element(by=By.CSS_SELECTOR, value='#indexsearch >
div.hero-banner-searchbox > div > div > form >
div.ffa9856b86.db27349d3a.cc9bf48a25 > div:nth-child(2) > div >
div.dec3155a06 > div > nav > div.bc13adc8ab.cd743661d2 > ul >
li:nth-child(2) > button').click()

time.sleep(5)

driver.find_element(by=By.CSS_SELECTOR,
value='#flexible-searchboxdatepicker >
div.alb3f50dcd.efdd20a9c1.b2fe1a41c3.c1b465858f.d46673fe81 >
div:nth-child(1) > div > div.alb3f50dcd.a1f3ecff04.b2fe1a41c3 >
label:nth-child(2) > span').click()

time.sleep(5)

driver.find_element(by=By.CSS_SELECTOR,
value='#flexible-searchboxdatepicker >
div.alb3f50dcd.efdd20a9c1.b2fe1a41c3.c1b465858f.d46673fe81 >
div:nth-child(2) > div > div.alb3f50dcd.a1f3ecff04.b2fe1a41c3 >
label:nth-child(1) > span').click()

time.sleep(5)

driver.find_element(by=By.CSS_SELECTOR, value='#indexsearch >
div.hero-banner-searchbox > div > div > form >
div.ffa9856b86.db27349d3a.cc9bf48a25 > div:nth-child(4) > button').click()

time.sleep(5)
```

Hier kommt die Funktion `extract results()` ins Spiel, zusammen mit den Bibliotheken `BeautifulSoup` und `Pandas`. Die Arbeit wird zunächst wieder größtenteils von `Selenium` erledigt: Dieses Mal wird die Funktion `find_element()` verwendet, um auszuwählen, welche Elemente des gleichen Typs ich durchlaufen möchte. Dann werden die Textwerte der darin enthaltenen Elemente abgerufen, sodass sie Variablen zugewiesen und in einem `Pandas`-Datenrahmen gespeichert werden können. In diesem Fall sind die Elemente desselben Typs, die das Programm durchlaufen wird, die Eigenschaftskarten. Von dort wird `Selenium` extrahieren und in einer Variablen speichern:

- der Name der Immobilie
- den Link zu seinem Buchungsangebot
- der Durchschnittspreis
- das als Symbol in der Karte verwendete Bild, gespeichert unter der Form seines Quelllinks

```
def extract_results():
    result_cards = driver.find_elements(by=By.CSS_SELECTOR,
value='div[data-testid="property-card"]')

    new_result = {}

    for result in result_cards:
        name = result.find_element(by=By.CSS_SELECTOR,
value='div[data-testid="title"]').text.strip()

        link =
result.find_element(By.CSS_SELECTOR, 'a[data-testid="title-link"]').get_attri
bute('href')

        price = result.find_element(By.CSS_SELECTOR,
'[data-testid="price-and-discounted-price"]').text

        image = result.find_element(By.CSS_SELECTOR,
'[data-testid="image"]').get_attribute('src')
```

Um die Adresse, Stadt und das Land der Immobilie zu erhalten, ist ein zusätzlicher Schritt erforderlich, und hier kommen `BeautifulSoup` und `Requests` ins Spiel. Durch die Verwendung der eigenen `get()`-Funktion und den von `Selenium` gesammelten Link als Parameter erhält `Requests` die Seite der Eigenschaft als Text, den `BeautifulSoup` dann in HTML zergliedert. `BeautifulSoup` findet dann das Element mit den Daten und weist es der Variablen als String zu. Dieser String wird dann mithilfe der Methode `split()` in ein Array umgewandelt, das den Wert an jedem darin enthaltenen Komma teilt.

```
hotel_url = requests.get(link)

doc = BeautifulSoup(hotel_url.text, 'html.parser')

get_address = doc.find('span', class_='hp_address_subtitle
js-hp_address_subtitle jq_tooltip').text

address = get_address.split(', ')
```

Von dort wird eine Reihe von Regeln verwendet, um die richtigen Daten zu extrahieren. Wenn eines der Elemente im Array 4 oder mehr Zahlen enthält, wird es den Stadt-Variablen zugewiesen. Wenn sich herausstellt, dass der erste Wert des Arrays die Stadt selbst ist und somit die eigentliche Straßenadresse fehlt, wird der Street-Variablen ein Platzhaltertext zugewiesen, der zu Beginn der ausgewählten Sprache entspricht. Ist dies nicht der Fall, wird der Street-Variable der erste Wert zugewiesen. Dann wird das letzte Element des Arrays der Country-Variablen zugewiesen.

```
for element in address:

    numbers_in_string = sum(character.isdigit() for character in
element)

    if numbers_in_string >= 4:

        city = element.strip()

if address[0] == city:

    if url == lp_en.url_booking:

        street = "Not available"

    elif url == lp_de.url_booking:

        street = "Nicht verfügbar"

    elif url == lp_fr.url_booking:

        street = "Non disponible"

else:

    street = address[0].strip()

last_element = len(address) - 1

country = address[last_element].strip()
```

Alle Variablen werden dann in einem Wörterbuch gespeichert, das dann selbst als neue Zeile zum Pandas-Datenrahmen hinzugefügt wird.

```
new_result = {'name': name, 'address': street, 'city': city,
'country': country, 'link': link, 'price': price, 'image': image}

dataframe.loc[len(dataframe)] = new_result
```

Erstellung der GUI - Das dritte Fenster und der Sortierprozess

Das dritte Fenster wird einfach 5 Elemente enthalten:

- ein Label, das den Benutzer fragt, wie er die gesammelten Daten sortieren möchte
- eine Schaltfläche, die die Daten alphabetisch sortiert, wenn sie angeklickt wird
- eine zweite Schaltfläche, die beim Anklicken die Daten nach den Preisen sortiert
- eine dritte Schaltfläche, die die Reihenfolge der Daten überhaupt nicht verändert

Unabhängig davon, welche dieser drei Schaltflächen gedrückt wird, wird die vierte Seite geladen, sobald der Sortiervorgang abgeschlossen ist.

```
order_question_label = Label(text = order_question)

sort_alphabetically_button = Button(text = sort_alphabet,
command=sort_by_abc)

sort_price_button = Button(text = sort_price, command=sort_by_pricing)

no_sort_button = Button(text = no_sort, command=fourth_page)
```

Die Funktionen zum Sortieren der Daten sind in demselben Dokument gespeichert, in dem sich auch die Scraping-Funktion befindet. Die Sortierung erfolgt einfach durch Verwendung der Methode `sort_values()` und Zuweisung des Namens der gewünschten Spalte an das Attribut `by`.

```
def sort_alphabetically():

    dataframe.sort_values('name')

def sort_by_price():

    dataframe.sort_values('price', ascending=True)
```

Erstellung der GUI - Das vierte Fenster und die Speicherung von Daten

Das vierte Fenster besteht aus 5 Elementen:

- ein Label, das den Benutzer fragt, wie er die gesammelten Daten speichern möchte
- eine Schaltfläche, die bei Betätigung eine CSV-Datei erzeugt
- eine zweite Schaltfläche, die bei Betätigung eine SQL-Skriptdatei erzeugt
- eine dritte Schaltfläche, die bei Betätigung sowohl eine CSV- als auch eine SQL-Skriptdatei erzeugt
- eine vierte Schaltfläche, mit der das fünfte Fenster geladen wird

Abgesehen von der letzten Schaltfläche laden alle Schaltflächen auch die letzte Seite, sobald die Datei(en) erzeugt wurde(n).

```
storage_question_label = Label(text = storage_question)

create_csv_button = Button(text = generate_file, command=generate_csv)

create_sql_button = Button(text = generate_script, command=generate_sql)
```

```
create_both_button = Button(text = generate_all, command=generate_both)
upload_to_local_db_button = Button(text = upload_to_db, command=fifth_page)
```

Genau wie die Sortierfunktionen sind auch die Speicherfunktionen in derselben Datei wie die Scraping-Funktion gespeichert. Das Erstellen einer CSV-Datei aus dem Pandas-Datenrahmen ist eigentlich ganz einfach, da der Datenrahmen selbst eine Methode namens `to_csv()` besitzt, die diese Aufgabe übernimmt.

```
def create_csv():
    global searched_city
    dataframe_name = searched_city + "_scraped_data.csv"
    folder = "CSVs"
    storage_path = os.path.join(folder, dataframe_name)
    dataframe.to_csv(storage_path, index=False, encoding='utf-8')
```

Für das SQL-Skript muss man einfach den gleichen Code verwenden, den man auch für das Erstellen und Schreiben einer Textdatei verwenden würde, man muss nur die Erweiterung der Datei in SQL ändern.

```
def create_sql():
    global searched_city
    table_name = searched_city + '_scraped_data'
    folder = "SQLs"
    storage_path = os.path.join(folder, table_name)
    counter = 0
    id = 1

    f = open(storage_path + '.sql', "a", encoding="utf-8")

    f.write(f'CREATE TABLE `{table_name}` (`ID` int(7) NOT NULL
    AUTO_INCREMENT, `name` varchar(255) NOT NULL, `address` varchar(255) NOT
    NULL, `city` varchar(255) NOT NULL, `country` varchar(255) NOT NULL, `link`
    text NOT NULL, `price` varchar(255) NOT NULL, `image` text NOT NULL, PRIMARY
    KEY (ID)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_general_ci;')

    f.write(f'INSERT INTO `{table_name}` (`ID`, `name`, `address`, `city`,
    `country`, `link`, `price`, `image`) VALUES')
```

```

for element in range(len(dataframe.index)):

    if id != len(dataframe.index):

        f.write("(" + str(id) + ", '" +
dataframe.at[dataframe.index[counter], 'name'] + "', '" +
dataframe.at[dataframe.index[counter], 'address'] + "', '" +
dataframe.at[dataframe.index[counter], 'city'] + "', '" +
dataframe.at[dataframe.index[counter], 'country'] + "', '" +
dataframe.at[dataframe.index[counter], 'link'] + "', '" +
dataframe.at[dataframe.index[counter], 'price'] + "', '" +
dataframe.at[dataframe.index[counter], 'image'] + "')"

        f.write(",")

        counter += 1

        id += 1

    else:

        f.write("(" + str(id) + ", '" +
dataframe.at[dataframe.index[counter], 'name'] + "', '" +
dataframe.at[dataframe.index[counter], 'address'] + "', '" +
dataframe.at[dataframe.index[counter], 'city'] + "', '" +
dataframe.at[dataframe.index[counter], 'country'] + "', '" +
dataframe.at[dataframe.index[counter], 'link'] + "', '" +
dataframe.at[dataframe.index[counter], 'price'] + "', '" +
dataframe.at[dataframe.index[counter], 'image'] + "')"

        f.write(";")

```

Erstellung der GUI - Das fünfte Fenster und das Hochladen von Daten in die Datenbank

Das fünfte Fenster enthält 2 unabhängige Elemente und 4 Paare von Elementen. Diese Paare bestehen jeweils aus einem Label und einem Eingabefeld. Die 2 unabhängigen sind:

- ein Label am oberen Rand des Fensters, das dem Benutzer sagt, was er zu tun hat
- eine Schaltfläche am unteren Rand des Fensters, die, wenn sie gedrückt wird, wenn alle Eingaben korrekt sind und die Anwendung eine Verbindung mit der gewünschten Datenbank herstellen kann, eine Tabelle erstellt und alle gesammelten Daten in diese einfügt

Die genannten Paare von Elementen sind für:

- den Host, auf dem die Zieldatenbank gehostet wird
- den Benutzer, den Namen des Kontos, mit dem Sie auf die genannte Datenbank zugreifen
- das Passwort, das man benutzt, um sich in das Konto einzuloggen
- den Namen der Datenbank, in der die Tabelle erstellt werden soll

Nachdem die Tabelle erstellt und die Daten eingefügt wurden, wird das letzte Fenster geladen.

```

db_upload_question_label = Label(text = db_upload_question)

db_host_input_label = Label (text = db_host_input)

db_host_input_field = Entry()

db_user_input_label = Label (text = db_user_input)

db_user_input_field = Entry()

db_password_input_label = Label (text = db_password_input)

db_password_input_field = Entry()

db_name_input_label = Label (text = db_name_input)

db_name_input_field = Entry()

db_send_data_button = Button(text = db_send_data, command=send_data)

```

Die Schaltfläche Daten senden ruft die Funktion send_data() auf. In dieser Funktion wird jeder der in den Eingabefeldern eingegebenen Werte einer eigenen Variablen zugewiesen, die dann als Parameter für die Funktion upload_to_database() verwendet wird.

```

def send_data():

    host = db_host_input_field.get()

    user = db_user_input_field.get()

    password = db_password_input_field.get()

    name = db_name_input_field.get()


    upload_to_database(host, user, password, name)

    sixth_page()

```

Diese Methode befindet sich, wie viele andere zuvor erwähnte, in einer anderen Datei als die GUI. Diese Methode nutzt die PyMySQL-Bibliothek, um zunächst eine Verbindung zur gewünschten Datenbank herzustellen, indem sie die übermittelten Eingaben als Argumente verwendet und dann SQL-Abfragen ausführt, die zunächst eine Tabelle erstellen und dann durch die im Pandas-Datenrahmen gesammelten Daten iterieren und sie in diese Tabelle einfügen.

```

def upload_to_database(h, u, p, d):

    global searched_city

    try:

        # Connect to the database

```

```

connection = pymysql.connect(host= h,

                               user= u,

                               password= p,

                               db= d)

cursor=connection.cursor()

counter = 0

table_name = searched_city + '_scraped_data'

sql_create = f'CREATE TABLE `{table_name}` (`ID` int(7) NOT NULL
AUTO_INCREMENT, `name` varchar(255) NOT NULL, `address` varchar(255) NOT
NULL, `city` varchar(255) NOT NULL, `country` varchar(255) NOT NULL, `link`
text NOT NULL, `price` varchar(255) NOT NULL, `image` text NOT NULL, PRIMARY
KEY (ID)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci'

cursor.execute(sql_create)

sql = f"INSERT INTO `{table_name}` (`name`, `address`, `city`,
`country`, `link`, `price`, `image`) VALUES (%s, %s, %s, %s, %s, %s, %s)"

for element in range(len(dataframe.index)):

    cursor.execute(sql, (dataframe.at[dataframe.index[counter],
'name'],dataframe.at[dataframe.index[counter],
'address'],dataframe.at[dataframe.index[counter], 'city'],
dataframe.at[dataframe.index[counter],
'country'],dataframe.at[dataframe.index[counter], 'link'],
dataframe.at[dataframe.index[counter], 'price'],
dataframe.at[dataframe.index[counter], 'image'])))

    counter += 1

connection.commit()

except Error as e:

    print(e)

finally:

    connection.close()

```

Erstellung der GUI - Das letzte Fenster und Neustart der Anwendung

Das letzte Fenster ist im Vergleich zum vorherigen sehr minimalistisch und enthält nur zwei Elemente:

- ein Label, das das Ende des Anwendungsprozesses ankündigt

- eine Schaltfläche, die, wenn sie gedrückt wird, den Benutzer zum ersten Fenster zurückbringt und den Pandas-Datenrahmen zurücksetzt

Wenn der Benutzer keine Verwendung mehr für die Anwendung hat, kann er sie einfach bearbeiten, während er, wenn er eine andere Stadt nach Informationen durchsuchen möchte, einfach auf die Schaltfläche klicken kann.

```
process_done_label = Label(text = process_done)

restart_button = Button(text = restart, command=restart_app)
```

Die von der Schaltfläche aufgerufene Methode `restart_app()` zerstört einfach die beiden Elemente, ruft die Methode `reset_dataframe()` aus der anderen Python-Datei und die Methode `start()` auf.

```
def restart_app():

    process_done_label.destroy()

    restart_button.destroy()

    reset_dataframe()

    start()
```

```
def reset_dataframe():

    dataframe.iloc[:0]
```

2.11 Testfall-Protokolle

Nr.	Gegenstand des Testfalls	Beschreibung des Testfalls	Voraussetzungen
1	Sprachauswahl	Prüfen, ob alle Texte in der gewünschten Sprache geladen werden.	1. Python-Anwendung erfolgreich gebootet
2.1	Scraper: Öffnung von Booking	Die Python-App soll in Chrome eine Seite von Booking öffnen, nachdem eine bestimmte Schaltfläche angeklickt wurde	1. Zweites Fenster wurde geladen 2. Akzeptable Eingaben wurden in das Eingabefeld eingefügt 3. Die Schaltfläche „Scraper starten“ wurde angeklickt
2.2	Scraper: Navigation	Der Scraper gibt den vom Benutzer angegebenen Wert korrekt in das Suchfeld ein und führt alle erforderlichen Schritte aus, um zur Ergebnisseite zu gelangen.	1. Booking wurde vom Scraper erfolgreich geöffnet
2.3	Scraper: Datensammlung	Der Scraper navigiert erfolgreich durch alle Ergebnisseiten und	1. Die erste Ergebnisseite wurde

		sammelt die relevanten Daten von ihnen.	vom Scraper erfolgreich erreicht
3.1	Sortierung: Alphabetisch	Die Anwendung sollte in der Lage sein, die Elemente innerhalb des Datenrahmens alphabetisch zu sortieren.	1. Scraping-Vorgang erfolgreich beendet 2. Dritte Seite wurde geladen
3.2	Sortierung: Preise (aufsteigend)	Die Anwendung sollte in der Lage sein, die Elemente innerhalb des Datenrahmens nach ihrem Preis zu sortieren.	1. Scraping-Vorgang erfolgreich beendet 2. Dritte Seite wurde geladen
4.1	Dateierzeugung: CSV	Die Anwendung kann eine CSV-Datei mit den vom Scraper gesammelten Daten erstellen.	1. Scraping-Vorgang erfolgreich beendet 2. Vierte Seite wurde geladen
4.2	Dateierzeugung: SQL-Skript	Die Anwendung kann eine SQL-Skript-Datei mit den vom Scraper gesammelten Daten erstellen.	1. Scraping-Vorgang erfolgreich beendet 2. Vierte Seite wurde geladen
5	Datenbank-Upload	Die Anwendung kann die gesammelten Daten in Form einer Tabelle erfolgreich in eine Datenbank hochladen.	1. Scraping-Vorgang erfolgreich beendet 2. Fünfte Seite wurde geladen
6	Neu starten	Die Anwendung kann per Knopfdruck neu gestartet werden.	1. Letzte Seite wurde geladen

Testfall 1

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. Schaltfläche "Deutsch" wurde angeklickt	Die GUI-Sprache wurde von Englisch auf Deutsch geändert.	Die GUI-Sprache wurde von Englisch auf Deutsch geändert.	Ursprünglich hatte jede Sprache ihre eigene Set-Methode. Ich beschloss, das zu ändern und machte es so, dass nur eine Methode für die Änderung der Sprache verantwortlich ist.

Testfall 2.1

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. "Villigen" in das Eingabefeld eingegeben 2. Schaltfläche "Scraper starten" wurde angeklickt	Die Python-Anwendung hat Booking in Chrome geöffnet.	Die Python-Anwendung hat Booking in Chrome geöffnet.	Nichts Nennenswertes zu berichten.

Testfall 2.2

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
"Villigen" in das Eingabefeld eingegeben - Schaltfläche "Scraper starten" wurde angeklickt	- Scraper hat im Suchfeld "Villigen" eingegeben - Scraper hat auf die Kalender Schaltfläche geklickt - Scraper hat einen zufälligen Zeitraum ausgewählt - Scraper hat auf die Suchschaltfläche geklickt	- Scraper hat im Suchfeld "Villigen" eingegeben - Scraper hat auf die Kalender Schaltfläche geklickt - Scraper hat einen zufälligen Zeitraum ausgewählt - Scraper hat auf die Suchschaltfläche geklickt	Manchmal friert die Anwendung ein. Dies liegt daran, dass die Website anders als gewöhnlich geladen wird. Manchmal lädt es eine Version, die einige der Schaltflächen im Kalenderfenster nicht hat, mit denen die Anwendung interagieren soll. Habe versucht, einen schnellen Weg darum herum zu finden, war aber erfolglos.

Testfall 2.3

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
"Villigen" in das Eingabefeld eingegeben - Schaltfläche "Scraper starten" wurde angeklickt - Der Scraper navigiert zu den Ergebnisseiten	- Der Scraper hat relevante Daten von der ersten Seite an gesammelt - Der Scraper hat zur nächsten Ergebnisseite navigiert - Der Scraper hat Daten von allen Ergebnisseiten gesammelt	- Der Scraper hat relevante Daten von der ersten Seite an gesammelt - Der Scraper hat zur nächsten Ergebnisseite navigiert - Der Scraper hat Daten von allen Ergebnisseiten gesammelt	Es war vielleicht nur ein einmaliger Fall, aber ich bemerkte, dass der Scraper nicht mehr funktionierte, als ich versuchte, eine Stadt mit mehr Hotels als gewöhnlich zu scrapern. Habe vergessen zu notieren, bei wie vielen Hotels es kaputt gegangen ist. Dieses Problem hat sich bisher nicht wiederholt.

Testfall 3.1

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. Schaltfläche alphabetisch sortieren wurde angeklickt	Elemente innerhalb des Datenrahmens wurden alphabetisch sortiert.	Elemente innerhalb des Datenrahmens wurden nicht alphabetisch sortiert.	Ich habe versucht, in den Code zu schauen, aber es scheint keinen Fehler zu geben. Warum die Methode nicht das tut, was sie soll, ist mir unbekannt.

Testfall 3.2

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. Schaltfläche nach Preis sortieren wurde angeklickt	Elemente innerhalb des Datenrahmens wurden nach Preis sortiert.	Elemente innerhalb des Datenrahmens wurden nicht nach Preis sortiert.	Die Methode teilt das gleiche Problem wie die vorherige. Ursache konnte nicht festgestellt werden.

Testfall 4.1

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. Schaltfläche CSV generieren wurde angeklickt	Im Ordner CSVs wurde eine neue CSV-Datei erstellt.	Im Ordner CSVs wurde eine neue CSV-Datei erstellt.	Nichts Nennenswertes zu berichten

Testfall 4.2

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. Schaltfläche SQL generieren wurde angeklickt	Im Ordner SQLs wurde eine neue SQL-Datei erstellt.	Im Ordner SQLs wurde eine neue SQL-Datei erstellt.	Nichts Nennenswertes zu berichten

Testfall 5

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
1. Schaltfläche Hochladen in eine Datenbank wurde angeklickt 2. Im ersten Eingabefeld wurde	1. In my_db ist eine neue Tabelle erschienen 2. Letzte Seite wurde geladen	1. In my_db ist eine neue Tabelle erschienen 2. Letzte Seite wurde geladen	Obwohl ich nicht die Zeit hatte, eine Methode zu implementieren, die die Ausnahme abfängt, wenn falsche Eingaben

"localhost" eingetragen 3. Im zweiten Eingabefeld wurde "root" eingetragen 4. Das dritte Eingabefeld wurde leer gelassen 5. Im vierten Eingabefeld wurde „my_db“ eingetragen 6. Schaltfläche Daten senden wurde angeklickt			gesendet werden, bricht die Anwendung nicht ab. Bei falscher Eingabe bleibt die Bewerbung einfach auf der fünften Seite stehen, anstatt die letzte Seite zu laden. Wenn danach die richtige Eingabe erfolgt, werden die Daten normal hochgeladen und dann die letzte Seite geladen.
---	--	--	---

Testfall 6

Eingabe	Erwartetes Ergebnis	Erhaltenes Ergebnis	Bemerkungen
Schaltfläche Neu Starten wurde angeklickt	Die Anwendung ist zur Seite des Sprachauswahlmenüs zurückgekehrt.	Die Anwendung ist zur Seite des Sprachauswahlmenüs zurückgekehrt.	Nichts Nennenswertes zu berichten

2.12 Fazit

Allgemeine Erfahrung

Insgesamt war die Arbeit an einem Web Scraper eine interessante Erfahrung. Ich war mir der Tools bewusst, die Informationen aus dem Internet für sich selbst sammeln konnten, aber ich hatte immer das Gefühl, dass sie extrem komplex waren, während sie in Wirklichkeit, sobald man den Dreh raus hat, ziemlich lustige und einfach zu machende Projekte sind. Das heißt, ich bedauere das Projekt auch etwas.

Die Fehlschläge des Projekts

Ich denke, dass alles, was in diesem Projekt entweder schiefgelaufen ist oder nicht erfüllt wurde, einen gemeinsamen Nenner hat: schlechte Zeitschätzung. Damit meine ich nicht nur die Aufgaben allein, sondern das Projekt als Ganzes. Ich dachte, ich würde alles schaffen, wenn auch mit einigem Kampf, aber es stellte sich heraus, dass ich die Menge an Unerwartetem stark unterschätzt hatte. Der Mangel an genügend Zeit hat dem Projekt den Mangel folgendes gebracht:

- Reduziert von einer Liste von URLs auf nur eine URL als Seed:** Dies wurde hauptsächlich durch den sehr starken Bot-Schutz verursacht, den viele der ausgewählten Zielseiten hatten. Ich habe versucht, verschiedene Lösungen zu implementieren, aber keine davon hat wirklich funktioniert, vielleicht weil ich so in Eile war und den notwendigen Code nicht richtig eingebaut habe. Das Umgehen der

Bot-Erkennung lag völlig außerhalb meines Wissens, da ich mich nicht auf das Thema vorbereitet hatte. Das Erlernen dieses völlig neuen Themas im Detail hätte viel wertvolle Zeit gekostet, weshalb ich mich entschied, das Problem schließlich zu umgehen, indem ich die Ziel-URL auf nur eine reduzierte, von der ich wusste, dass sie keinen allzu starken Bot-Schutz hatte. Dies ist jedoch definitiv ein Thema, in das ich mich in Zukunft intensiv vertiefen möchte, damit es kein Projekt dieser Art jemals wieder behindert.

- **Nur ein Bild gesammelt statt 5:** Dies blieb aus Zeitmangel meist unerfüllt. Dass der Scraper mindestens ein Bild pro Eigenschaft gesammelt hat, war meiner Meinung nach genug, um zumindest zu zeigen, dass ich weiß, dass es fertig ist, und dass ich es nicht völlig vergessen habe, es zu implementieren.
- **Fehlende Filter - Sortierung funktioniert nicht:** Obwohl nicht genug Zeit blieb, um ein Filtersystem zu implementieren, dachte ich, ich könnte versuchen, zumindest ein Sortiersystem zu implementieren, um dies auszugleichen, aber selbst das funktioniert aus unbekannten Gründen nicht. Hätte ich meine Zeit besser verwaltet, wäre ich vielleicht in der Lage gewesen, den Fehler zu finden und es zum Laufen zu bringen, aber leider musste ich es zurücklassen, da es zu viel kostbare Zeit verschlang.
- **Kein benutzerdefinierte WordPress-Plugin - Lösung durch Plugin eines Drittanbieters:** Immer aus Zeitmangel war es mir nicht möglich, ein benutzerdefiniertes Plugin zu entwickeln, das den Scraper in eine WordPress-Site integriert. Ich habe versucht, zumindest eine Teillösung dafür zu finden, indem ich ein Plugin gefunden habe, das CSV-Dateien in Tabellen umwandeln kann, sodass die gesammelten Inhalte auf einer Seite angezeigt werden können.

Raum für Verbesserungen - Was funktioniert, aber besser gemacht werden könnte

Während meine Lösungen für die folgenden Punkte funktioniert haben, denke ich immer noch, dass es Möglichkeiten gegeben hätte, sie zu verbessern, wenn ich mehr Wissen gehabt hätte, bevor ich in dieses Projekt eingestiegen bin, und mehr Zeit gehabt hätte:

- **Bessere Lösung für mehrsprachige Unterstützung:** Die Lösung für Sprachpakete war eine, die ich meistens spontan gefunden habe, aber ich frage mich, ob es einen besseren Weg gegeben hätte, sie zu implementieren. Zum Beispiel eine Bibliothek, die Funktionen hat, die den Originaltext für einen übersetzen.
- **Besser mit Fehlern umgehen:** Damit beziehe ich mich auf zwei konkrete Fälle. Die erste ist die, über die ich bereits in den Test-Protokollen gesprochen habe. Manchmal lädt die Hauptseite von Booking etwas anders aus als sonst. Dies wirft die Anwendung ab und führt dazu, dass sie hängen bleibt, weil sie das Element, mit dem sie interagieren soll, nicht finden kann. Eine Lösung wäre gewesen, einige if-Bedingungen zu implementieren, bei denen der Scraper, als ob er ein Element nicht gefunden hätte, einen etwas anderen Weg nehmen würde, um weiterzuarbeiten. Der zweite betrifft die SQL-Abfragen im Code. Ich habe sie einfach so geschrieben, dass sie nur Tabellen mit Werten und sonst nichts erstellen konnten. Aus Effizienzgründen hätte ich eine Regel hinzufügen sollen, dass, wenn eine Tabelle mit genau demselben Namen in der Datenbank erkannt würde, diese Tabelle entweder gelöscht und durch die neue ersetzt würde. Entweder das, oder, noch professioneller, hätte eine ganze Methode hinzugefügt, die die beiden Tabellen angehängt und gleichzeitig sichergestellt hätte, dass am Ende keine Duplikate vorhanden sind.

Letztendlich hat mir dieses Projekt trotz des Gefühls der Enttäuschung, das die Misserfolge verursacht haben, einen Einblick gegeben, was ich verbessern muss. Nicht nur in Bezug auf diese Art von Projekten, sondern auch auf meine Herangehensweise an Projekte im Allgemeinen.

3. Eine effektivere Lösung - Octoparse

3.1. Was ist Octoparse?

Octoparse ist eine Cloud-basierte Lösung zur Extraktion von Webdaten, mit der Benutzer relevante Informationen aus verschiedenen Arten von Websites extrahieren können. Mit Octoparse können Nutzer aus einer Vielzahl von Branchen unstrukturierte Daten auslesen und in verschiedenen Formaten wie Excel, einfachem Text und HTML speichern.

Benutzer können ein Element auf einer Webseite anklicken, um die Art der zu extrahierenden Daten auszuwählen. Octoparse ermöglicht es Benutzern, mehrere Extraktionsaufgaben gleichzeitig auszuführen. Die Aufgaben können so geplant werden, dass sie in regelmäßigen Abständen ausgeführt werden, oder sie können in Echtzeit ausgeführt werden. Benutzer können auch Produktkommentare und -bewertungen sowie Social-Media-Kanäle auslesen, um Informationen über die Verbraucherstimmung zu sammeln.

Der Wizard-Modus von Octoparse bietet dem Benutzer eine schrittweise Anleitung zum Extrahieren von Daten, während der Advanced-Modus erweiterte Funktionen für kompliziertere Webseiten bietet.

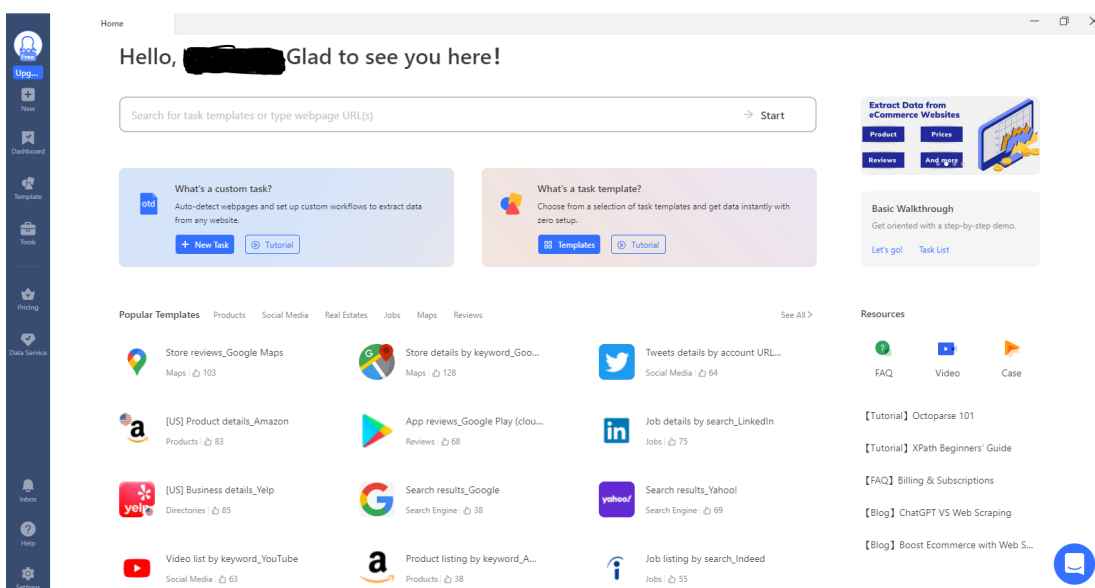
Octoparse bietet Dienstleistungen auf einer monatlichen Abonnementbasis an, die Support per E-Mail und über eine Online-Wissensdatenbank umfassen.

3.2 Verwendung von Octoparse

Zunächst einmal muss man natürlich die Anwendung herunterladen. Glücklicherweise bietet Octoparse eine kostenlose Version seines Scraping-Tools an, die mehr als genug sein sollte, um unser kleines Ziel zu erreichen: Daten über Hotels in Mali aus den Google-Suchergebnissen zu scrapen.

Nach der Installation fordert die Anwendung uns auf, ein Konto auf der Octoparse-Website zu erstellen, um das Tool nutzen zu können. Alles, was wir tun müssen, ist, dieser Aufforderung nachzukommen, und schon können wir loslegen.

Nachdem wir unser Konto erstellt und uns in der Anwendung angemeldet haben, erscheint ein Fenster, das wie folgt aussieht.



Jetzt müssen wir auf das vertikale Menü auf der linken Seite gehen und den Mauszeiger über die Schaltfläche "Neu" bewegen. Daraufhin öffnet sich ein weiteres kleines Fenster, das uns verschiedene Optionen bietet. Diejenige, die wir anklicken wollen, ist "Benutzerdefinierte Aufgabe".

Daraufhin wird das folgende Fenster angezeigt.

Task Group: My Group (dropdown), New Group (button)

URL Source: Enter manually (selected), Import from file, Import from task, Batch generate

URL Input: Enter or import one or more URLs here (line separated)

Please enter no more than 10K URLs.

Save (button)

Mit "Task Group" ist einfach eine Art von Ordner gemeint. Diese Option ist einfach da, um uns zu helfen, die Aufgaben besser zu organisieren.

Wie Sie sehen können, gibt es auch mehrere Optionen für den Start der Aufgabe, aber für unsere Zwecke beschränken wir uns auf die Option, die in der Abbildung oben bereits ausgewählt ist.

Aber welche URL soll man genau in den Textbereich eingeben? So könnten wir es machen.

Öffnen Sie zunächst Google Chrome und suchen Sie nach "Hotels in Mali". Als eines der Suchergebnisse sollten Sie einen Abschnitt erhalten, der wie folgt aussieht.

Hotels | Mali :

Sun, 4 Jun

Mon, 5 Jun

2

Top-rated

Budget options

Luxury stays

Free cancellation

For tonight



Radisson Collection Hotel, Bamako

CHF 234

4.4 ★★★★★ (362) · 4-star hotel

Pool · Spa



Au bord de l'eau

CHF 96

4.2 ★★★★★ (444)

Pool · Free parking



ONOMO Hotel Bamako

CHF 117

3.9 ★★★★★ (1.1K) · 3-star hotel

Pool · Free parking

View 108 hotels →



Sobald Sie diesen gefunden haben, klicken Sie auf "Hotels anzeigen".

Wir müssen nun den Link der Seite kopieren und in Octoparse eingeben. Alles, was jetzt noch zu tun ist, ist auf die Schaltfläche "Save" zu klicken.

Hier wird das Tool bereits ausgeführt und versucht, die gewünschte Seite zu scrapen. Allerdings besteht eine hohe Wahrscheinlichkeit, dass eine Art "Straßensperre" auftaucht. In unserem Fall ist es die Seite "Cookie Akzeptanz" von Google. Zum Glück für uns kann Octoparse uns helfen, dieses Hindernis zu überwinden.

Wie Sie leicht erkennen können, gibt es ein kleines Fenster auf der rechten Seite. Eine der anklickbaren Optionen ist ein Etikett mit dem Text "Not the right webpage?". Wir wollen auf diese Option klicken.

Google

Before you continue to Google

We use cookies and data to

☒ Deliver and maintain Google services

☒ Track outages and protect against spam, fraud and abuse

☒ Measure audience engagement and site statistics to understand how our services are used and improve the quality of those services

If you choose to 'Accept all', we will also use cookies and data to

☒ Develop and improve new services

☒ Deliver and measure the effectiveness of ads

☒ Show personalised content, depending on your settings

Tips

List data detected. Preview your data at the bottom and choose how you'd like to extract the data.

☒ Extract the data in the list

☒ Add a page scroll ⓘ

Edit

Create workflow

Cancel

Switch auto-detect results (1/3)

Not the right webpage?

Das kleine Fenster zeigt uns dann verschiedene Optionen bzw. Lösungen an, um dieses Hindernis zu überwinden. In unserem Fall wollen wir auf die Option "Close a pop-up" klicken.

Hier werden wir in der Lage sein, direkt mit der Vorschau der Webseite zu interagieren. Unser nächster Schritt besteht also darin, nach unten zu scrollen, bis wir die Schaltfläche "Alle akzeptieren" finden, und darauf zu klicken. Sie werden sehen, wie das kleine Fenster reagiert und Ihnen den CSS-Selektor des Elements anzeigt, mit dem Sie interagieren möchten. Klicken Sie dann auf "Confirm".

- ✧ Develop and improve new services
- 📊 Deliver and measure the effectiveness of ads
- ✦ Show personalised content, depending on your settings
- 📄 Show personalised ads, depending on your settings

If you choose to 'Reject all', we will not use cookies for these additional purposes.

Non-personalised content is influenced by things like the content that you're currently viewing, activity in your active Search session, and your location. Non-personalised ads are influenced by the content that you're currently viewing and your general location. Personalised content and ads can also include more relevant results, recommendations and tailored ads based on past activity from this browser, like previous Google searches. We also use cookies and data to tailor the experience to be age-appropriate, if relevant.

Select 'More options' to see additional information, including details about managing your privacy settings. You can also visit g.co/privacytools at any time.

[Reject all](#) [Accept all](#)

+

Tips

⬆

Close a pop-up window

Tag: DIV (Division)

Text: Null

« /IV > DIV > BUTTON > DIV »

[Cancel](#) [Confirm](#)



Travel

Explore

Things to do

Flights

Hotels

hotels in mali

Sun, 4 Jun < >

Mon, 5 Jun < >

2 ▾

All filters

★ 4+ rating

⬤ 4- or 5-star

Pool

Under CHF 75

Price ▾



When to visit



What you'll pay

Mali · 108 results

About these results ⓘ



Hôtel l'Amitié Bamako

CHF 134

Sponsored · From Booking.com

4.1 ★ (2,162)

5-star hotel

Free breakfast

Free Wi-Fi

Free parking

Outdoor pool

Air conditioning

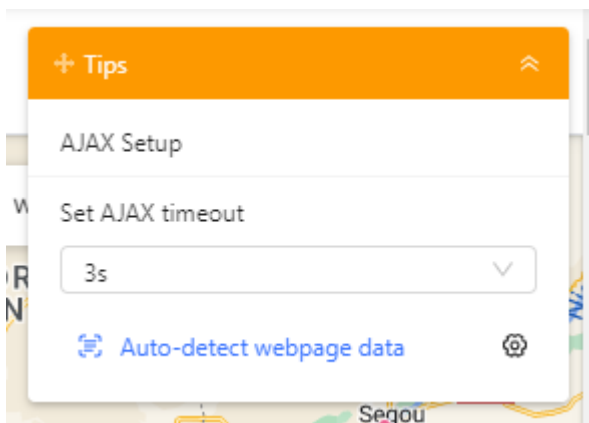
Pet-friendly

Fitness centre

Spa

[View offer](#)

Nun wollen wir noch einmal einen Blick auf das kleine Fenster zu unserer Rechten werfen. Sie können dort sehen, dass eine der Optionen "Auto-detect webpage data" ist. Dies ist sehr nützlich, da das Tool dann automatisch damit beginnt, die Informationen der Seite zu ermitteln, die wir sammeln möchten.

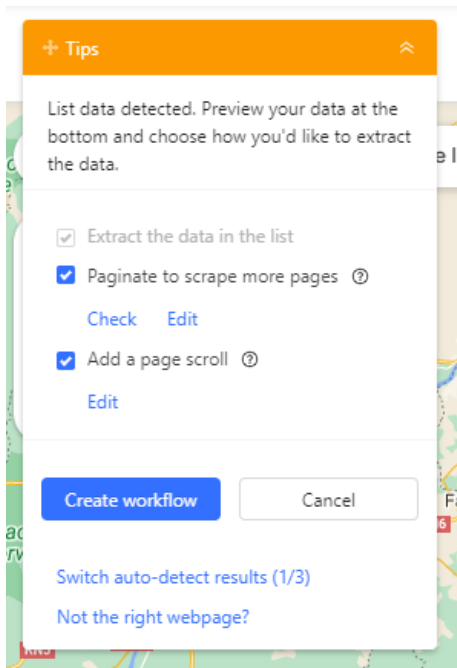


Sobald das Tool mit seiner Analyse fertig ist, finden wir in der unteren Hälfte des Anwendungsfensters eine Vorschau aller relevanten Daten, die es gefunden hat. Während die Namen der Spalten größtenteils Kauderwelsch sind (es handelt sich um die IDs der Elemente, aus denen die Daten extrahiert wurden), macht ein Blick auf die Zeilen darunter ziemlich klar, welche Art von Daten jede Spalte enthält.

Data Preview 22 data fields pre-selected, total 20 lines of data. Preview the first [10] line(s). Preview more data											
Data Fields	No.	Title	pvooxe_URL	oxgzuc_URL	spnmc_URL	kixhkb	kfi5wf	jdzyld	lxjaod	uqzrme	
Page1	1	Hôtel l'Amitié Bamako	https://www.google....	https://www.google....	https://www.google....	CHF 134	4.1	(2,162)	Amenities for Hôtel l...	5-star hotel	Free bre
	2	Laico Hôtel El Farouk...	https://www.google....	https://www.google....	https://www.google....	CHF 140	3.9	(534)	Amenities for Laico ...	5-star hotel	Free bre
	3	Radisson Collection ...	https://www.google....	https://www.google....	https://www.google....	CHF 233	4.4	(362)	Amenities for Radiss...	4-star hotel	Free bre
	4	Au bord de l'eau	https://www.google....	https://www.google....	https://www.google....	CHF 96	4.2	(444)	Amenities for Au bor...		Breakfas
	5	ONOMO Hotel Bam...	https://www.google....	https://www.google....	https://www.google....	CHF 117	3.9	(1,060)	Amenities for ONO...	3-star hotel	Free bre
	6	ATAI HOTEL RAM	https://www.google....	https://www.google....	https://www.google....	CHF 165	4.2	(1,140)	Amenities for ATAI A	5-star hotel	Breakfas

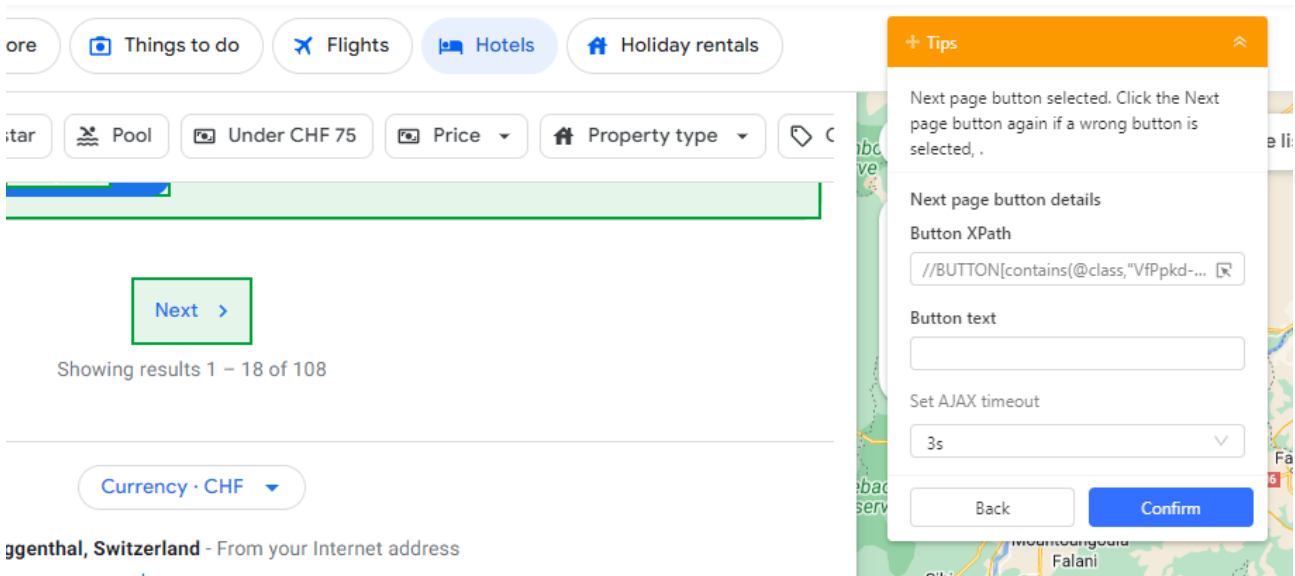
"Titel" enthält die Namen aller Hotels, "kixhkb" alle Durchschnittspreise, "kfi5wf" die Google-Bewertungen und so weiter.

Wenn wir uns jedoch alle Daten in der Vorschau ansehen, werden wir schnell feststellen, dass sie nur die Daten der ersten Ergebnissseite enthält. Was tun wir also, wenn wir mehrere Ergebnissseiten haben, von denen wir Daten sammeln wollen? Nun, alles was wir tun müssen, ist, einen Blick auf das kleine Fenster auf der rechten Seite zu werfen.

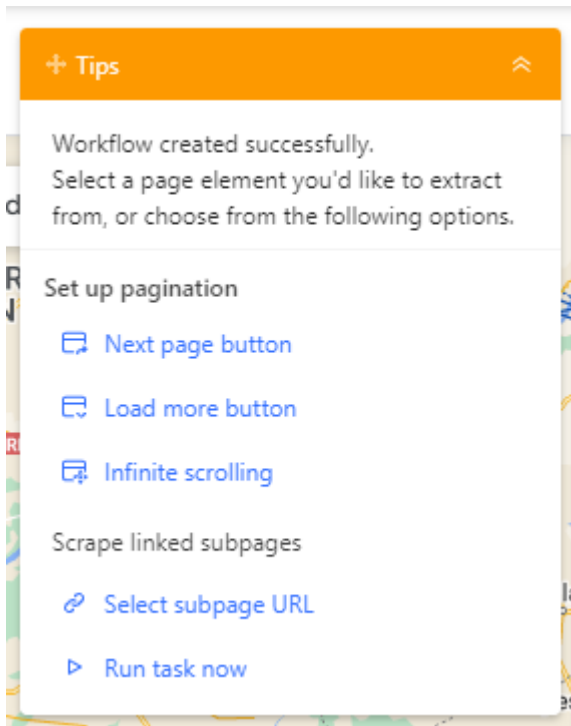


Wir werden sehen, dass zwei Kästchen bereits für uns angekreuzt worden sind. Um sicherzugehen, sollten Sie auf die Schaltfläche "Bearbeiten" unter dem ersten Kontrollkästchen klicken, um sicherzustellen, dass das Tool die richtige Schaltfläche ausgewählt hat.

Dazu scrollen Sie einfach nach unten und klicken auf die Schaltfläche "Weiter" auf der Vorschauseite, sobald diese interaktiv wird, und sehen, ob sich in dem kleinen Fenster der XPath der Schaltfläche ändert. In unserem Fall scheint das Tool die richtige Schaltfläche gefunden zu haben.



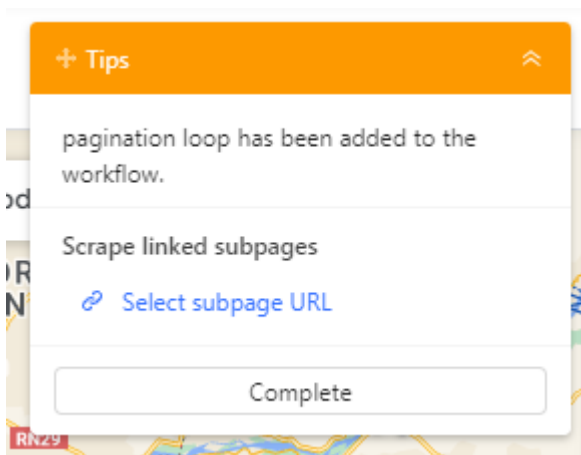
Jetzt müssen Sie nur noch auf die Schaltfläche "Create Workflow" klicken.



Jetzt ist der Workflow erstellt, aber aus irgendeinem Grund verlangt das Tool, dass wir erneut eine Paginierung einfügen. Dies kann durch Anklicken der Option "Next page button" in dem kleinen Fenster erfolgen.

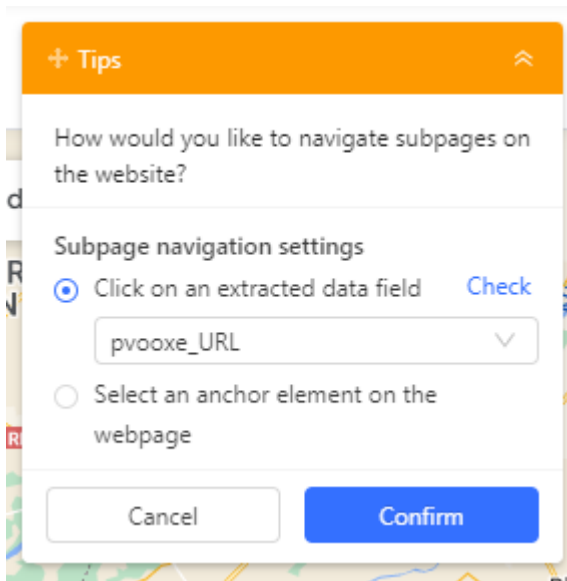
Hier müssen Sie nur die bereits oben erwähnten Schritte zur Auswahl der Schaltfläche wiederholen und auf "Confirm" klicken.

Das kleine Fenster bietet uns dann eine weitere Option: "Select subpage URL".

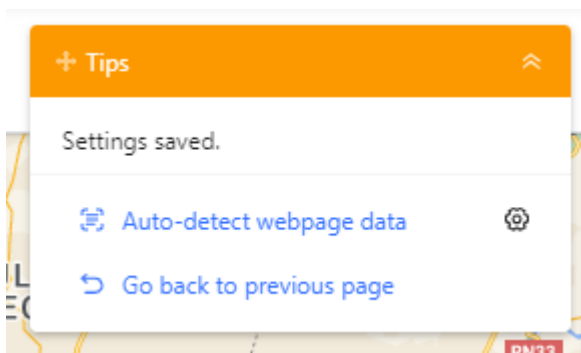


Wie der Text in der Beschriftung andeutet, ermöglicht uns dies, Daten von Unterseiten zu extrahieren, die mit unserer Haupt-URL verbunden sind.

Wenn wir uns für diese Option entscheiden, werden wir in dem kleinen Fenster gefragt, wie wir zu diesen Unterseiten navigieren möchten. In unserem Fall ist die vom Tool vorgeschlagene Lösung bereits die optimale: Wir möchten lediglich, dass das Tool auf den Link des Hotelangebots verweist.



Sobald wir auf „Confirm“ geklickt haben, gelangt man über die Vorschauseite auf die Angebotsseite eines der Suchergebnisse und das kleine Fenster gibt uns dann noch einmal die Möglichkeit, relevante Daten automatisch zu erkennen.



Was wir nun von den Angebotsseiten benötigen, ist lediglich die Adresse, die Telefonnummer und der Link zu ihrer Website. Sie werden jedoch feststellen, dass die automatische Erkennung diese Daten nicht erkennt. Was ich persönlich versucht habe, ist, die zu extrahierenden Daten manuell auszuwählen, und während die Datenvorschau auf den ersten Blick die von mir gewünschten Daten anzeigte, stürzte das Tool kurz darauf ab (die Ursache hierfür ist unbekannt). Daher wird die Unterseitenoption für dieses Beispiel weggelassen.

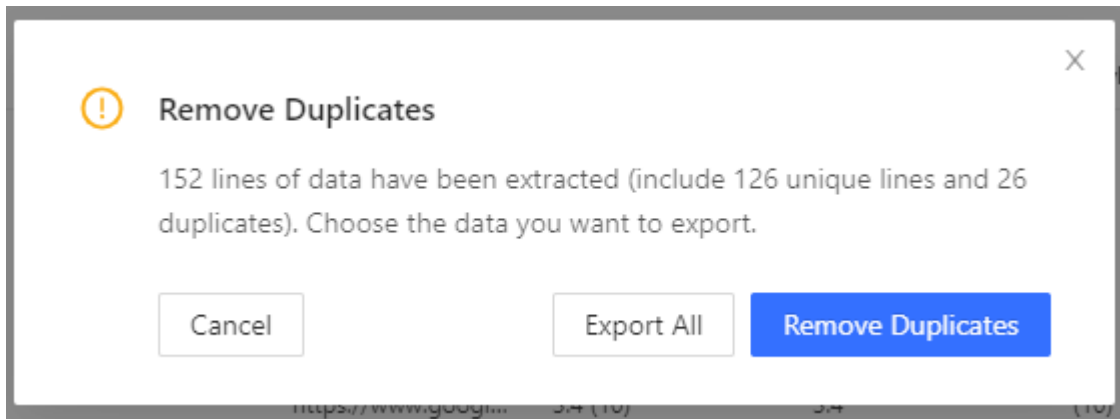
Jetzt müssen Sie es nur noch ausprobieren und ausführen. Sie werden gefragt, ob Sie es auf Ihrem Gerät oder in einer Cloud ausführen möchten. Wählen Sie hier einfach die Option aus, die Sie für die beste halten. Stellen Sie sicher, dass Sie den „Standard“-Modus auswählen, da der andere Modus nur für Premium-Benutzer gedacht ist.

Hier erscheint ein neues Fenster und wir können einfach zusehen, wie unser neu entwickelter Schaber seine Arbeit erledigt.

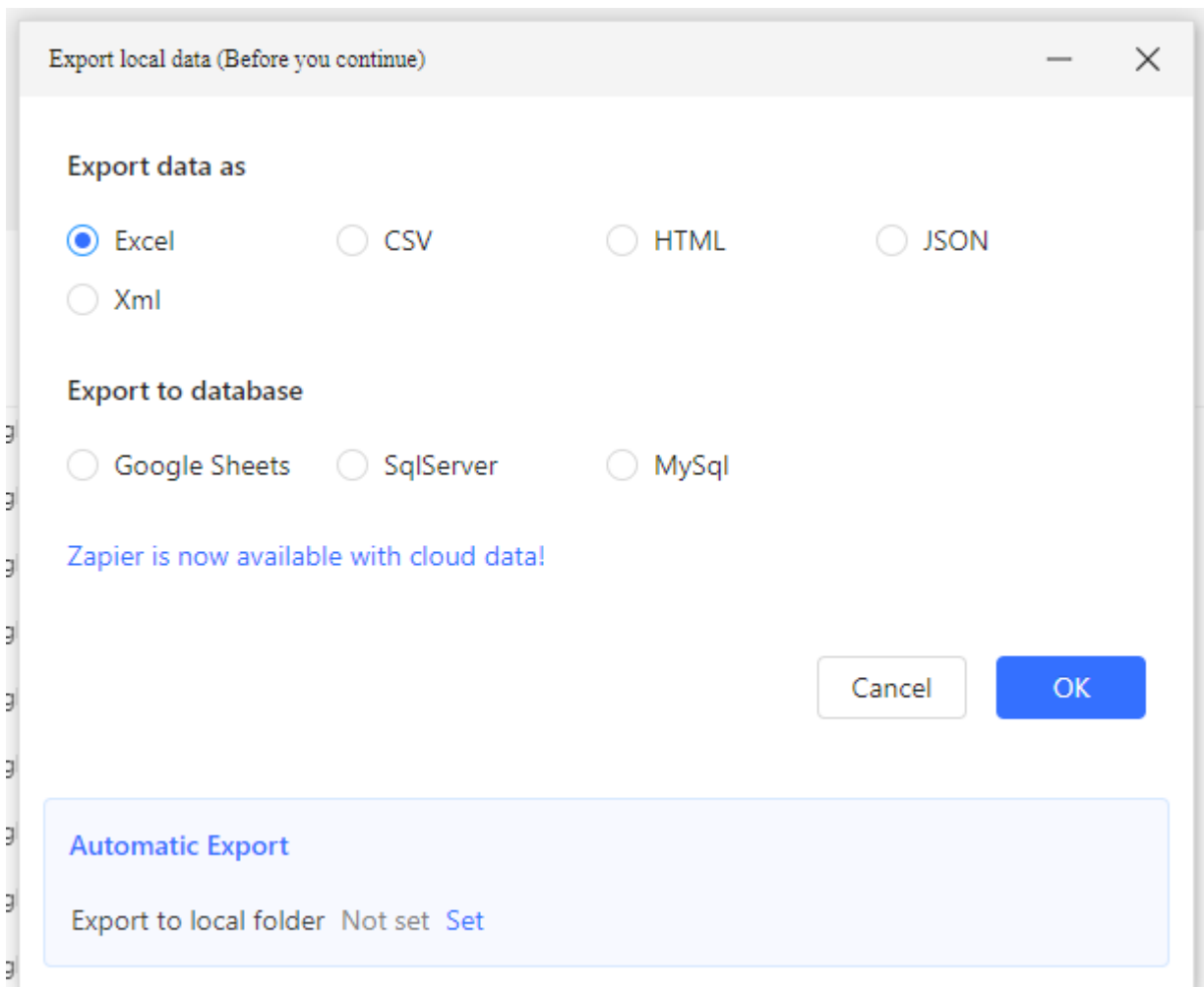
Sobald dies erledigt ist, öffnet sich ein Fenster, in dem Sie gefragt werden, ob Sie die Daten exportieren möchten oder nicht.

Eine nette Funktion der Octoparse-Scraper ist, dass sie automatisch erkennen können, ob in den extrahierten Daten Duplikate vorhanden sind. Wenn Duplikate erkannt werden, fragt Octoparse Sie vor dem

Exportieren, ob Sie die Duplikate entfernen möchten oder nicht (es wird auch angezeigt, wie viele der gesammelten Datenzeilen Duplikate sind).



Unabhängig davon, welche Option wir gewählt haben, werden wir zu einem weiteren Fenster weitergeleitet, in dem wir auswählen können, in welchem Format wir die gerade erfassten Daten speichern möchten. Wie Sie im Bild unten sehen können, gibt es eine große Auswahl.



Sie müssen die Datei nur noch entsprechend benennen und an einem beliebigen Ort speichern.

3.3 Weitere interessante Funktionen

Templates

Möglicherweise haben Sie im Bild ganz am Anfang dieses Octoparse-spezifischen Abschnitts bemerkt, dass Octoparse viele Vorlagen bietet. Bei Vorlagen handelt es sich lediglich um vorgefertigte Scraper, mit denen bestimmte Daten aus bestimmten Quellen extrahiert werden können. Es ist jedoch unklar, ob diese Vorlagen vom Octoparse-Team oder seiner Community erstellt wurden oder vielleicht sogar eine Mischung aus beidem.

Unabhängig davon bedeutet dies, dass Sie je nachdem, welche Daten Sie benötigen, jetzt möglicherweise sogar den gesamten Prozess der Erstellung einer benutzerdefinierten Aufgabe durchlaufen müssen, da möglicherweise bereits eine Vorlage für das, was Sie tun möchten, vorhanden ist.

Sie müssen sich jedoch darüber im Klaren sein, dass die Nutzung, genau wie einige andere Octoparse-Funktionen, nicht alle kostenlos ist. Während Sie einige von ihnen nach Herzenslust kostenlos nutzen können, müssen Sie bei anderen eine Nutzungsgebühr zahlen.

Senden von Anfragen

Wenn es Ihnen zu schwer fällt, eine eigene Aufgabe zu erstellen, und es keine Vorlage für Ihre Aufgabe gibt, können Sie jederzeit eine Anfrage einreichen.

Sie müssen lediglich ein Formular ausfüllen, in dem Sie lediglich die Ziel-URL (in unserem Fall die Ergebnisseite für die Hotels in Mali) eingeben und angeben, was daraus extrahiert werden soll. Allerdings müssen Sie hier 2 wichtige Dinge beachten:

1. Es ist nicht bekannt, ob und wie viel der Erwerb dieser benutzerdefinierten Vorlage kosten wird
2. Nur nach ihren Worten werden „sehr gefragte Vorlagen“ hinzugefügt. Sollten Sie also keine Antwort erhalten und Ihr Fall dringend sein, können Sie stattdessen einen Custom Crawler Service über Chatbot anfordern. Mir ist jedoch wiederum nicht bekannt, ob dieser Dienst kostenlos oder kostenpflichtig ist oder nicht.

4. Glossar

- **robots.txt:** Ein Standard, der von Websites verwendet wird, um besuchenden Web Crawlern und anderen Web Robotern anzuzeigen, welche Teile der Website sie besuchen dürfen.
- **Traceback:** Es handelt sich um einen Bericht, der die Funktionsaufrufe in einem Code an einem bestimmten Punkt enthält.
- **Sitemap:** Es ist eine Datei, die Informationen über die Seiten, Videos und andere Dateien auf einer Website und die Beziehungen zwischen ihnen enthält.
- **Continuous Integration (CI):** Dabei handelt es sich um die automatisierte Integration von Codeänderungen mehrerer Beteiligter in ein einziges Softwareprojekt.
- **Parsen:** Dabei wird eine Befehlsfolge - in der Regel ein Programm - in leichter zu verarbeitende Komponenten zerlegt, die auf korrekte Syntax analysiert und dann mit Tags versehen werden, die jede Komponente definieren.
- **Streaming:** Ein Verfahren zur Übertragung oder zum Empfang von Daten (insbesondere von Video- und Audiomaterial) über ein Computernetz als gleichmäßiger, kontinuierlicher Fluss, der es ermöglicht, mit der Wiedergabe zu beginnen, während der Rest der Daten noch empfangen wird.
- **Socket:** Es handelt sich um einen Kommunikationsverbindung-Punkt (Endpunkt), den man in einem Netz benennen und adressieren kann.
- **Pythonische Idiom:** Es ist das, was man schreibt, wenn man nur mit dem richtigen Weg kämpft, um sein Problem zu lösen; nicht mit der Programmiersprache oder irgendeinem seltsamen Bibliotheksfehler oder einem unangenehmen Datenabruf-Problem oder etwas anderem, das mit dem eigentlichen Problem nichts zu tun hat.
- **De-Facto-Standard:** Ein De-Facto-Standard ist ein Brauch oder eine Konvention, der/die durch öffentliche Akzeptanz oder Marktkräfte eine beherrschende Stellung erlangt hat.
- **Data-Wrangling:** Es handelt sich um die Verarbeitung von Daten in verschiedenen Formaten und Analysen und deren Kombination mit anderen Datensätzen, um aussagekräftige Erkenntnisse zu gewinnen.

Quellenverzeichnis

- What is Web Scraping and What is it used for?, Martin Perez, <https://www.parsehub.com/blog/what-is-web-scraping/>
- Is Web Scraping Legal?: Can You Get in Trouble for Web Scraping Publicly Available Data?, Martin Perez, <https://www.parsehub.com/blog/web-scraping-legal/>
- BeautifulSoup: Build a Web Scraper With Python, Martin Breuss, <https://realpython.com/beautiful-soup-web-scraper-python/>
- Web crawling vs web scraping, Colm Kenny, <https://www.zyte.com/learn/difference-between-web-scraping-and-web-crawling/#:~:text=The%20short%20answer,to%20combine%20crawling%20and%20scraping>
- The Best Programming Languages for Web Scraping, Valentina Skakun, <https://scrape-it.cloud/blog/web-scraping-languages>
- What is BeautifulSoup?, Educative Answers Team, <https://www.educative.io/answers/what-is-beautiful-soup>
- Selenium with Python Tutorial: Getting started with Test Automation, Shaumik Daityari und Pradeep Krishnakumar, <https://www.browserstack.com/guide/python-selenium-to-run-web-automation-test#:~:text=Selenium%20is%20an%20open%20source,can%20choose%20the%20script%20accordingly>
- Python's Requests Library (Guide), Alex Ronquillo, <https://realpython.com/python-requests/>
- A Beginner's Guide to the Python time Module, Alex Ronquillo, <https://realpython.com/python-time-module/#:~:text=The%20Python%20time%20module%20provides,the%20efficiency%20of%20your%20code>
- What Are Pandas in Python? Everything You Need to Know, ActiveState Team, <https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/>
- PyMySQL — Connecting Python and SQL for Data Science, Kenneth Leung, <https://towardsdatascience.com/pymysql-connecting-python-and-sql-for-data-science-91e7582d21d7>
- What is Tkinter used for and how to install it?, ActiveState Team, <https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/>
- How to Scrape Booking.com (2023 Update), ScrapFly Team, <https://scrapfly.io/blog/how-to-scrape-bookingcom/>