# WAS neu Ist/geändert wurde

**Anhang Liste hinzugefügt**

**quelle Liste hinzugefügt**

**1.1** wurde viel gekürzt

**1.2** ersetzt: Es wird einen Prozess orientiert Dokumentation erstellt. Ich werde nach der Wasserfallmodel die einzelnen Schritte dokumentieren

**2.1.7** hinzugefügt: das ist das Gesamtproject ich mach nur das importieren der neuen Produkte in den online Shop.

**2.1.9** Projekt Risiken: gelöscht

**2.2.2** Analyse des ist zustand: gelöscht

**2.2.3** Erstellung des soll Konzept: gelöscht

**2.3** in zeit Planung 3.3 verschoben

**3.4** Quellen hinzugefügt (als Anhang und Quelle):

* Programmierer Stunden Lohn
* it-chef Stunden Lohn
* 6 Euro pauschal

**4.2** Eigentliche umsetzung

Numerierung hinzugefuegt

Selenium beschreibung satzt hinzugefuegt

Anhang 18: Aktivität Diagramm geupdatet

**4.5** Fehler Behebung und Anpassung gelöscht

**5.2** hinzugefügt: Es wurde eine Schulung durchgeführt in der Funktionsweise das Project vorgestellt wurde

**5.3** zeitlich: Hinzugefügt es hat sich verschoben aber die Zeit wurde nachgeholt

**Pages Layout:**

allgemein Black Mode zu light Mode geändert

Titel oben + logo + Linien unten Pages + namen +Linien kommt noch...

Quelle habe ich auch als bild

|  |  |  |
| --- | --- | --- |
| **Prüfungsphase:** Sommer 2024 |  | **Name, Vorname**: Gregory Maleterre Ausbildungsberuf: Fachinformatiker für Anwendungsentwicklung  Prüflingsnummer: ???? Id.Nr.: 1586681-50 |

|  |
| --- |
| Effizientes E-Commerce: Erstellen, Erweitern und Realisieren einer Optimierten Produktverwaltung in WooCommerce mit Python, Excel und CSV |

|  |  |  |
| --- | --- | --- |
| **Projektgeber**: Pixelding Provinzstraße 57  13409 Berlin |  | **Ausbildungsbetrieb**: BFW Berlin-Brandenburg e. V. Kastanienallee 25 16567 Mühlenbeck  **Ansprechpartner**: Herr Poguntke Telefon: +49 33056 86316  Mail: martin.poguntke@bfw-berlin-brandenburg.de |

|  |  |  |
| --- | --- | --- |
| **Eidesstattliche Erklärung:**  Hiermit erkläre ich, Gregory Maleterre, die hier vorliegende Arbeit (Dokumentation) selbstständig und nur unter Zuhilfenahme der aufgeführten Quellen angefertigt zu haben. Der aufgeführte zeitliche Rahmen wurde eingehalten. | | |
| Ort/Datum |  | Unterschrift |
|  |  |  |

IHK Bewilligter Antrag

Zeitmitschreibung der Projektarbeit

Prüfungsteilnehmer

Name: Gregory

Vorname: Maleterre

Login: (Login des Online Portals)

|  |  |  |
| --- | --- | --- |
| Datum | Tätigkeit | Zeit in Stunden |
|  | Planungsphase |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Summe Planungsphase |  |
|  |  |  |
|  | Durchführungsphase |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Summe Durchführungsphase |  |
|  |  |  |
|  | Qualitätsmanagement |  |
|  |  |  |
|  |  |  |
|  | Summe Qualitätsmanagement |  |
|  |  |  |
|  | Übergabe-, Auswertungsphase |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Abbildungsverzeichnis

Anhang 1: Aktivität Diagramm

Anhang 2: Programmierer Stundenlohn

Anhang 3: Head of IT-Stundenlohn

Anhang 4: Home­office-Pauschale

Anhang 5: Aktivität Diagramm

Anhang 6: URL-Programmierung

Anhang 7: Log Output

Anhang 8: Erstellen von Ordnern für Produkte, PDFs und Bilder Programmierung

Anhang 9: Folders Anzahl

Anhang 10: Download Programmierung

Anhang 11: JSON-Output

Anhang 12: Protokoll

Anhang 13: JSON

Anhang 14: Protokoll

Anhang 15: JSON Output

Anhang 16: Protokoll

Anhang 17: Produkt CSV Import Schema

Anhang 18: Aktivität Diagramm

Anhang 19: CSV Output

Anhang 20: WooCommerce Import GUI 1

Anhang 21: WooCommerce Import GUI 2

Anhang 22: WooCommerce Import GUI 3

Anhang 23: XLSX zu CSV

Anhang 24: WooCommerce Fehler Meldung

Abkürzungen

CSV - Comma-separated Values

FTP - File Transfer Protocol

GUI - Graphical User Interface

HTML - Hypertext Markup Language

JSON - JavaScript Object Notation

PDF - Portable Document Format

SEO - Search Engine Optimization

SKU - Stock Keeping Unit

URL - Uniform Ressource Locator

UTF-8 - 8-bit Unicode Transformation Format

XAMPP - Cross-Plattform (X), Apache (A), MySQL (M), PHP (P), and Perl (P)

XLSX - Microsoft Excel Open XML Spreadsheet

Quellenverzeichnis

<https://www.stepstone.de/gehalt/Programmierer-in.html>

<https://www.stepstone.de/gehalt/Head-of-IT.html>

<https://www.test.de/Homeoffice-Pauschale-So-rechnen-Sie-mit-dem-Finanzamt-ab-5788023-0/>

https://github.com/woocommerce/woocommerce/wiki/Product-CSV-Import-Schema#csv-columns-and-formatting

Deckblatt

**Vorwort/ Ihre Frage**

Gehen Sie in Ihrer Dokumentation bitte genauer auf den Punkt "Datenextraktion von den Webseiten der Lieferanten" ein. Warum wurde sich für diesen Weg entschieden und warum nicht für eine definierter Kommunikation über z.B. Schnittstellen?

Es gibt einen klaren Unterschied zwischen den Webseiten der Hersteller und denen der Lieferanten. Während es bei den Herstellern oft die Möglichkeit einer definierten Kommunikation über Schnittstellen gibt, ist dies bei den Lieferanten nicht der Fall. Diejenigen, die diese Option anbieten, stellen in der Regel nur begrenzte Informationen zur Verfügung, die eher für ein Fachpublikum gedacht sind.

Das SEO-Team ist möglicherweise nicht mit dem Spezialgebiet der Putzmittel vertraut und benötigt daher umfassendere Informationen, die es auch für ein breiteres Publikum verständlich aufbereiten kann. In Anbetracht dieser Anforderungen erwies sich die Extraktion von Informationen und Daten mit Python als die bessere Lösung.

Die Verwendung von Python ermöglichte eine flexible und anpassungsfähige Extraktion direkt von den Webseiten der Lieferanten. Auf diese Weise konnten wir sicherstellen, dass das SEO-Team über die notwendigen Informationen verfügte, um die Produktbeschreibungen optimal zu gestalten und gleichzeitig die Anforderungen des Kunden zu erfüllen. Diese Herangehensweise bot uns die Möglichkeit, eine umfassende Datengrundlage zu schaffen, die sowohl für interne als auch externe Zwecke geeignet war.

# Projektdokumentationshinweise Verzeichnisse

## Verwendungsvermerk

Es wurde in Absprache mit dem Projektgeber, Pixelding, und dem Kunden, Putzmittelshop24, eine Vereinbarung getroffen und genehmigt, dass ich ihre Daten für die Dokumentation meines Projekts verwenden darf.

## Dokumentation zur Projektarbeit

Das Projekt zielt darauf ab, einen reibungslosen Prozess für die Integration neuer Produkte in einen bestehenden Online-Shop auf Basis von WordPress WooCommerce zu schaffen. Eine prozessorientierte Dokumentation wird erstellt. Ich werde die einzelnen Schritte nach dem Wasserfallmodell dokumentieren.

# Planungsphase

## Ausgangssituation

### Projektumfeld

Pixelding ist ein verlässlicher Partner in Sachen Web- und Grafikdesign aus der Medien-Hauptstadt Berlin. Mit einem Team von 2–10 Beschäftigten vereinen sie Expertise, Leidenschaft und effektive Kommunikation, um die Visionen ihrer Kunden zum Leben zu erwecken.Ihre Kernkompetenzen umfassen Webdesign, Suchmaschinenoptimierung (SEO), die Entwicklung virtueller Touren sowie die Gestaltung und Umsetzung von Online-Shops und Buchungssystemen. Zusätzlich bieten sie professionelle Werbekampagnen und unterstützen bei der Gestaltung von Printmedien. Weitere Informationen über Pixelding finden Sie auf unserer Website unter: <https://www.pixelding.de>

Das Projekt umfasst die Erweiterung eines Online-Shops und wird in verschiedenen Abteilungen verarbeitet. Alexander Wolobuew, der Eigentümer von Pixelding, ist der verantwortliche Leiter des Projekts, welches größtenteils von der Abteilung für Programmierung durchgeführt wird. Die Anpassung von Bildern liegt in der Zuständigkeit des Grafikteams, während das SEO-Team für die Textanpassungen verantwortlich ist.

Putzmittelshop24 ist ein etablierter Online-Shop für Reinigungsprodukte mit Sitz in Berlin. Der Shop wird von Thomas Pries geleitet und bietet eine breite Palette hochwertiger Reinigungsmittel für private Haushalte und gewerbliche Kunden.

Weitere Informationen über Putzmittelshop24 finden Sie auf unserer Website unter: https://putzmittelshop24.de/

### IST-Zustand

Der gegenwärtige Zustand des Online-Shops basiert auf einem manuellen Prozess zur Erstellung der Produktliste. Dabei werden Namen und Informationen von den Websites der Großhändler kopiert und eingefügt. Sowohl der Download von PDFs als auch von Bildern erfolgt manuell und beansprucht erheblich viel Zeit. Die manuelle Benennung und Strukturierung der Ordner für die Speicherung von Bildern und PDFs führt zu potenziellen Tippfehlern und erschwert die Auffindbarkeit des richtigen Pfads erheblich.

Zusätzlich ist der Kunde gezwungen, seine Produkte einzeln in die WooCommerce-Plattform über die langsame grafische Benutzeroberfläche von WordPress einzugeben. Dieser mühsame Prozess erfordert nicht nur das manuelle Verknüpfen von Bildern, sondern auch das Kopieren und Einfügen von Preisen und anderen Informationen. Die resultierende Arbeitsweise ist äußerst zeitaufwendig und ineffizient.

Die manuelle Natur dieser Abläufe birgt das Risiko von menschlichen Fehlern, was zu ungenauen Produktinformationen und Verzögerungen bei der Aktualisierung der Produktliste führen kann. Insgesamt ist der aktuelle Prozess nicht nur zeitintensiv, sondern auch anfällig für Fehler und stellt eine erhebliche Herausforderung für eine effiziente Verwaltung und Aktualisierung des Produktbestands dar.

### SOLL-Zustand

Unser Kunde benötigt sämtliche tausende neue Produkte mit Fotos, PDF-Downloads und Preisen und Beschreibungen auf seiner Webseite. Darüber hinaus benötigt er eine unkomplizierte Infrastruktur zur Preisgestaltung sowie die Möglichkeit, die Texte aller Produkte für SEO-Zwecke zu aktualisieren.

Die optimierte Lösung automatisiert den Prozess der Produktintegration in den Online-Shop erheblich. Ein automatischer Algorithmus generiert eine strukturierte Liste aller neuen Produkte von den Lieferantenwebseiten. Diese Liste bildet die Grundlage für sämtliche Aktualisierungen im System.

Fotos und PDFs werden automatisch heruntergeladen und in einem systematisch organisierten Ordner abgelegt. Dadurch wird eine fehleranfällige manuelle Handhabung vermieden, und die Auffindbarkeit der Dateien wird optimiert.

Zwei separate Excel-Dateien werden automatisch erstellt: Eine für das SEO-Team zur Aktualisierung von Produktbeschreibungen und eine für das Grafikdesign-Team zur optionalen Aktualisierung von Bildern. Die Excel-Dateien werden im Anschluss automatisch in das erforderliche CSV-Format für WooCommerce konvertiert.

Dies schließt auch die spezielle Excel-Datei des Kunden ein, um Preise, Versandkosten und Lagerbestände unkompliziert zu verwalten. Diese automatisierte Lösung gewährleistet nicht nur eine erhebliche Zeitersparnis, sondern minimiert auch das Risiko von Fehlern.

Der gesamte Prozess ist effizienter und benutzerfreundlicher, sodass der Kunde mühelos neue Produkte in seinem Online-Shop präsentieren kann.

### Projektziele

Aufgrund der fehlenden Option einer definierten Kommunikation über beispielsweise Schnittstellen, beinhaltet der Prozess die Verwendung von Python zur Extraktion von Daten von den Webseiten der Lieferanten. Die gesammelten Daten werden in einer JSON-Datei gespeichert. Die Bilder und PDFs jedes Produkts werden in einem Ordner abgelegt, dessen Name sich aus dem Produktnamen ableitet, und der Pfad dazu wird in der JSON-Datei referenziert. JSON-File und Folder werden nicht nur auf Google Drive gesichert, sondern auch auf zwei separaten Festplatten für zusätzliche Redundanz abgelegt. Anschließend erfolgt die Umwandlung der JSON-Daten in das notwendige WooCommerce CSV-Format mittels Python. Die Dateien werden daraufhin auf den Test-WooCommerce-Shop im lokalen Host hochgeladen.

Das Grafiker Team erhält eine autogenerierte XLSX-Liste der Produktfotos zur eingehenden Prüfung. Gleichzeitig wird dem SEO-Team eine XLSX-Liste bereitgestellt, um Texte umzuschreiben und zu ergänzen. Parallel dazu wird eine Excel-Tabelle für den Kunden erstellt, um die Verwaltung seiner Preise und Lagerbestände zu erleichtern.

Schließlich wird ein System implementiert, das die Excel-Dateien wiederum in das WooCommerce CSV-Format konvertiert. Die CSV-Dateien werden lokal auf WordPress hochgeladen, um umfassende Tests durchzuführen. Bei erfolgreicher Durchführung werden die CSV-Dateien vom lokalen Shop exportiert und in den Online-Shop des Kunden übertragen. Nach erneuten Tests zur Sicherstellung eines reibungslosen Ablaufs werden sämtliche Projektdetails, Probleme und Fortschritte in Asana und Google Drive dokumentiert.

### Personellen Schnittstellen

In diesem Projekt habe ich unter der Verantwortung von Alexander Wolobuew gearbeitet, parallel dazu auch mit den Grafikern und SEO-Spezialisten.

### Technische Schnittstellen

* **Python**: Eine Programmiersprache, die häufig für die Automatisierung, Datenanalyse und Webentwicklung verwendet wird.
* **WordPress**: Eine Open-Source-Plattform zur Erstellung und Verwaltung von Websites und Blogs.
* **WooCommerce**: Ein E-Commerce-Plugin für WordPress, das es ermöglicht, einen Online-Shop zu erstellen und zu betreiben.
* **IONOS Server:** Ein Open Network Operating System, das zur Steuerung und Verwaltung von Netzwerken verwendet wird.
* **Windows:** Ein Betriebssystem von Microsoft, das auf PCs weit verbreitet ist.
* **XAMPP:** Eine kostenlose und Open-Source-Software, die einen lokalen Server für die Entwicklung von Webanwendungen bereitstellt.
* **GitHub:** Eine Plattform für die Versionsverwaltung und Zusammenarbeit bei der Softwareentwicklung.
* **Visual Studio Code:** Ein plattformübergreifender Code-Editor, der von Entwicklern für die Softwareentwicklung verwendet wird.
* **Google Drive**: Ein cloudbasierter Speicherdienst von Google, der die Speicherung und gemeinsame Nutzung von Dateien ermöglicht.
* **Asana**: Eine webbasierte Anwendung zur Teamkollaboration und Aufgabenverwaltung.
* **Slack**: Ein Messaging-Tool für Teams, das die Kommunikation und Zusammenarbeit unterstützt.
* **FileZilla**: Eine kostenlose und Open-Source-FTP-Software zur Dateiübertragung zwischen Computern und Servern.
* **phpMyAdmin**: Ein webbasiertes Administrationswerkzeug für die Verwaltung von MySQL-Datenbanken.

### Projektabgrenzung

Dieses Projekt ist Teil eines größeren Vorhabens, das darauf abzielt, den Online-Shop von Putzmittel24 zu erweitern und verschiedene zusätzliche Funktionen und Inhalte bereitzustellen. Innerhalb dieses größeren Projektrahmens sind bestimmte Grenzen und Abgrenzungen definiert, um den Fokus und die Ziele klar zu halten. Zu den wichtigsten Aspekten der Projektabgrenzung gehören:

**Erweiterung des Online-Shops:** Das Hauptziel des Projekts ist die Integration neuer Produkte in den bestehenden Online-Shop von Putzmittel24. Dabei sollen sämtliche Informationen zu den Produkten, einschließlich Namen, SKU, Fotos, Booklet-PDFs und Preise, erfasst werden.

**Bereitstellung von Reinigungstipps**: Putzmittel24 plant die Einrichtung eines Informationsblocks auf seiner Website, der Reinigungstipps enthält. Dieser Block soll den Kunden zusätzliche nützliche Informationen bieten und das Engagement auf der Website erhöhen.

**Video-Kanal auf YouTube**: Um die Reichweite zu erhöhen und Kunden anzusprechen, wird auch die Einrichtung eines Video-Kanals auf YouTube geplant. Hier sollen Videos mit Anwendungstipps, Produktvorstellungen und anderen relevanten Inhalten veröffentlicht werden.

**Präsenz in sozialen Medien**: Zusätzlich zur YouTube-Plattform strebt Putzmittel24 eine Präsenz auf Facebook und Instagram an. Durch regelmäßige Beiträge und Interaktionen mit Kunden soll die Bekanntheit der Marke gesteigert und die Kundenbindung verbessert werden.

**Verbesserung der Website für Barrierefreiheit**: Ein weiteres wichtiges Ziel des Projekts ist die Verbesserung der Website von Putzmittel24 im Hinblick auf Barrierefreiheit. Dies umfasst die Anpassung der Website, um sie für Menschen mit Behinderungen zugänglicher zu machen und sicherzustellen, dass sie die Website ohne Einschränkungen nutzen können.

Ich bin nur für die Erweiterung des Online-Shops zuständig.

### Projektbegründung

Die Projektbegründung basiert auf einem klaren Ist-Zustand, der die ineffizienten und fehleranfälligen manuellen Prozesse bei der Verwaltung des Produktbestands des Online-Shops verdeutlicht. Der gegenwärtige Zustand zeigt, dass die manuelle Extraktion von Produkt-informationen von den Websites der Großhändler zeitaufwendig ist und ein hohes Risiko für menschliche Fehler birgt. Dies führt zu ungenauen Produktinformationen und Verzögerungen bei der Aktualisierung der Produktliste.

Im Soll-Zustand wird deutlich, dass der Kunde eine umfassende Lösung benötigt, um den Prozess der Produktintegration erheblich zu optimieren und zu automatisieren. Die Anforderungen umfassen die automatisierte Extraktion von Produktinformationen von den Lieferantenwebseiten, das Herunterladen und Speichern von Fotos und PDFs, die automatische Generierung von Excel-Dateien zur Aktualisierung von Produktbeschreibungen und Bildern sowie die Konvertierung in das erforderliche CSV-Format für WooCommerce. Diese Lösung bietet nicht nur eine erhebliche Zeitersparnis, sondern minimiert auch das Risiko von Fehlern und verbessert die Effizienz der Produktverwaltung erheblich.

Die Entscheidung, Python für die Implementierung dieser Lösung zu verwenden, ergibt sich aus der Tatsache, dass eine definierte Kommunikation über Schnittstellen nicht möglich ist, da keine Zugangsdaten vorhanden sind. Somit ist Python die geeignete Plattform, um die automatisierten Prozesse effektiv umzusetzen und die Anforderungen des Kunden zu erfüllen.

## Projektablauf

### Technische Planung des Projektzieles

Die technische Planung des Projektziels beinhaltet die detaillierte Analyse und Planung der Schritte, die erforderlich sind, um das definierte Ziel des Projekts zu erreichen. Dies umfasst die Auswahl der geeigneten Technologien, Tools und Methoden sowie die Festlegung des Arbeitsablaufs und der Ressourcen.

Im vorliegenden Projektziel, die Automatisierung des Produktintegrationsprozesses in den Online-Shop des Kunden, stehen folgende technische Schritte im Vordergrund:

1. **Auswahl der Programmiersprache und Frameworks:** Die Wahl fällt auf Python als Hauptprogrammiersprache aufgrund seiner Vielseitigkeit und der verfügbaren Bibliotheken für Web-Scraping und Datenverarbeitung. Als Framework wird BeautifulSoup für das Web-Scraping und pandas für die Datenverarbeitung verwendet.
2. **Implementierung des Web-Scraping-Algorithmus**: Entwicklung eines Python-Programms, das automatisch die Webseiten der Lieferanten durchsucht, die benötigten Produktinformationen extrahiert und in einer strukturierten Form speichert.
3. **Automatisierte Dateiverwaltung:** Einrichtung eines Systems zur automatischen Speicherung von heruntergeladenen Fotos und PDFs in einem ordnungsgemäß organisierten Ordnerstrukturen, um die Auffindbarkeit und Organisation zu optimieren.
4. **Entwicklung von Excel-Exportfunktionen:** Implementierung von Funktionen zur automatischen Generierung von Excel-Dateien für das SEO-Team und das Grafikdesign-Team, die für die Aktualisierung von Produktbeschreibungen und Bildern zuständig sind.
5. **Integration mit WooCommerce:** Konvertierung der Excel-Dateien in das erforderliche CSV-Format für WooCommerce und automatische Aktualisierung des Produktbestands im Online-Shop des Kunden.
6. **Testen und Validierung:** Durchführung von umfassenden Tests, um sicherzustellen, dass der entwickelte Algorithmus fehlerfrei funktioniert und den Anforderungen des Kunden entspricht. Validierung des Systems durch den Kunden, um sicherzustellen, dass es seinen Anforderungen entspricht.

# Ressourcen- und Ablaufplan

## Personalplanung

An der Programmierung der Python CSV- und XLSX-Funktionen arbeite ich alleine.

## Sachmittelplanung

In der Sachmittelplanung werden die benötigten Ressourcen für das Projekt identifiziert, beschafft und verwaltet. Dazu gehören physische Gegenstände wie Computer, für die bereits vorhandene Geräte verwendet werden, sowie Softwarelizenzen wie WooCommerce und Visual Studio Code, die bereits bezahlt wurden oder kostenlos verfügbar sind.

## Zeitliche Planung

Das Projekt ist geplant, vom 19. Februar 2024 bis zum 12. April 2024 abgeschlossen zu werden.

**# Zeitplanung**

**## Technische Analyse des Ist-Zustandes: 5 Stunden**

- Untersuchung des aktuellen manuellen Prozesses zur Integration neuer Produkte: 2 Stunden

- Evaluierung der bestehenden Datenextraktionsmethoden und -werkzeuge: 2 Stunden

- Meeting: 1 Stunde

**## Herstellen der Arbeitsumgebung: 3 Stunden**

- Aufsetzen einer geeigneten Entwicklungsumgebung, einschließlich der erforderlichen Softwaretools wie Python, BeautifulSoup oder Selenium, Xampp: 3 Stunden

**## Datenextraktion von 4 Webseiten: 31 Stunden**

- Python Skript schrieben: 10 Stunden

- Durchführung von Modultests auf die Datenextraktion mittels Python: 5 Stunden

- Durchführung eines Systemtests, um die gesamte Datenextraktionspipeline zu überprüfen: 5

Stunden

- Skript anpassen und herunterladen: 9 Stunden

- Sicherung auf Google Drive sowie auf zwei separaten Festplatten für zusätzliche Redundanz:

1 Stunde

- Meeting: 1 Stunde

**## Datenkonvertierung für WooCommerce: 11 Stunden**

- Umwandlung der JSON-Daten in das erforderliche WooCommerce CSV-Format mittels

Python: 7 Stunden

- WordPress Localhost-Shop installieren: 2 Stunden

- Tests: 1 Stunde

- Hochladen der Dateien auf den Test-WooCommerce-Shop im lokalen Host: 1 Stunde

**## XLSX-Liste der Produktfotos für das Grafikerteam: 1 Stunde**

- Bereitstellung einer XLSX-Liste der Produktfotos zur eingehenden Prüfung: 1 Stunde

**## XLSX-Liste der Texte für das SEO-Team: 1 Stunde**

- Bereitstellung einer XLSX-Liste für das SEO-Team zur Überarbeitung und Ergänzung von Texten: 1 Stunde

**## Erstellung einer Excel-Tabelle für den Kunden: 1 Stunde**

- Erstellung einer Excel-Tabelle zur erleichterten Verwaltung von Preisen und Lagerbeständen durch den Kunden: 1 Stunde

**## Implementierung eines Systems für die Datenrückkonvertierung: 2 Stunden**

- Implementierung eines Systems zur Konvertierung der bearbeiteten Excel-Dateien zurück in das WooCommerce CSV-Format: 2 Stunden

**## Lokale Tests auf WordPress: 7 Stunden**

- Hochladen der CSV-Dateien lokal auf WordPress: 1 Stunde

- umfassende Tests: 6 Stunden

**## Export und Übertragung in den Online-Shop des Kunden: 2 Stunden**

- Export der CSV-Dateien vom lokalen Shop und Übertragung in den Online-Shop des Kunden: 2 Stunden

**## Erneute Tests zur Sicherstellung eines reibungslosen Ablaufs: 6 Stunden**

- Durchführung erneuter Tests, um die ordnungsgemäße Funktion sicherzustellen: 6 Stunden

**## Dokumentation von Projektdetails, Problemen und Fortschritten: 10 Stunden**

- Dokumentation sämtlicher Projektdetails, auftretender Probleme und erzielter Fortschritte in Asana und Google Drive. 9 Stunden

- Meeting: 1 Stunde

**## Gesamt: 80 Stunden**

Anhang 1: Aktivität Diagramm

## Kostenplanung

|  |  |  |  |
| --- | --- | --- | --- |
| Person | Stundenlohn | Anzahl Stunden | Total |
| ich | 24,69 € | 80 | 1.975,2 € |
| Chef | 42,5 € | 3 | 127,5 € |
| Sub total | | | 2.102,7 € |
| Energie pauschal pro Tag 80/8 \* 6 | | | 60 € |
| TOTAL | | | 2.162,7 € |

*Quelle :* [*https://www.stepstone.de/gehalt/Programmierer-in.html*](https://www.stepstone.de/gehalt/Programmierer-in.html)

-> Programmierer Stundenlohn von 24,69 €

*Quelle :* [*https://www.stepstone.de/gehalt/Head-of-IT.html*](https://www.stepstone.de/gehalt/Head-of-IT.html)

-> Head of IT-Stundenlohn von 42,5 €

*Quelle :* [*https://www.test.de/Homeoffice-Pauschale-So-rechnen-Sie-mit-dem-Finanzamt-ab-5788023-0/*](https://www.test.de/Homeoffice-Pauschale-So-rechnen-Sie-mit-dem-Finanzamt-ab-5788023-0/)

-> Home­office-Pauschale 6 € pro Tag

Anhang 2 Programmierer Stundenlohn

Anhang 3 Head of IT-Stundenlohn

Anhang 4 Home­office-Pauschale

# Durchführungsphase

## Herstellen der Arbeitsumgebung

1. **Installation von Software und Tools**: Dazu gehören Python, Git, XAMPP (für die lokale Entwicklungsumgebung), Visual Studio Code (oder ein ähnlicher Texteditor) und möglicherweise andere Bibliotheken oder Frameworks, je nach den spezifischen Anforderungen des Projekts.
2. **Einrichten des Projektverzeichnisses:** Es erfolgt die Erstellung eines dedizierten Verzeichnisses für das Projekt auf dem Arbeitscomputer. Die Dateien und Ordner werden in einer logischen Struktur organisiert, um eine einfache Navigation und Auffindbarkeit zu gewährleisten.
3. **Einrichtung der Versionskontrolle**: Um Änderungen am Code zu verfolgen und zu verwalten, wird ein Git-Repository eingerichtet und im Projektverzeichnis initialisiert.
4. **Dokumentation erstellen**: Einrichtung einer Markdown-Datei zum Festhalten von Notizen und Beobachtungen.
5. **Einrichtung von Kommunikationskanälen**: Slack und Asana werden eingerichtet, um die Kommunikation mit anderen Teammitgliedern, dem Kunden und anderen relevanten Parteien zu ermöglichen.

## Eigentlich Umsetzung

### Daten von den Webseiten holen

Dieser Prozess gilt für alle 4 Webseiten:

Anhang 5: Aktivität Diagramm

#### 4.2.1.1 Navigation auf der Webseite mit Python

Ein Python-Skript wird zunächst erstellt, das die URLs der Haupt- und Unterseiten erfasst und in einem Protokolltextdokument mit dem Namen 'Protokoll.txt' speichert. In dem Protokolltextdokument 'Protokoll.txt' ist auch eine Seitennummerierung enthalten, die mit der erwarteten Anzahl der Seiten verglichen werden kann, die aus der Navigation berechnet werden kann **(siehe Test 1).**

Anhang 6: URL-Programmierung

Anhang 7: Log Output

#### 4.2.1.2 Erstellen von Ordnern für Produkte, PDFs und Bilder

Daraufhin wird eine Ordnerstruktur auf der Festplatte erstellt, wobei für jedes Produkt ein Ordner basierend auf dem Namen der entsprechenden Seite angelegt wird. Dabei wird überprüft, ob die Namen der Ordner gültig sind, um sicherzustellen, dass sie nur erlaubte Zeichen enthalten **(siehe Test 2).**

Anhang 8: Erstellen von Ordnern für Produkte, PDFs und Bilder Programmierung

Anhang 9: Folders Anzahl

#### 4.2.1.3 Download

Dies gilt für alle Elemente der Seiten (im folgenden Beispiel: h1):

**# Herunterladen eines Elements von einer Seite**

Zunächst wird das Skript nur auf einer Seite getestet, um die URL und den Text des h1-Elements in einer JSON-Datei aufzulisten. Vor dem Loop wird eine Liste mit dem Namen all\_values initialisiert. Für jede Seite wird ein Dictionary namens product\_info initialisiert, in dem der Wert des h1-Elements unter product\_info['h1'] gespeichert wird. Python fügt dieses Dictionary dann der Liste hinzu, und schließlich wird die Liste in eine JSON-Datei geschrieben. Das Protokolldokument enthält nun die URLs der Seiten sowie das h1-Element der getesteten Seite."

Anhang 10: Download Programmierung

Anhang 11: JSON-Output

Anhang 12: Protokoll

**# Herunterladen eines Elements von allen Seiten**

Wenn das Auflisten des Test-h1-Elements erfolgreich ist, wird das Skript für alle Seiten ausgeführt und im Protokolldokument vermerkt. Bei auftretenden Problemen wird die URL der Seite im Protokolldokument angegeben, bei der das Problem aufgetreten ist. Das Skript wird entsprechend angepasst (**siehe Test 3 als Unit Test**). Nach erfolgreicher Ausführung des Skripts auf allen Seiten wird versucht, das nächste Element zu extrahieren.

Nachdem das Skript erfolgreich das h1-Element von allen Seiten extrahiert hat, wird der gleiche Ansatz auf andere Elemente angewendet, einschließlich Beschreibung, Kurzbeschreibung, PDFs, Bilder, Kategorien, Tags usw. Jedes Element wird auf dieselbe Weise im JSON-Format aufgelistet und Bemerkungen im Protokolldokument vermerkt.

Anhang 13: JSON

Wenn BeautifulSoup das gesuchte Element nicht finden kann oder falsch extrahiert, wird das Skript auf Selenium umgeschaltet, um eine genauere Suche durchzuführen. Selenium bietet den Vorteil, dass es effektiv ist, wenn Webseiten dynamische Inhalte haben oder JavaScript zur Generierung von Elementen verwendet wird, die von BeautifulSoup möglicherweise nicht erfasst werden. Alle Probleme oder Fehler werden im Protokolldokument vermerkt, und das Skript wird entsprechend angepasst, um das Problem zu lösen.

Anhang 14: Protokoll:

**# Herunterladen aller Produkte (alle Elemente auf allen Seiten)**

Sobald das Skript erfolgreich für einzelne Elemente funktioniert, wird es auf alle Seitenangewendet, um die gesamten Elemente zu extrahieren. Dieser Prozess wird systematisch durchgeführt, um sicherzustellen, dass alle relevanten Informationen von jeder Seite erfasst werden. Jegliche auftretenden Probleme werden im Protokolldokument festgehalten und entsprechende Anpassungen am Skript vorgenommen, um diese zu beheben **(siehe Test 3 als Integration Test).**

Anhang: 15 JSON Output

Anhang 16: Protokoll

Der erste Teil des Projekts ist abgeschlossen. Das Skript, die JSON-Datei und das Protokoll werden im persönlichen Git-Repository gespeichert, das sowohl auf dem Google Drive von Pixelding als auch auf dem Arbeitscomputer von Pixelding und meinem Heimcomputer verfügbar ist. Die PDF- und Bilderordner werden ebenfalls in diesen Repositorien abgelegt, um eine konsistente und zugängliche Speicherung sicherzustellen.

### Daten in lokale WordPress importieren

Dieser Prozess gilt für alle 4 JSON-Dateien**:**

#### 4.2.2.1 lokale WordPress installieren

Zuerst wird für eine Webseite eine lokale Wordpress-Instanz eingerichtet. Anschließend wird WooCommerce auf dieser lokalen Wordpress-Instanz installiert, gefolgt von der Installation des Plugins WooCommerce Germanized.

#### 4.2.2.2 Daten für WordPress WooCommerce in CSV, und XLSX umbauen

Um die Daten für die Integration in den WordPress WooCommerce-Shop vorzubereiten, wurde ein Python-Skript entwickelt. Dieses Skript liest strukturierte Informationen über Produkte aus einer JSON-Datei und wandelt sie in das erforderliche CSV- und XLSX-Format um. Die JSON-Datei enthält wichtige Details wie Namen, Beschreibungen und andere relevante Informationen zu den Produkten. Während CSV und Excel die gleichen Daten enthalten, bietet Excel aufgrund seiner tabellarischen Darstellung eine übersichtlichere und benutzerfreundlichere Möglichkeit, die Informationen zu überprüfen und zu bearbeiten.

In dem CSV unter dem Feld 'Typ' bieten sich zwei Auswahlmöglichkeiten an: 'Einfaches Produkt' und 'Variable'. Variable Produkte können entweder den Wert 'Variable' oder 'Variation' haben. Variationen sind, wie der Name schon sagt, Variationen oder unterschiedliche Ausprägungen eines Produkts. Artikelnummern für einfache Produkte und Variation können aus der JSON-Datei eingetragen werden. Für variable Produkte muss eine eindeutige Artikelnummer von Python erstellt werden.

Anhang 17: Produkt CSV Import Schema

Anhang 18: Aktivität Diagramm

**# Skript ausführen für einen Produkt**

Das Skript beginnt damit, einen ersten Index der JSON-Daten zu lesen und in die beiden verschiedenen Formate zu konvertieren, um zu testen, ob der Skript funktioniert. **(siehe Test 4 als Unittest).**

**# Skript ausführen für alle Produkte**

Wenn dieser Test erfolgreich ist, wird das Skript auf den gesamten JSON-Datensatz angewendet **(siehe Test 4 als Integration Test).**

Durch die Entwicklung dieses Skripts wurde sichergestellt, dass die Daten für den Online-Shop in den erforderlichen Formaten vorliegen und einfach in das WooCommerce-System integriert werden können. Dies trägt dazu bei, den Prozess der Produktintegration zu rationalisieren und die Effizienz bei der Verwaltung des Online-Shops zu verbessern.

Anhang 19: CSV Output

#### 4.2.2.3 CSV in WordPress WooCommerce importieren

Dann wird die CSV-Datei mit den Produktdaten über die WordPress WooCommerce-GUI hochgeladen. Im Falle eines Fehlers in der CSV-Datei wird dies hier gemeldet. **(siehe Test 5)** Nach erfolgreicher Integration wird überprüft, ob die Produkte auf der Website ordnungsgemäß dargestellt werden und ob alle Informationen korrekt angezeigt werden**. (siehe Test 6)**

Anhang 20: WooCommerce Import GUI 1

Anhang 21: WooCommerce Import GUI 2

Anhang 22: WooCommerce Import GUI 3

### XLSX für Grafiker, SEO, und Kunde anfertigen

Dieser Prozess gilt für alle 4 JSON-Dateien:

#### 4.2.3.1 XLSX-Liste für den Kunden

Der Kunde benötigt eine XLSX-Liste aller Produkte, die entweder als einfaches Produkt oder als Variation definiert sind. Mit dieser Liste kann er Preise, Einkaufspreise und Lagerbestände verwalten. Das gleiche Skript wie oben wird ausgeführt, jedoch mit dem Unterschied, dass Variable-Produkte nicht in die XLSX-Datei geschrieben werden.

#### 4.2.3.2 XLSX-Liste für das SEO-Team

Das SEO-Team benötigt eine XLSX-Liste aller Produkte, um die Beschreibungen und Kurzbeschreibungen anzupassen. Dabei können sie auch die Parameter von Variable-Produkten anpassen, falls erforderlich. Diese XLSX-Datei wurde zusammen mit der CSV-Generierung automatisch erstellt.

**4.3.3.3 XLSX-Liste für die Grafiker**

Das Grafiker-Team benötigt ebenfalls eine XLSX-Liste aller Produkte, um die Bilder zu verwalten. Sie können die Bildpfade hier in Bezug auf die Ordnerstruktur ändern. Die Grafiker müssen die neuen Bilder manuell in die entsprechenden Ordner hochladen. Diese XLSX-Datei wurde zusammen mit der CSV-Generierung automatisch erstellt.

### Daten in lokale WordPress updaten

#### 4.2.4.1 Erstellen einer neuen lokale WordPress-WooCommerce-Instanz

Für die Aktualisierung der Daten in WordPress WooCommerce wird zunächst eine neue lokale WordPress-WooCommerce-Instanz erstellt. Die 4 ursprünglichen CSV-Dateien werden erneut geladen. Die vorherigen WordPress-WooCommerce-Instanzen dienen als Backup und können zum Vergleich herangezogen werden.

#### 4.2.4.2 Konvertierung der XLSX-files zu CSV-files mit Python

Anschließend werden die vier XLSX-Dateien des Kunden in vier CSV-Dateien mit Python umgewandelt. Dasselbe wird für die vier XLSX-Dateien des SEO-Teams und die vier XLSX-Dateien des Grafikteams durchgeführt. Mithilfe von Python werden nur die erforderlichen Spalten exportiert. Die Artikelnummern werden als Zeichenkette exportiert, wobei führende Nullen beibehalten werden. Die Dezimalzahlen für Verkaufs- und Einkaufspreise werden im Export durch einen Punkt anstelle eines Kommas dargestellt. Die deutschen Buchstaben werden bei der Kodierung mit 'utf-8' behandelt **(siehe Test 7).**

Anhang 23: XLSX zu CSV

#### 4.2.4.3 Hochladen der CSV in den neuen Lokalen WordPress

Die zwölf CSV-Dateien werden dann über die WooCommerce-GUI importiert, wobei die Option "Aktualisieren" ausgewählt wird **(siehe test5).** Nach erfolgreicher Prüfung kann die lokale WordPress-WooCommerce-Instanz auf den Online-Shop des Kunden übertragen werden **(siehe Test 6).**

### Daten in Onlineshop hochladen

#### 4.2.5.1 Exportieren der CSV-File aus dem Lokal Shop

Um die Daten in den Online-Shop zu integrieren, werden zunächst die Produktinformationen aus dem lokalen Shop über die WooCommerce GUI als CSV exportiert. Dabei werden die URLs für Bilder und PDFs noch lokal referenziert.

#### 4.2.5.2 Anpassen der Bilder und PDF URL in der CSV-File

Anschließend werden diese CSV-Dateien in Visual Studio Code geöffnet, um die lokalen URLs durch die Online-URLs des Online-Shops zu ersetzen.

*4.2.5.3 Onlinelagerung der Bilder und PDF*

Die Bilder und PDFs werden dann mit FileZilla (FTP) auf den Server des Kunden hochgeladen, wo WooCommerce sie später finden kann.

#### 4.2.5.4 Import der CSV-File in den Onlineshop

Nachdem der Online-Shop in den Bearbeitungsmodus versetzt wurde, erfolgt der Import der CSV-Dateien in WooCommerce über die GUI. WooCommerce lädt automatisch alle Produkte hoch. Die Bilder und PDFs werden gefunden und an den richtigen Orten referenziert**. (siehe Test 5 als Abnahmetest).** Abschließend wird überprüft, ob alle Daten korrekt hochgeladen und im Online-Shop angezeigt werden. Nach Abschluss der Überprüfung wird der Bearbeitungsmodus des Online-Shops deaktiviert**. (siehe Test 6 als Abnahmetest).**

Speichern in GitHub, Google und auf den Computer immer wieder!

## Testen

Tests spielen eine entscheidende Rolle in der Softwareentwicklung und bieten einen Mechanismus zur Bewertung der Funktionalität und Zuverlässigkeit von Code. Obwohl Tests keine fehlerfreie Leistung in allen Situationen garantieren können, bleiben sie das effektivste Mittel, um potenzielle Probleme vor der Bereitstellung zu identifizieren und zu beheben. Im Folgenden werden die während des oben beschriebenen Programmierprozesses durchgeführten Tests aufgelistet.

**# Test 1:** Es wird überprüft, ob alle Seiten erfolgreich gelesen werden können, um sicherzustellen, dass alle relevanten Informationen erfasst werden. Dabei wird auch die Gesamtanzahl der Seiten ermittelt, indem die Anzahl der Seiten in der Navigation gezählt und mit der Anzahl der gescrollten und nummerierten Seiten im Protokoll verglichen wird.

**# Test 2:** Es wird überprüft, ob die Anzahl der erstellten Ordner der Anzahl der erfassten Seiten entspricht. Dabei wird sicherzustellen versucht, dass jeder Produktordner basierend auf dem Namen der entsprechenden Seite korrekt erstellt wurde. Des Weiteren wird überprüft, ob die Namen der Ordner gültig sind und nur erlaubte Zeichen enthalten. Etwaige nicht erlaubte Zeichen werden automatisch von Python ersetzt, um die Konsistenz und Integrität der Ordnerstruktur zu gewährleisten.

**# Test 3:** Im Protokolldokument wird unter der Seitennummer und der URL eine neue Zeile mit dem gesuchten Element hinzugefügt. Es wird vermerkt, ob das Element gefunden wurde oder nicht. Das Python-Skript wird in einer Schleife ausgeführt. Innerhalb dieser Schleife wird *ein if-else*- oder *Try-Except*-Block eingerichtet, um sicherzustellen, dass das Skript auch dann ausgeführt wird, wenn ein Element nicht gefunden wird. Die Struktur des Protokolls zeigt gefundene Elemente nach eine Folge Minuszeichen (-) und nicht gefundene Elemente nach Pluszeichen (+).

Anhang 16: Protokoll

**# Test 4:** Dieses Testverfahren besteht darin, manuell visuell zu überprüfen, ob die Daten im Excel-Dokument in die richtigen Spalten übertragen wurden und ob die Variationen und Einheiten korrekt übernommen wurden.

**# Test 5:** Dieses Testverfahren erfolgt während des Hochladens der CSV-Datei über das WordPress WooCommerce-GUI-Upload-Formular. Dabei werden Fehler in der CSV-Datei gemeldet, entweder während des Hochladens oder direkt nach Abschluss des Vorgangs. Es wird auch die Anzahl der importierten Produkte angezeigt, sodass sie mit der Anzahl der Zeilen aus der Excel-Datei verglichen werden kann.

Anhang 24: WooCommerce Fehler Meldung

**# Test 6:** Nach der erfolgreichen Integration wird jede Seite überprüft, um sicherzustellen, dass die Produkte ordnungsgemäß dargestellt werden und dass alle Informationen korrekt angezeigt werden. Dazu wird jede Seite einzeln aufgerufen und mit ihrer ursprünglichen URL verglichen. Beim Lokal Test wird mit den Seiten des Herstellers verglichen, beim Abnahmetest wird mit den Seiten des Lokalshops verglichen. Außerdem werden alle Varianten-Dropdown-Menüs angeklickt, um sicherzustellen, dass sie ordnungsgemäß funktionieren.

**# Test 7:** Hierbei geht es darum, zu überprüfen, ob die deutschen Buchstaben ordnungsgemäß behandelt werden und ob die Dezimalzahlen für Verkaufs- und Einkaufspreise im Export durch einen Punkt anstelle eines Kommas dargestellt werden.

# Übergabe und Auswertung

## Projektübergabe

Die neuen Produkte wurden erfolgreich in den Online-Shop integriert. Die Dokumentation und Kopien des Codes, der Bilder und der PDF befinden sich auf den Computern und Google Drive von Pixelding sowie auf GitHub.

Die Excel-Dateien sind ebenfalls auf Google Drive gespeichert, und der Kunde kann bei Bedarf seine Preise ändern. Die SEO-Abteilung kann die Beschreibung und Kurzbeschreibung monatlich für SEO-Zwecke aktualisiert werden.

Zu diesem Zweck kann der Python-Code zum Konvertieren von Excel- in CSV-Dateien verwendet werden.

## Mitarbeiterschulung, Bedienungsanleitung

Es wurde eine Schulung durchgeführt in der Funktionsweise das Project vorgestellt wurde.

## Soll-Ist-Vergleich (inhaltlich, Zeitlich, kostentechnisch)

**Inhaltlich:**

Im Hinblick auf die inhaltliche Umsetzung des Projekts können wir feststellen, dass die ursprünglichen Ziele erfolgreich erreicht wurden. Die Python-Skripte wurden entwickelt, um Daten von den Webseiten der Hersteller herunterzuladen, ohne eine definierte Kommunikation über beispielsweise Schnittstellen. Diese Daten wurden anschließend in JSON-Dateien referenziert und organisiert, die als Grundlage zur Herstellung der CSV- und XLSX-Dokumente für Kunden, Grafiker und SEO dienten. Die CSV-Dateien konnten erfolgreich in den WooCommerce Online-Shop importiert werden. Darüber hinaus wurden Tests durchgeführt, um sicherzustellen, dass die Skripte ordnungsgemäß funktionieren und die Daten korrekt verarbeitet werden. Die Überprüfung der Website nach der Integration hat gezeigt, dass die Produkte ordnungsgemäß dargestellt werden und alle Informationen korrekt angezeigt werden. Insgesamt ist das Projekt inhaltlich erfolgreich umgesetzt worden.

**Zeitlich:**

Alle Punkte konnten pünktlich abgearbeitet werden außer dem Download-Prozess. Der beanspruchte mehr Zeit als erwartet, bedingt durch die Dynamik der generierten Inhalte, die nicht direkt von BeautifulSoup erfasst werden konnten. Der Übergang zu Selenium stellte anfangs eine kleine Herausforderung dar, erwies sich jedoch als äußerst lohnend. Trotz des anfänglichen Zeitverlustes konnte dieser später durch eine effiziente Vorgehensweise und zügige Fortschritte in den nachfolgenden Schritten wieder aufgeholt werden.

**Kostentechnisch:**

Die Kosten für das Projekt blieben im Rahmen des Budgets. Es waren keine zusätzlichen Ressourcen oder Ausgaben erforderlich, um das Projekt erfolgreich abzuschließen. Die Kosten wurden effizient verwaltet und alle Ressourcen wurden sorgfältig genutzt, um die Ziele des Projekts zu erreichen. Insgesamt wurden die Kosten für das Projekt gut kontrolliert und blieben im geplanten Rahmen.

## Technische Auswertung und Ausblick

Die Arbeit an diesem Projekt war äußerst lohnend. Die Implementierung des Python-Skripts zur Datenverarbeitung hat gezeigt, wie effizient und zeitsparend Automatisierung sein kann. Der Ausblick für das Programm ist vielversprechend. In Zukunft könnte es weiterentwickelt werden, um zusätzliche Funktionen und Verbesserungen zu integrieren, wie beispielsweise eine benutzerfreundliche Oberfläche für den CSV Excel Konvertierung.

## Persönliche Auswertung

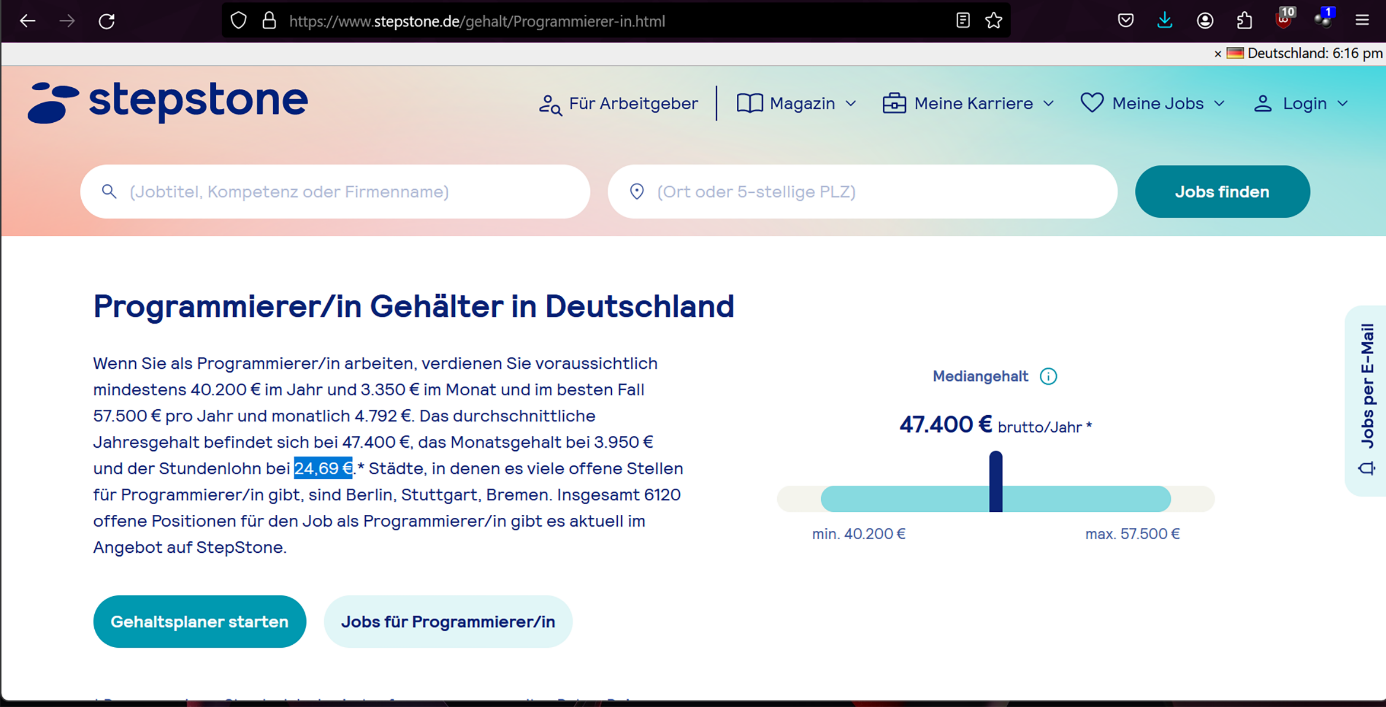
Die Zeit verging wie im Flug, und ich habe es wirklich genossen, an diesem Projekt zu arbeiten. Es war eine großartige Gelegenheit, meine Fähigkeiten in Python-Programmierung und Datenverarbeitung zu verbessern und gleichzeitig einen konkreten Beitrag zum Unternehmen zu leisten.

Abbildung 1 Aktivität Diagramm

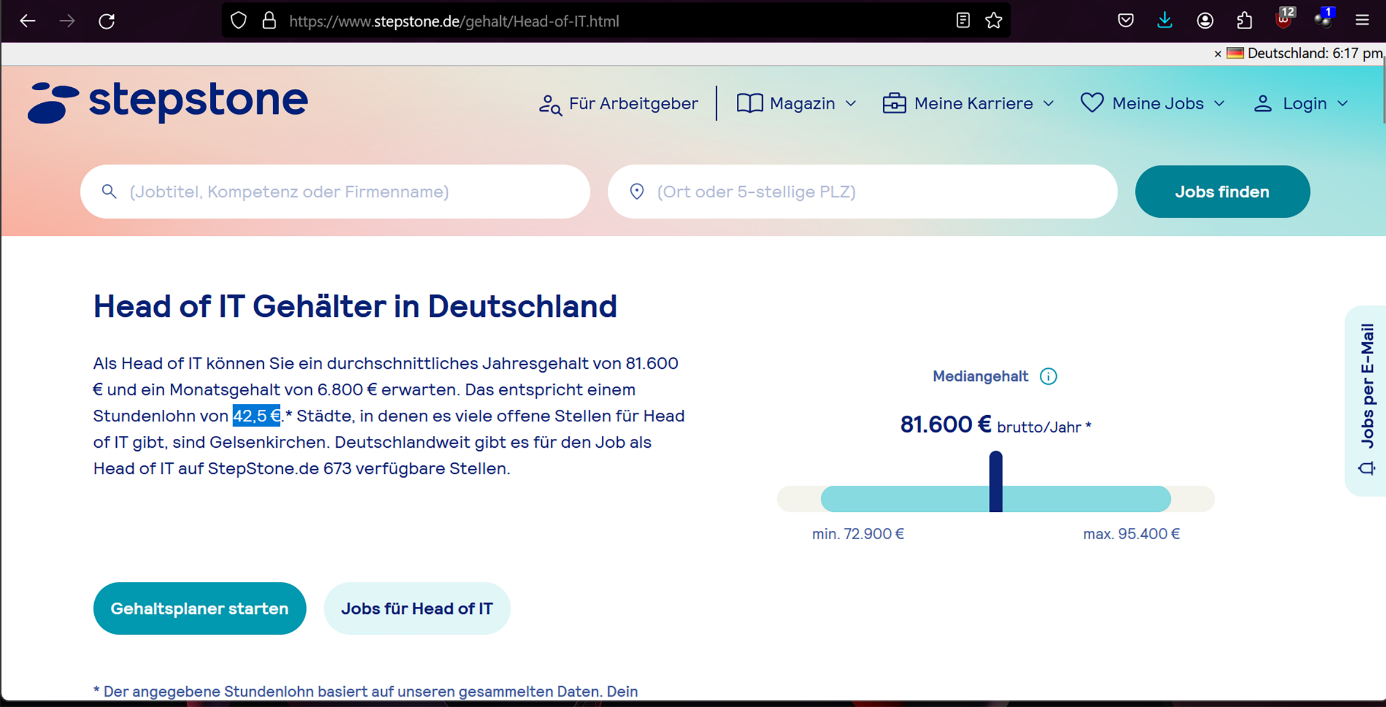
A diagram of a software development process

Description automatically generated

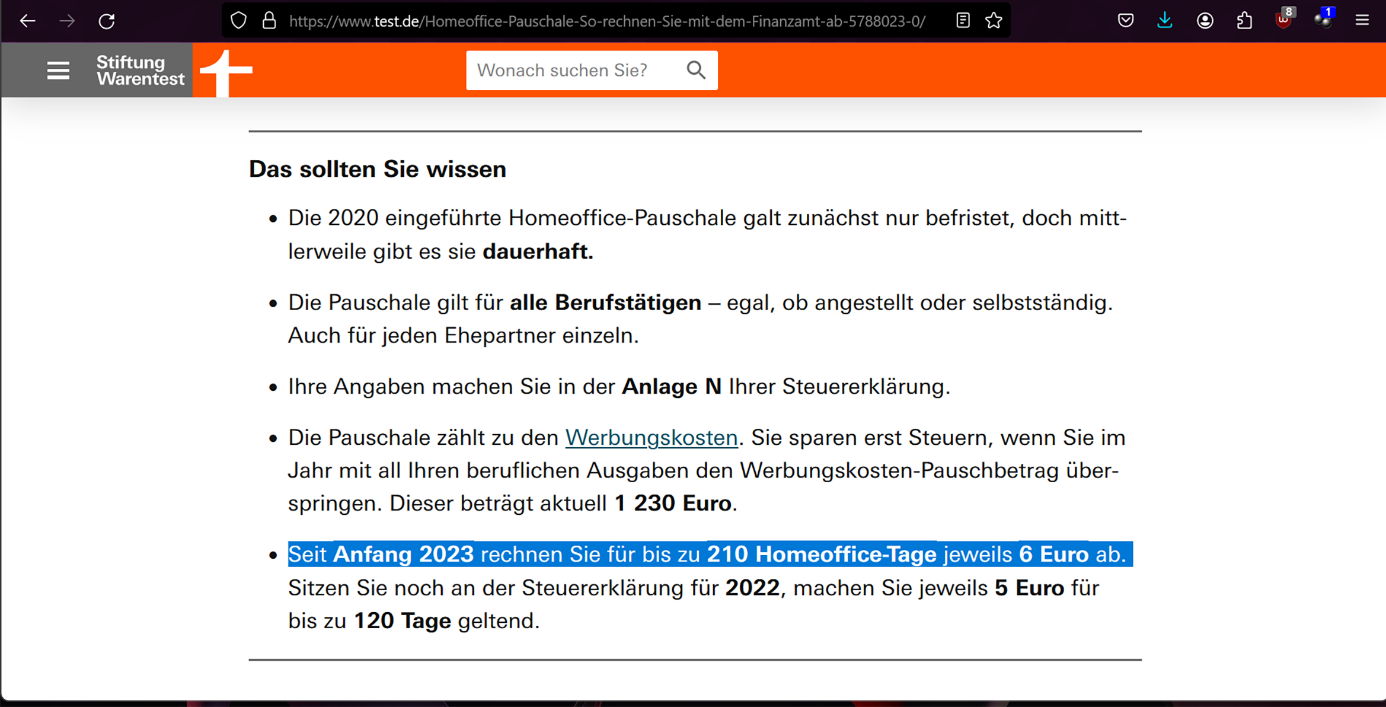
Anhang 2: Programmierer Stundenlohn



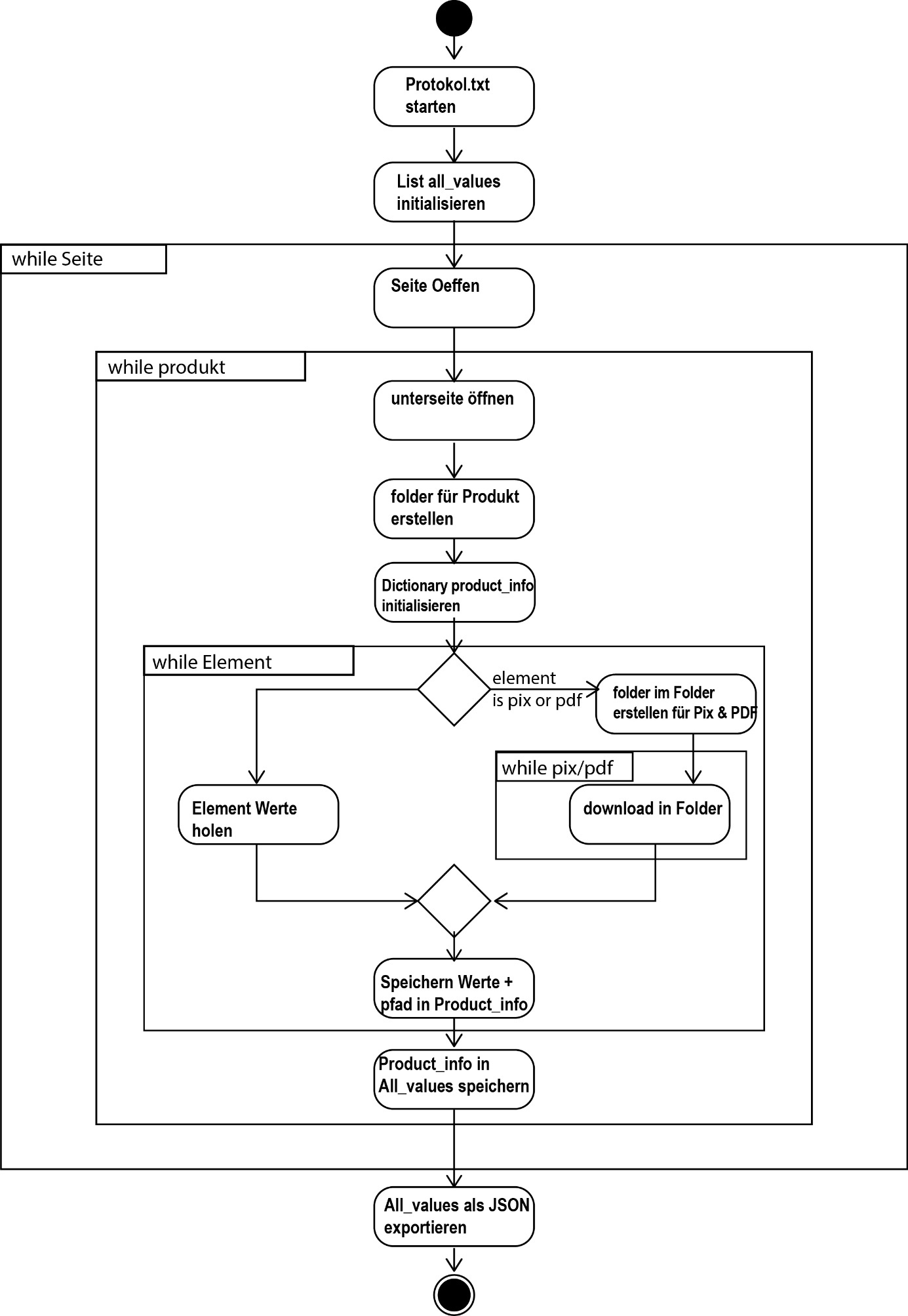
Anhang 3: Head of IT-Stundenlohn



Anhang 4: Home­office-Pauschale



Anhang 5: Aktivität Diagramm

****

Anhang 6: URL-Programmierung

#             \_

#            | |

#  \_   \_ \_ \_\_| |\_\_\_

# | | | | '\_\_| / \_\_|

# | |\_| | |  | \\_\_ \

#  \\_\_,\_|\_|  |\_|\_\_\_/

urls = [f'https://shop.buzil.de/anwendungsbereich/?p={i}&f=' for i in range(1, 14)]

all\_values = []

# Open the log file in append mode to avoid overwriting previous logs

with open(log\_file\_name, 'a') as log\_file:

    # Define a custom print function that writes to both console and log file

    def custom\_print(\*args, \*\*kwargs):

        # Convert the arguments to strings

        messages = [str(arg) for arg in args]

        # Concatenate the strings with a space separator

        message = ' '.join(messages)

        print(message, \*\*kwargs) # Print to console

        log\_file.write(message + '\n') # Write to log file

    n = 0

    subpage = 0

    for url in urls:

        n += 1

        custom\_print(f'main page {n} from 13')

        custom\_print("----------------------------------")

        custom\_print("url:", url)

        # Make an HTTP GET request to the website

        response = requests.get(url)

        # Check if the request was successful

        if response.status\_code == 200:

            # Parse the HTML content

            soup = BeautifulSoup(response.text, 'html.parser')

            # Find all <div> elements with the specified class

            item = soup.find('div', class\_='tile-view-phone')

            # todo Loop Produkts

            a\_tags = item.find\_all('a', class\_='product--link')

            for a\_tag in a\_tags:

                subpage += 1

                custom\_print('')

                custom\_print(f'page {subpage} from 145')

                product\_info = {}  # Create a new dictionary for each a\_tag

                product\_info['a'] = a\_tag['href']

                custom\_print(f'- found {product\_info["a"]}')

                custom\_print('')

Anhang 7: Log Output

main page 1 from 13

----------------------------------

url: https://shop.buzil.de/anwendungsbereich/?p=1&f=

page 1 from 145

- found https://shop.buzil.de/spezial/air-provence-g-565t

page 2 from 145

- found https://shop.buzil.de/kuechen-/lebensmittelhygiene/bistro-g-435

page 3 from 145

- found https://shop.buzil.de/sanitaer/bucasan-clear-g-463

page 4 from 145

- found https://shop.buzil.de/sanitaer/bucasan-fresh-p-920

page 5 from 145

- found https://shop.buzil.de/sanitaer/bucasan-hit-p-921

page 6 from 145

- found https://shop.buzil.de/sanitaer/bucasan-sanibond-g-457

page 7 from 145

- found https://shop.buzil.de/sanitaer/bucasan+-intense-p-922

page 8 from 145

- found https://shop.buzil.de/spezial/buz-air-ambra-g-564

page 9 from 145

- found https://shop.buzil.de/kuechen-/lebensmittelhygiene/buz-calc-ex-g-437

page 10 from 145

- found https://shop.buzil.de/spezial/buz-fresh-magic-g-567

page 11 from 145

- found https://shop.buzil.de/oberflaeche/buz-glass-p-912

…

Anhang 8: Erstellen von Ordnern für Produkte, PDFs und Bilder Programmierung

# create folder with that name

invalid\_chars = ['<', '>', ':', '"', '/', '\\', '|', '?', '\*']

# Extract the name from the href

folder\_name = os.path.basename(product\_info['a'])

# Ersetze ungültige Zeichen durch Unterstriche

for char in invalid\_chars:

    folder\_name = folder\_name.replace(char, '\_')

# Create a folder with the name from the href

folder\_path = os.path.join(os.getcwd(), folder\_name)

os.makedirs(folder\_path, exist\_ok=True)

custom\_print(f'- - making folder {folder\_path}')

Anhang 9: Folders Anzahl



Anhang 10: Download Programmierung

#... rest of code

# todo linked page

# Make an HTTP GET request to the linked page

response2 = requests.get(a\_tag['href'])

# Check if the request was successful

if response2.status\_code == 200:

# Parse the HTML content

    soup2 = BeautifulSoup(response2.text, 'html.parser')

    # Extract all <div> elements with the specified ID

    section = soup2.find('div', id='contentwrap')

    #  \_     \_\_

    # | |   /  |

    # | |\_\_ `| |

    # | '\_ \ | |

    # | | | || |\_

    # |\_| |\_\\_\_\_/

    # Extract the h1 element within the section

    h1\_element = section.find('h1')

    if h1\_element:

        h1\_text = h1\_element.get\_text().strip()

        # strip is trimming empty space

        product\_info['h1'] = h1\_text

        custom\_print(f'- - FOUND H1')

    else:

        product\_info['h1'] = ''

        custom\_print(f'+ + h1 is missing ?')

# ----------------------------------------

    # write to all values

    all\_values.append(product\_info)

#    \_

#   (\_)

#    \_ \_\_\_  \_\_\_  \_ \_\_

#   | / \_\_|/ \_ \| '\_ \

#   | \\_\_ \ (\_) | | | |

#   | |\_\_\_/\\_\_\_/|\_| |\_|

#  \_/ |

# |\_\_/

# Specify the directory path where the JSON file is located

directory\_path = ''

# Specify the file names

json\_file\_name = f'buzil\_{current\_datetime}.json'

# Save all\_values as a JSON file for each URL

with open(json\_file\_name, 'w', encoding='utf-8') as json\_file:

    json.dump(all\_values, json\_file, ensure\_ascii=False, indent=4)

print("JSON file page created.")

json\_file\_path = os.path.join(directory\_path, json\_file\_name)

Anhang 11: JSON-Output

[

    {

        "a": "https://shop.buzil.de/spezial/air-provence-g-565t",

        "h1": "Air Provence"

    }

]

Anhang 12: Protokoll

main page 1 from 13

----------------------------------

url: https://shop.buzil.de/anwendungsbereich/?p=1&f=

page 1 from 145

- found https://shop.buzil.de/spezial/air-provence-g-565t

- - making folder C:\Users\greg\Documents\sites\project\air-provence-g-565t

- - - FOUND H1

Anhang 13: JSON

[

    {

        "a": "https://shop.buzil.de/spezial/air-provence-g-565t",

        "h1": "Air Provence"

    },

    {

        "a": "https://shop.buzil.de/kuechen-/lebensmittelhygiene/bistro-g-435",

        "h1": "Bistro"

    },

    {

        "a": "https://shop.buzil.de/sanitaer/bucasan-clear-g-463",

        "h1": "Bucasan® Clear"

    },

    {

        "a": "https://shop.buzil.de/sanitaer/bucasan-fresh-p-920",

        "h1": "Bucasan® Fresh"

    },

…

Anhang 14: Protokoll

main page 1 from 13

----------------------------------

url: https://shop.buzil.de/anwendungsbereich/?p=1&f=

page 1 from 145

- found https://shop.buzil.de/spezial/air-provence-g-565t

- - making folder C:\Users\greg\Documents\sites\project\air-provence-g-565t

- - - FOUND H1

page 2 from 145

- found https://shop.buzil.de/kuechen-/lebensmittelhygiene/bistro-g-435

- - making folder C:\Users\greg\Documents\sites\project\bistro-g-435

- - - FOUND H1

page 3 from 145

- found https://shop.buzil.de/sanitaer/bucasan-clear-g-463

- - making folder C:\Users\greg\Documents\sites\project\bucasan-clear-g-463

- - - FOUND H1

page 4 from 145

- found https://shop.buzil.de/sanitaer/bucasan-fresh-p-920

- - making folder C:\Users\greg\Documents\sites\project\bucasan-fresh-p-920

- - - FOUND H1

page 5 from 145

- found https://shop.buzil.de/sanitaer/bucasan-hit-p-921

- - making folder C:\Users\greg\Documents\sites\project\bucasan-hit-p-921

- - - FOUND H1

…

Anhang 15: JSON Output

[

    {

        "a": "https://shop.buzil.de/spezial/air-provence-g-565t",

        "h1": "Air Provence G 565T",

        "image\_urls": [

            "https://shop.buzil.de/media/image/0a/1a/ca/2QOLn7huZ4R9dO1\_1280x1280.jpg",

            "https://shop.buzil.de/media/image/c8/8b/07/2QOLn7huZ4R9dO1\_200x200.jpg",

            "https://shop.buzil.de/media/image/7e/d6/56/2QOLn7huZ4R9dO1.jpg",

            "https://shop.buzil.de/media/image/5d/0e/77/2QOLn7huZ4R9dO1\_600x600.jpg",

            "https://shop.buzil.de/media/image/3b/f1/21/2QOLn7huZ4R9dO1\_600x600@2x.jpg"

        ],

        "image\_local": [

            "air-provence-g-565t/img/2QOLn7huZ4R9dO1\_1280x1280.jpg",

            "air-provence-g-565t/img/2QOLn7huZ4R9dO1\_200x200.jpg",

            "air-provence-g-565t/img/2QOLn7huZ4R9dO1.jpg",

            "air-provence-g-565t/img/2QOLn7huZ4R9dO1\_600x600.jpg",

            "air-provence-g-565t/img/2QOLn7huZ4R9dO1\_600x600@2x.jpg"

        ],

        "product\_detail\_description\_section": [

            {

                "header": "Gebrauchsfertiger Raumspray mit Aktiv-Geruchsblocker",

                "content": "In Wohn- und Geschäftsräumen, öffentlichen Einrichtungen wie Krankenhäusern, Altenheimen sowie in der Gastronomie, Sanitär- und WC-Anlagen, Aufenthalts- und Umkleideräumen, Sport- und Wellnessbereichen etc.; für die schnelle und langanhaltende Geruchsvernichtung und Luftverbesserung"

            },

            {

                "header": "Produktvorteile",

                "content": [

                    "Aromatischer Duft, der an frisch geernteten Lavendel erinnert sorgt für eine angenehme Raumatmosphäre",

                    "Neutralisiert schnell unangenehme Gerüche mit Aktiv-Geruchsblocker",

                    "Hinterlässt keine Flecken",

                    "Umweltfreundlicher Zerstäuber ohne Treibgas"

                ]

            }

        ],

…

Anhang 16: Protokoll

main page 1 from 13

----------------------------------

url: https://shop.buzil.de/anwendungsbereich/?p=1&f=

page 1 from 145

- found https://shop.buzil.de/spezial/air-provence-g-565t

- - making folder C:\Users\Praktikanten Alpha\Desktop\GegPraktikum\greg\_praktikum1\buzil\air-provence-g-565t

- - - FOUND H1

- - - DESCRIPTION-DATA

- - - - downloading description\_data: 1

- - - - downloading description\_data: 2

- - - ADDITIONAL DATA

- - - - downloading additionail detail: 1

- - - - - downloading additionail detail

- - - - downloading additionail detail: 2

- - - - - downloading additionail detail

- - - HINT

- - - DOWNLOADING PRODUKKT DOWNLOAD TITLE 1

- - - PRODUKT DOWNLOAD CONTENT

- - - - product\_download\_content found

- - - - - product\_download\_content li found: 1

- - - - - - product\_download li

- - - - - - - downloading text and pdf for Betriebsanweisungen

- - - - - - - - downloading PDF for Betriebsanweisungen

- - - - - product\_download\_content li found: 2

…

Anhang 17: Produkt CSV Import Schema A white paper with black text

Description automatically generated

A white sheet of paper with black text

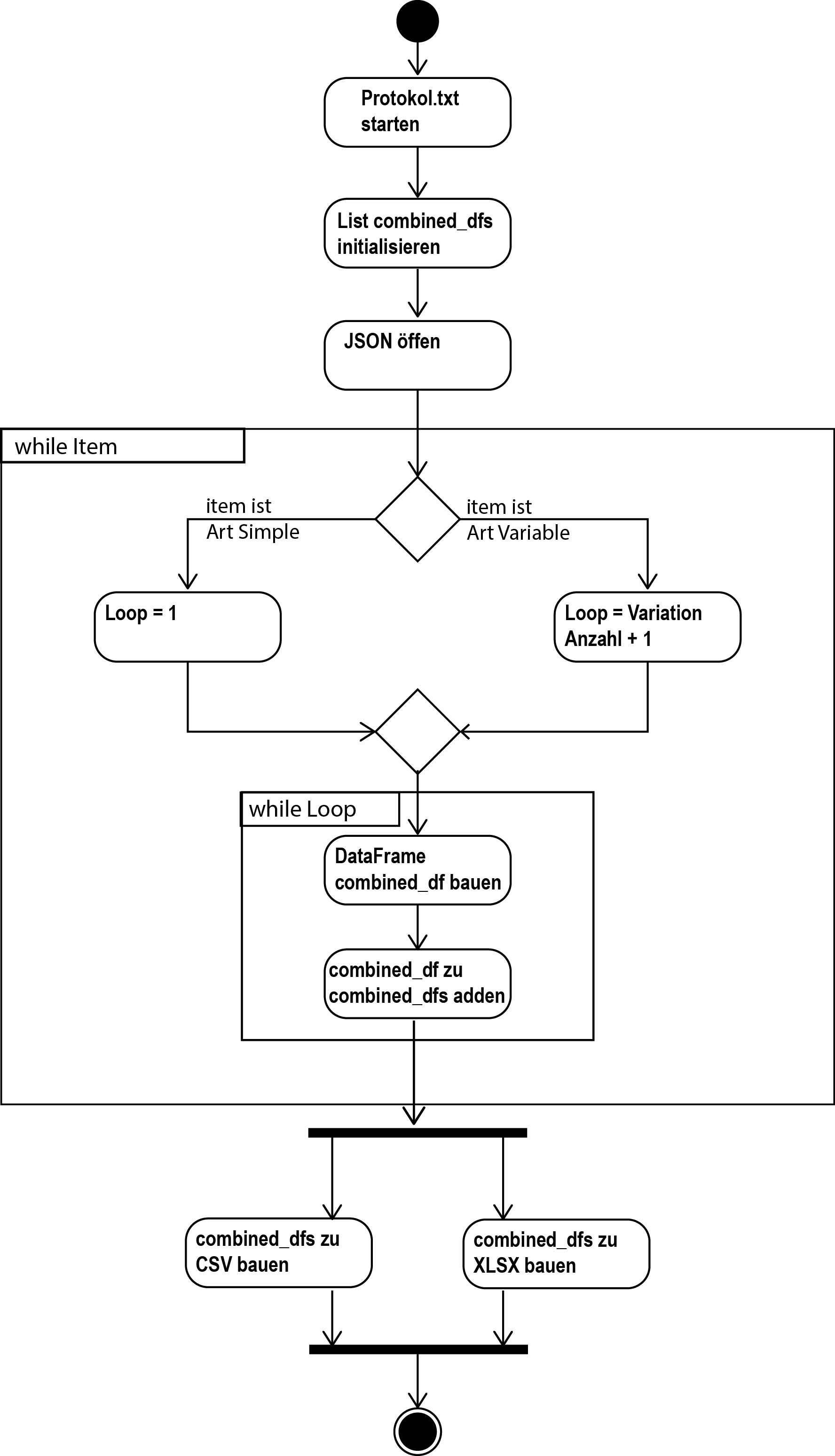
Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Anhang 18: Aktivität Diagramm



Anhang 19: CSV Output

A screen shot of a computer

Description automatically generated

Anhang 20: WooCommerce Import GUI 1

**A screenshot of a computer

Description automatically generated**

Anhang 21: WooCommerce Import GUI 2

**A screenshot of a computer

Description automatically generated**

Anhang 22: WooCommerce Import GUI 3

**A screenshot of a computer

Description automatically generated**

Anhang 23: XLSX zu CSV

import pandas as pd

# this will reformat xlsx to an csv file

# it s well suited for german, and will respect leading zeros for article numbers

# .\_\_                 \_\_                        \_\_  .\_\_

# |\_\_| \_\_\_\_   \_\_\_\_\_\_\_/  |\_\_\_\_\_\_\_\_ \_\_ \_\_   \_\_\_\_\_/  |\_|\_\_| \_\_\_\_   \_\_\_\_   \_\_\_\_\_\_

# |  |/    \ /  \_\_\_/\   \_\_\\_  \_\_ \  |  \\_/ \_\_\_\   \_\_\  |/  \_ \ /    \ /  \_\_\_/

# |  |   |  \\\_\_\_ \  |  |  |  | \/  |  /\  \\_\_\_|  | |  (  <\_> )   |  \\\_\_\_ \

# |\_\_|\_\_\_|  /\_\_\_\_  > |\_\_|  |\_\_|  |\_\_\_\_/  \\_\_\_  >\_\_| |\_\_|\\_\_\_\_/|\_\_\_|  /\_\_\_\_  >

#         \/     \/                          \/                    \/     \/

# 1 put name of input File xlsx oder ods

# 2 put name of output csv file

# 1 Read the Excel file

df = pd.read\_excel('file\_name.xlsx', dtype=str) # Replace with your file name/path

# Keep only the 'Beschreibung' 'Kurzbeschreibung' columns or column needed

df = df[['Artikelnummer', 'Beschreibung', 'Kurzbeschreibung']]

# 3 Write to a CSV file

df.to\_csv('file\_name.csv', index=False, encoding='utf-8')

# Adjust the output file name as needed

Anhang 24: WooCommerce Fehler MeldungA screenshot of a computer

Description automatically generated