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

|  |
| --- |
| Titel  der Dokumentation |

|  |  |  |
| --- | --- | --- |
| Projektgeber: |  | Ausbildungsbetrieb: BFW Berlin-Brandenburg e. V. Kastanienallee 25 16567 Mühlenbeck  Ansprechpartner: Name Telefon: +4933056 862xxx  Mail: [xxx@bfw-berlin-brandenburg.de](mailto:xxx@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 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Inhaltsverzeichnis:

1. Projektdokumentationshinweise Verzeichnisse 1

1.1. Verwendungsvermerk 1

1.2. Dokumentation zur Projektarbeit 1

1.3. Abbildungsverzeichnis 1

1.4. Tabellenverzeichnis 1

1.5. Abkürzungen 1

1.6. Quellenverzeichnis 1

2. Planungsphase 1

2.1. Ausgangssituation 1

2.1.1. Projektumfeld 1

2.1.2. IST-Zustand 2

2.1.3. SOLL-Zustand 2

2.1.4. Projektziele 3

2.1.5. Personellen Schnittstellen 4

2.1.6. Technische Schnittstellen 4

2.1.7. Projektabgrenzung 4

2.1.8. Projektbegründung 4

2.1.9. Projektrisiken 4

2.2. Projektablauf 4

2.2.1. Technische Planung des Projektzieles 4

2.2.2. Analyse des IST-Zustandes 4

2.2.3. Erstellung des SOLL-Konzeptes 4

3. Ressourcen- und Ablaufplan 6

3.1. Personalplanung 6

3.2. Sachmittelplanung 6

3.3. Zeitliche Planung 6

3.4. Kostenplanung 6

4. Durchführungsphase 6

4.1. Eventuelle Beschaffung gemäß Planung 6

4.2. Herstellen der Arbeitsumgebung 6

4.3. Eigentlich Umsetzung 6

4.4. Testen 6

4.5. Fehler, Fehlerbehebung und Anpassung 6

5. Übergabe und Auswertung 6

5.1. Projektübergabe 6

5.2. Mitarbeiterschulung, Bedienungsanleitung 7

5.3. SOLL-IST-Vergleich (inhaltlich, zeitlich, kostentechnisch) 7

5.4. Technische Auswertung und Ausblick 7

5.5. Persönliche Auswertung 7

Deckblatt

# Projektdokumentationshinweise Verzeichnisse

Text test

## Verwendungsvermerk

Test text: darf man Sachen von der Firma benutzeb ist vorher zu klären

## Dokumentation zur Projektarbeit

ghfgzf

## Abbildungsverzeichnis

Uh7gz: alle bilder von der Dok

## Tabellenverzeichnis

alle Tabellen von der Dok vielleicht alle in Bilder

## Abkürzungen

nur abk schreiebn die in der Dok sind

## Quellenverzeichnis

nur wen welche komme, Ki Bilder generieren

# Planungsphase

## Ausgangssituation

### Projektumfeld

es genehmigt worden ist, dass ich Projekt der IHK-Prüfungskommission vorgelegt werden darf.

wer ist die firma wiveil leute was machen sie

Was macht die Firma

Projektgeber: Pixelding ist ein verlässlicher Partner in Sachen Web- und Grafikdesign aus der Medien-Hauptstadt Berlin:

Sie machen Webdesign, Online Shops, Sozial Media, Virtuelle Touren, Copyright Identity, Druckmedien, Photographie, Werbe Kampagnen

Wie groß ist die Firma:

Alex, der Boss, ein Programmierer, Philip, ein Grafiker, Anjal im Ausland, Praktikanten 4 – 5 für SEO und Grafik.

In welcher Abteilung wird das Projekt durchgeführt:

Online Shop, hauptsächlich Programmierung, Teil vom Projekt an SEO und Grafik Team delegiert

Mitarbeiter und Vorgesetze mit denen Sie interagieren im Projekt:

SEO-Team, Grafiker, Boss

KundenFirma: XXXXXXXX (ich frage noch, ob ich den namens benutzen darf) Ist im Verkauf tätig und will sein Onlinepräsenz verbessern, bzw. mehr verkaufen

### 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

das koperien von Antrag

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

soll zu stand als Stichpunkte

Der Ablauf beinhaltet die Nutzung von Python zur Datenextraktion 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 Grafikerteam 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

Die Menschen mit der wir im Unterhenm sind nennen oder sagen er möchte nicht bennant werden nicht mehr als 2. ein reicht aus bzw der Auftraggeber

Alex, der big boss (ich frage noch, ob ich ihn nennen darf)

Der Herr x, der Kunde (ich frage noch, ob ich ihn nennen darf)

Seo Team, Grafiker: 2 3 Leute

### Technische Schnittstellen

könne muessen nicht sein, wenn nicht nötig server und was fuer den Project wichtig istbzw daten bank die benutz wird oder bereits exoistiert, api aus dem internet,

### Projektabgrenzung

ist der project allein stehen punkte weg lassen  
wenn Project ein Teil Projekt von einem Grösserem Project

### Projektbegründung

unterschied zwishen ist und soll

### Projektrisiken

Kranksein? mit ansprechpartenr darueber sprechen, etwas nicht rechtzeitig geliefert

## Projektablauf

### Technische Planung des Projektzieles

### Analyse des IST-Zustandes

### Erstellung des SOLL-Konzeptes

|  |  |
| --- | --- |
| **Technische Analyse des Ist-Zustandes:** | 5 |
| Untersuchung des aktuellen manuellen Prozesses zur Integration neuer Produkte. | 2 |
| Evaluierung der bestehenden Datenextraktionsmethoden und -werkzeuge. | 2 |
| Meeting | 1 |
| **Herstellen der Arbeitsumgebung:** | **3** |
| Aufsetzen einer geeigneten Entwicklungsumgebung, einschließlich der erforderlichen Softwaretools wie Python.  BeautifulSoup or Selenium, Xampp | 3 |
| **Datenextraktion von 4 Webseiten:** | **31** |
| Python Script schrieben | 10 |
| Durchführung von Modultests auf die Datenextraktion mittels Python. | 5 |
| Durchführung eines Systemtests, um die gesamte Datenextraktionspipeline zu überprüfen. | 5 |
| Script anpassen und herunterladen | 9 |
| Sicherung auf Google Drive sowie auf zwei separaten Festplatten für zusätzliche Redundanz. | 1 |
| Meeting | 1 |
| **Datenkonvertierung für WooCommerce:** | **11** |
| Umwandlung der JSON-Daten in das erforderliche WooCommerce CSV-Format mittels Python. | 7 |
| WordPress local shop installieren | 2 |
| test | 1 |
| Hochladen der Dateien auf den Test-WooCommerce-Shop im lokalen Host. | 1 |
| **XLSX-Liste der Produktfotos für das Grafikerteam:** | **1** |
| Bereitstellung einer XLSX-Liste der Produktfotos zur eingehenden Prüfung. | 1 |
| **XLSX-Liste der Texten für das SEO-Team:** | **1** |
| Bereitstellung einer XLSX-Liste für das SEO-Team zur Überarbeitung und Ergänzung von Texten. | 1 |
| **Erstellung einer Excel-Tabelle für den Kunden:** | **1** |
| Erstellung einer Excel-Tabelle zur erleichterten Verwaltung von Preisen und Lagerbeständen durch den Kunden. | 1 |
| **Implementierung eines Systems für die Datenrückkonvertierung:** | **2** |
| Implementierung eines Systems zur Konvertierung der bearbeiteten Excel-Dateien zurück in das WooCommerce CSV-Format. | 2 |
| **Lokale Tests auf WordPress:** | **7** |
| Hochladen der CSV-Dateien lokal auf WordPress | 1 |
| umfassende Tests. | 6 |
| **Export und Übertragung in den Online-Shop des Kunden:** | **2** |
| Export der CSV-Dateien vom lokalen Shop und Übertragung in den Online-Shop des Kunden. | 2 |
| **Erneute Tests zur Sicherstellung eines reibungslosen Ablaufs:** | 6 |
| Durchführung erneuter Tests, um die ordnungsgemäße Funktion sicherzustellen. | 6 |
| **Dokumentation von Projektdetails, Problemen und Fortschritten:** | **10** |
| Dokumentation sämtlicher Projektdetails, auftretender Probleme und erzielter Fortschritte in Asana und Google Drive. | 9 |
| Meeting | 1 |
| **Summe** | **80** |

# Ressourcen- und Ablaufplan

## Personalplanung

nur ich

## Sachmittelplanung

was gebraucht wird

## Zeitliche Planung

erst mal leer lassen

## Kostenplanung

was ich verdiene im echten leben pro Stunde \*80 bei meeting schaeten was der ansprechpartenr kostet energie pro schal 6 euro am Tag home office

# Durchführungsphase

## Eventuelle Beschaffung gemäß Planung

## Herstellen der Arbeitsumgebung

wi ich mich zurecht finde und arbeitumbgebung fertig mache

## Eigentlich Umsetzung

ich brauch hardware uich instaliere das welche daten bak ich da brauche

## Testen

ganz wichtig

## Fehler, Fehlerbehebung und Anpassung

wenn kein Problem weg lassen

# Übergabe und Auswertung

## Projektübergabe

wie das project abgegeben wurde

## Mitarbeiterschulung, Bedienungsanleitung

?

## SOLL-IST-Vergleich (inhaltlich, zeitlich, kostentechnisch)

wieveil zeit fuer jeden punkt eine ganze seite

## Technische Auswertung und Ausblick

was hat mir das gebracht?  
ausblcik was passiert mit diesem Programm jetzt?

## Persönliche Auswertung

Wie es war fuer mich

Max zeilen abstand 1,5

**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 definiertere Kommunikation über z.B. Schnittstellen? was kann ich dazu sagen??**

NUR WEIL ICH DAS KANN, Preis von der Lizenz und Zeit zu lernen Webharvey oder Visual Web Ripper, ist es am Ende nicht günstiger.

Wenn ich selber programmiere, ist es lighter, ich habe einen gutem Überblick und wenn es nicht geht bin ich auch dran selber schuld. Die Seiten sind auch schlecht gebaut und es ist viel anzupassen damit es am Ende klappt

Für die Datenextraktion von den Webseiten der Lieferanten wurde sich entschieden, weil diese Methode mehr Flexibilität und Kontrolle bietet. Hier sind einige Gründe, warum dieser Weg gewählt wurde und nicht eine direkte Kommunikation über Schnittstellen:

1. **Datenunabhängigkeit**: Durch die Extraktion von Daten direkt von den Webseiten der Lieferanten sind wir nicht auf spezifische Schnittstellen oder APIs angewiesen. Dies ermöglicht es uns, Daten von einer Vielzahl von Lieferanten zu extrahieren, unabhängig von ihren jeweiligen Systemen oder APIs.
2. **Anpassungsfähigkeit**: Webseitenstrukturen können sich im Laufe der Zeit ändern, und nicht alle Lieferanten bieten robuste APIs oder Schnittstellen an. Durch die direkte Extraktion von Daten können wir flexibel auf Veränderungen in der Webseite reagieren und unsere Extraktionslogik entsprechend anpassen.
3. **Kosteneffizienz**: Die Entwicklung und Wartung von Schnittstellen oder APIs kann zeit- und ressourcenintensiv sein. Durch die Nutzung vorhandener Webseiteninhalte können wir Kosten reduzieren und die Implementierungszeit verkürzen.
4. **Granularität der Daten**: Oftmals bieten Schnittstellen oder APIs nur begrenzte Daten an, während wir durch die Extraktion von Webseiteninhalten Zugriff auf eine breitere Palette von Informationen haben können, die für unseren Zweck relevant sind.
5. **Zugänglichkeit**: Einige Lieferanten verfügen möglicherweise nicht über Schnittstellen oder APIs, die für externe Entwickler zugänglich sind. In solchen Fällen ist die direkte Extraktion von Webseiteninhalten oft die einzige praktikable Lösung.

Insgesamt bietet die Extraktion von Daten von den Webseiten der Lieferanten eine flexible, anpassungsfähige und kosteneffiziente Lösung, die es uns ermöglicht, die benötigten Daten zu erhalten, unabhängig von den spezifischen Gegebenheiten der einzelnen Lieferanten.

L'extraction de données à partir des sites web des fournisseurs a été choisie parce que cette méthode offre plus de flexibilité et de contrôle. Voici quelques raisons pour lesquelles nous avons choisi cette méthode plutôt qu'une communication directe via des interfaces :

1) Indépendance des données : en extrayant les données directement des sites web des fournisseurs, nous ne dépendons pas d'interfaces ou d'API spécifiques. Cela nous permet d'extraire des données d'un grand nombre de fournisseurs, indépendamment de leurs systèmes ou API respectifs.

2) Adaptabilité : les structures des sites web peuvent évoluer au fil du temps et tous les fournisseurs ne proposent pas des API ou des interfaces robustes. En extrayant directement les données, nous pouvons réagir de manière flexible aux modifications du site web et adapter notre logique d'extraction en conséquence.

3. rentabilité : le développement et la maintenance d'interfaces ou d'API peuvent demander beaucoup de temps et de ressources. En utilisant le contenu existant du site web, nous pouvons réduire les coûts et raccourcir le temps de mise en œuvre.

4. granularité des données : Souvent, les interfaces ou les API ne fournissent que des données limitées, alors qu'en extrayant le contenu des pages web, nous pouvons avoir accès à un plus large éventail d'informations pertinentes pour notre objectif.

5) Accessibilité : certains fournisseurs peuvent ne pas disposer d'interfaces ou d'API accessibles aux développeurs externes. Dans de tels cas, l'extraction directe du contenu des pages web est souvent la seule solution viable.

Globalement, l'extraction de données à partir des pages web des fournisseurs offre une solution flexible, adaptable et rentable qui nous permet d'obtenir les données dont nous avons besoin, indépendamment des spécificités de chaque fournisseur.

**wie hätte es sonst funktioniert mit einer definierten Kommunikation über z.B. Schnittstellen? kannst erklaeren?**

Wenn eine definierte Kommunikation über Schnittstellen oder APIs gewählt worden wäre, hätte das bedeutet, dass wir direkt mit den Systemen der Lieferanten kommunizieren würden, um die benötigten Daten abzurufen. Hier sind einige Schritte, wie es funktionieren könnte:

1. **Schnittstellenintegration**: Zunächst müssten wir mit jedem Lieferanten individuell Schnittstellen oder APIs integrieren. Dies könnte bedeuten, dass wir uns mit verschiedenen Systemen und Datenformaten auseinandersetzen müssen, je nach den Systemen, die von den Lieferanten verwendet werden.
2. **Authentifizierung und Autorisierung**: Wir müssten sicherstellen, dass wir die erforderlichen Zugriffsberechtigungen haben, um auf die Daten der Lieferanten zuzugreifen. Dies könnte die Implementierung von Authentifizierungsmechanismen wie API-Schlüsseln oder OAuth umfassen.
3. **Datenabfrage**: Nachdem die Integration abgeschlossen ist, könnten wir über die Schnittstellen oder APIs Anfragen an die Systeme der Lieferanten senden, um die benötigten Daten abzurufen. Dies würde normalerweise die Definition spezifischer Abfragen oder Endpunkte für verschiedene Datentypen umfassen.
4. **Datenverarbeitung**: Nachdem die Daten abgerufen wurden, müssten wir sie möglicherweise transformieren oder anpassen, um sie in das gewünschte Format für unsere Zwecke zu bringen. Dies könnte das Mapping von Feldern, die Konsolidierung von Daten aus verschiedenen Quellen oder die Bereinigung von Daten umfassen.
5. **Regelmäßige Aktualisierung**: Da sich Daten kontinuierlich ändern können, müssten wir sicherstellen, dass unsere Integrationen regelmäßig aktualisiert werden, um die neuesten Daten von den Lieferanten zu erhalten. Dies könnte die Implementierung von automatisierten Aktualisierungsprozessen oder Zeitplänen umfassen.

*Obwohl die Verwendung von Schnittstellen oder APIs eine standardisierte und möglicherweise robustere Lösung sein kann, kann sie auch mit einigen Herausforderungen verbunden sein, insbesondere wenn die Lieferanten unterschiedliche Systeme verwenden oder keine umfassenden Schnittstellen anbieten. Die direkte Extraktion von Webseiteninhalten bietet in solchen Fällen oft eine flexiblere und kosteneffizientere Lösung.*

comment cela aurait-il fonctionné avec une communication définie via des interfaces par exemple ? tu peux expliquer ?

Si une communication définie via des interfaces ou des API avait été choisie, cela aurait signifié que nous aurions communiqué directement avec les systèmes des fournisseurs pour récupérer les données nécessaires. Voici quelques étapes sur la manière dont cela pourrait fonctionner :

1. intégration des interfaces : nous devrions d'abord intégrer individuellement des interfaces ou des API avec chaque fournisseur. Cela pourrait signifier que nous devons nous occuper de différents systèmes et formats de données, en fonction des systèmes utilisés par les fournisseurs.

2) Authentification et autorisation : nous devrions nous assurer que nous disposons des droits d'accès nécessaires pour accéder aux données des fournisseurs. Cela pourrait impliquer la mise en œuvre de mécanismes d'authentification tels que les clés API ou OAuth.

3) Requête de données : une fois l'intégration terminée, nous pourrions envoyer des requêtes aux systèmes des fournisseurs via les interfaces ou les API afin de récupérer les données requises. Cela impliquerait normalement la définition de requêtes ou de points de terminaison spécifiques pour différents types de données.

4) Traitement des données : une fois que les données ont été récupérées, nous pourrions avoir besoin de les transformer ou de les adapter afin de les mettre dans le format souhaité pour nos besoins. Cela pourrait inclure le mappage des champs, la consolidation des données provenant de différentes sources ou le nettoyage des données.

5) Mise à jour régulière : étant donné que les données peuvent changer en permanence, nous devrions veiller à ce que nos intégrations soient régulièrement mises à jour afin d'obtenir les données les plus récentes auprès des fournisseurs. Cela pourrait inclure la mise en œuvre de processus de mise à jour automatisés ou de calendriers.

Bien que l'utilisation d'interfaces ou d'API puisse être une solution standardisée et potentiellement plus robuste, elle peut aussi présenter certains défis, notamment si les fournisseurs utilisent des systèmes différents ou ne proposent pas d'interfaces complètes. Dans de tels cas, l'extraction directe du contenu des pages web offre souvent une solution plus flexible et plus rentable.

In deiner Dokumentation könntest du die Entscheidung für die Datenextraktion von den Webseiten der Lieferanten detailliert erklären. Hier sind einige Gründe, die du erwähnen könntest:

1. **Flexibilität und Skalierbarkeit**: Die Extraktion von Daten von den Webseiten der Lieferanten ermöglicht es, flexibel auf Änderungen in der Struktur oder dem Design der Webseiten zu reagieren. Wenn sich das Layout der Webseiten ändert, kann das Extraktionsprogramm entsprechend angepasst werden, um weiterhin zuverlässig Daten zu extrahieren.
2. **Unabhängigkeit von Schnittstellen oder APIs**: Einige Lieferanten bieten möglicherweise keine öffentlichen APIs oder Schnittstellen an, über die Daten abgerufen werden können. Durch die Extraktion von Daten direkt von den Webseiten umgeht man diese Einschränkungen und erhält dennoch Zugriff auf die benötigten Informationen.
3. **Vollständiger Zugriff auf verfügbare Daten**: Durch das Web-Scraping kann man auf eine breitere Palette von Informationen zugreifen, die möglicherweise nicht über Schnittstellen verfügbar sind. Dies könnte zusätzliche Details zu Produkten, Preisen oder anderen relevanten Informationen umfassen.
4. **Kostenersparnis**: Die Entwicklung und Wartung von Schnittstellen oder APIs kann kostenintensiv sein. Durch die Verwendung von Web-Scraping-Techniken kann man Kosten reduzieren und dennoch die benötigten Daten extrahieren.
5. **Zeitersparnis**: Die direkte Extraktion von Daten von den Webseiten der Lieferanten kann oft schneller sein als die Integration mit verschiedenen Schnittstellen oder APIs. Dies ermöglicht es, den Entwicklungsprozess zu beschleunigen und schneller Ergebnisse zu erzielen.

In deiner Dokumentation könntest du diese Gründe ausführlich erläutern und möglicherweise auch auf spezifische Herausforderungen oder Vorteile eingehen, die sich aus der Wahl dieser Methode ergeben haben. Es könnte auch hilfreich sein, Beispiele oder Fallstudien zu verwenden, um die Entscheidung zu veranschaulichen und zu untermauern.

Einleitender Satz: Schnittstellen gelten oft als bevorzugte Methode für den Datenaustausch. Im ersten Abschnitt werde ich erörtern, warum sie eine attraktive Option darstellen. Anschließend werde ich auf potenzielle Schwierigkeiten eingehen, bevor ich abschließend meine eigene Meinung dazu darlege.

Erster Absatz (Was dafür spricht): Die Nutzung definierter Schnittstellen oder APIs hätte bedeutet, direkt mit den Systemen der Lieferanten zu kommunizieren, um die benötigten Daten abzurufen. Doch dieser Ansatz birgt einige Herausforderungen. Zunächst müssten individuelle Schnittstellen oder APIs für jeden Lieferanten integriert werden, was einen erheblichen Aufwand bedeuten würde. Zudem wären Authentifizierung und Autorisierung notwendig, um Zugriff auf die Daten zu erhalten. Auch die Definition spezifischer Abfragen oder Endpunkte sowie die regelmäßige Aktualisierung der Integrationen wären erforderlich.

Zweiter Absatz (Was dagegenspricht): Trotz der potenziellen Vorteile sind Schnittstellenintegrationen oft komplex und zeitaufwändig. Die Abhängigkeit von den Systemen der Lieferanten könnte zu Schwierigkeiten führen, besonders wenn diese Systeme nicht standardisiert sind oder sich häufig ändern. Zudem könnten Sicherheitsrisiken bestehen, wenn sensible Daten über externe Schnittstellen zugänglich sind. Auch könnten Schnittstellenbeschränkungen oder Gebühren seitens der Lieferanten auftreten, die die Implementierung und den Betrieb der Integrationen beeinträchtigen könnten.

Dritter Absatz (Eigene Meinung): Angesichts der genannten Vor- und Nachteile erscheint die Entscheidung, Daten von den Webseiten der Lieferanten mit Python und den Bibliotheken BeautifulSoup und Selenium zu extrahieren, als eine praktikablere Lösung. Neben der Flexibilität bei der Anpassung an Veränderungen in den Webseitenstrukturen und der schnelleren Implementierung können wir auch erhebliche Kosteneinsparungen erzielen. Die Eigenentwicklung ermöglicht zudem eine effizientere Fehlerbehebung, da es einfacher ist, den eigenen Code zu korrigieren. Die Lernzeit für die Verwendung von BeautifulSoup und Selenium ist im Vergleich zu anderen Lösungen ebenfalls gering, und das Webscraping mit diesen Bibliotheken stellt keine große Herausforderung dar. Im Gegensatz dazu erfordert die Einarbeitung in spezialisierte Programme wie WebHarvy zusätzliche Zeit und finanzielle Ressourcen, was die Gesamtkosten des Projekts erhöhen könnte.

Schlusssatz: Insgesamt bietet die Extraktion von Daten von den Webseiten der Lieferanten mit Python eine robuste und kostengünstige Lösung für die Herausforderungen der Produktintegration in unseren Online-Shop.

1. **Flexibilität und Anpassungsfähigkeit:** Die Nutzung von Python und Bibliotheken wie BeautifulSoup und Selenium ermöglicht eine flexible und anpassungsfähige Datenextraktion. Durch das direkte Scraping der Webseiteninhalte können Änderungen in der Struktur der Webseiten leichter bewältigt und die Extraktionslogik entsprechend angepasst werden. Im Gegensatz dazu erfordert die Verwendung von Schnittstellen oder APIs oft spezifische Anpassungen für jeden Lieferanten, was zeitaufwändig und weniger flexibel sein kann.
2. **Kosteneffizienz:** Die Entwicklung eigener Skripte zur Datenextraktion mit Python kann kosteneffizienter sein als die Integration von Schnittstellen oder APIs. Die Verwendung von Python und Open-Source-Bibliotheken erfordert keine Lizenzgebühren und kann die Gesamtkosten des Projekts senken.
3. **Sicherheit und Zugänglichkeit:** Bei der direkten Datenextraktion mit Python haben wir direkten Zugriff auf die gewünschten Daten, ohne auf die Verfügbarkeit oder Zugänglichkeit von Schnittstellen oder APIs angewiesen zu sein. Dies kann besonders wichtig sein, wenn die Lieferanten keine oder nur eingeschränkte Schnittstellen zur Verfügung stellen.
4. **Kontrolle und Selbstbestimmung:** Durch die Entwicklung eigener Skripte haben wir die volle Kontrolle über den Extraktionsprozess. Wir sind nicht von den Einschränkungen oder Änderungen in den Schnittstellen oder APIs der Lieferanten abhängig. Bei Problemen oder Änderungen können wir schnell reagieren und den Code entsprechend anpassen.

**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 definiertere Kommunikation über z.B. Schnittstellen? was kann ich dazu sagen??**

Intro: Eine präzisere Kommunikation über Schnittstellen oder APIs stellt oft eine verlockende Option für die Extraktion von Daten von Webseiten dar. Im ersten Abschnitt werden wir erkunden, was diesen Ansatz besonders attraktiv macht. Anschließend werden wir die potenziellen Herausforderungen betrachten, die sich dabei ergeben können. Im dritten Abschnitt teile ich meine persönlichen Erfahrungen und Einsichten zu diesem Thema.

Vorteile einer definierteren Kommunikation über Schnittstellen: Die Nutzung von Schnittstellen oder APIs bietet klare Vorteile. Erstens ermöglichen sie eine standardisierte und strukturierte Datenübertragung, wodurch die Integration mit den Systemen der Lieferanten erheblich vereinfacht wird. Zweitens kann die direkte Anbindung an die Systeme der Lieferanten die Aktualität und Zuverlässigkeit der Daten verbessern, da sie in Echtzeit abgerufen werden können. Drittens ermöglicht die Verwendung von Schnittstellen eine effizientere und automatisierte Datenübertragung, was Zeit und Ressourcen spart. Darüber hinaus erfordert dieser Ansatz keine Programmierung und kann schnell implementiert werden. Zudem stehen dem Kunden Online-Hilfe und Support zur Verfügung, um den Integrationsprozess weiter zu erleichtern.

Nachteile einer definierteren Kommunikation über Schnittstellen: Trotz der Vorteile können bei der Verwendung von Schnittstellen auch potenzielle Nachteile auftreten. Erstens kann die Integration mit verschiedenen Lieferanten aufgrund der Notwendigkeit individueller Anpassungen für jede Schnittstelle komplex und zeitaufwändig sein. Zweitens sind Schnittstellen möglicherweise nicht immer verfügbar oder bieten nicht alle benötigten Daten an, was zu Lücken in den Produktinformationen führen kann. Darüber hinaus erfordert die Nutzung von Schnittstellen auch Zeit und Ressourcen, um die Bedienung zu erlernen, geeignete Schnittstellen zu finden und diese zu testen, was den Integrationsprozess weiter verkomplizieren kann. Zudem besteht eine Abhängigkeit vom Kundenservice, wenn Probleme auftreten, was die Effizienz und den Betrieb der Integrationen beeinträchtigen kann.

Meine eigene Meinung: Ich sehe die Verwendung von Python und Web-Scraping-Bibliotheken wie BeautifulSoup und Selenium als eine effektive und kostengünstige Alternative zur Nutzung von Schnittstellen. Diese Methode ermöglicht eine flexible und anpassungsfähige Datenextraktion direkt von den Webseiten der Lieferanten. Durch die Entwicklung eigener Skripte habe ich die Kontrolle über den Extraktionsprozess und kann schnell auf Änderungen oder Probleme reagieren. Darüber hinaus können wir Kosten senken und eine effiziente Datenintegration sicherstellen. Datenextraktion mittels Web-Scraping stellt keine große Herausforderung dar, und selbst wenn es einmal schwierig wird, bietet es die Möglichkeit, neue Fähigkeiten zu erlernen und den eigenen Wissensschatz zu erweitern. Und es macht unheimlich viel Spaß.

Schlussatz: Insgesamt bieten sowohl die Nutzung definierter Schnittstellen als auch die Datenextraktion mit Python verschiedene Vor- und Nachteile. Die Wahl des geeigneten Ansatzes hängt von den spezifischen Anforderungen des Projekts sowie von Faktoren wie Kosten, Flexibilität und Kontrolle, und persönlichen Präferenzen ab. Eine kontinuierliche Bewertung und Anpassung dieser Methoden im Verlauf des Projekts ist jedoch entscheidend, um sicherzustellen, dass sie den sich ändernden Anforderungen und Gegebenheiten gerecht werden. Ich bin mit der Nutzung von Python für diesen Projekt sehr zufrieden.

**Aktikitaets diagram**

**Netzplan**

Hier sind einige mögliche Interpretationen von "Schnittstellen" in deinem Kontext:

Web Scraping: Die Schnittstelle zwischen deinem Programm und den Webseiten der Lieferanten. Du könntest erklären, wie dein Programm die Webseiten analysiert, um Informationen zu extrahieren.

Dateiübertragung: Die Schnittstelle zwischen deinem Programm und den Dateispeichern (z.B. Google Drive, lokale Festplatten). Du könntest beschreiben, wie dein Programm Daten speichert und abruft.

Datenkonvertierung: Die Schnittstelle zwischen den verschiedenen Datenformaten (z.B. JSON, CSV). Du könntest erklären, wie dein Programm Daten in das richtige Format konvertiert.

Kommunikation mit anderen Teams: Die Schnittstelle zwischen deinem Programm und anderen Teams, z.B. dem Grafikdesign-Team oder dem SEO-Team. Du könntest beschreiben, wie dein Programm Daten austauscht oder Informationen bereitstellt.

In deinem Dokument könntest du also unter dem Punkt "Schnittstellen" genau erklären, wie dein Programm mit diesen verschiedenen Aspekten interagiert und wie die Kommunikation erfolgt. Dies könnte beinhalten, wie Daten gesammelt, gespeichert, verarbeitet und weitergegeben werden.

WebHarvy ist eine weitere Software, die für das Web-Scraping verwendet werden kann. Es ist eine Desktop-Anwendung, die speziell für das Extrahieren von Daten aus Webseiten entwickelt wurde. Mit WebHarvy kannst du einfach die gewünschten Webseiten aufrufen, die Daten selektieren, die du extrahieren möchtest, und dann die extrahierten Daten in einem gewünschten Format speichern, z.B. als CSV- oder Excel-Datei.

Der Hauptvorteil von WebHarvy liegt in seiner Benutzerfreundlichkeit und seiner Fähigkeit, Daten aus komplexen Webseiten ohne Programmierung zu extrahieren. Es bietet eine intuitive Benutzeroberfläche, die es auch weniger technisch versierten Benutzern ermöglicht, Web-Scraping-Aufgaben durchzuführen.

Wenn dir WebHarvy angeboten wurde, könnte es eine gute Option sein, besonders wenn du keine Programmierkenntnisse hast oder eine schnellere Lösung wünschst. Allerdings solltest du beachten, dass Desktop-Anwendungen wie WebHarvy normalerweise eine einmalige Lizenzgebühr erfordern und möglicherweise nicht so flexibel oder anpassbar sind wie Python-basierte Lösungen mit Bibliotheken wie BeautifulSoup und Selenium.

Top of Form

Testing: Die And retry

Wenn das Programm beim ersten Mal schnell und gut läuft, ist es sicher ein gravierender Fehler drinen.

Module testing: nur einzeln elemnte testen

Global testing: das ganze program laufen lassen aber berentzt auf nur ein paar Seiten

Global global Testing : das ganze Program laufen

Neben von der json schreibt python ein protokole was gemacht wurde

White box test, black box tests

Protokoll sieht so aus:

1 url from 2 main urls

https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten.html

getMainSnippet

- url to open https://www.noelle-profi-brush.de/produkte/fussmatten.html

- - url\_to\_find https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten.html

- opening 1 from 1 https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten.html

...........Links.......

...........Links.......

...........Links.......

...........Links.......

...........Links.......

...........Links.......

...........Links.......

- FOUND Links: 1

- - https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten.html

- - - FOUND Produkts pages: 7

- - - - https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten/produktdetails/allwetter-grasmatte-911.html

- - - - https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten/produktdetails/gummi-ringmatte-941.html

- - - - https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten/produktdetails/gummi-wabenmatte-922.html

- - - - https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten/produktdetails/kokos-drahtgittermatte-1024.html

- - - - https://www.noelle-profi-brush.de/produkte/fussmatten/fussmatten/produktdetails/kokos-veloursmatte-1488.html

* **Selenium**: Selenium is a web browser automation tool that allows you to control web browsers programmatically. It is often used for tasks such as filling out web forms, clicking buttons, and navigating through web pages. Selenium can interact with dynamic web content and JavaScript-heavy websites, making it a powerful tool for web scraping. It simulates user interaction with the web page, allowing you to extract data as if you were using a web browser yourself.
* **BeautifulSoup**: BeautifulSoup is a Python library for parsing HTML and XML documents. It provides simple methods and Pythonic idioms for navigating, searching, and modifying the parse tree. BeautifulSoup is often used in conjunction with other libraries like Requests to scrape data from web pages. While BeautifulSoup itself does not interact with web browsers like Selenium does, it is very effective for extracting data from static web pages and parsing the HTML structure.

**WordPress**: WordPress ist eine Open-Source-Software, die ursprünglich als Blogging-Plattform entwickelt wurde, sich aber im Laufe der Zeit zu einem leistungsstarken Content-Management-System (CMS) entwickelt hat. Es wird weltweit von Millionen von Websites verwendet und bietet eine benutzerfreundliche Oberfläche sowie eine Vielzahl von Plugins und Themes für die Erstellung verschiedener Arten von Websites, darunter Blogs, Unternehmenswebsites, Online-Shops und mehr.

**WooCommerce**: WooCommerce ist ein beliebtes E-Commerce-Plugin für WordPress. Es ermöglicht es Benutzern, schnell und einfach einen Online-Shop zu erstellen und zu verwalten, indem es Funktionen wie Produktlisten, Warenkörbe, Kassenseiten und Zahlungsabwicklung integriert. WooCommerce bietet eine Vielzahl von Erweiterungen und Themes, mit denen Benutzer ihren Online-Shop an ihre individuellen Bedürfnisse anpassen können. Es ist eine kostengünstige und skalierbare Lösung für kleine bis mittlere Unternehmen, die einen Online-Verkaufskanal aufbauen möchten.

**WooCommerce Germanized:** WooCommerce Germanized ist eine Erweiterung für WooCommerce, die speziell für den deutschen Markt entwickelt wurde. Sie erleichtert die Einhaltung deutscher Gesetze und Richtlinien für den Online-Handel, einschließlich des deutschen Datenschutzrechts (DSGVO), des Widerrufsrechts und der Impressumspflicht. WooCommerce Germanized bietet Funktionen wie eine optimierte Checkout-Seite für deutsche Kunden, die Integration von Rechtstexten und die Anpassung von E-Mail-Benachrichtigungen an deutsche Rechtsvorschriften. Es ist eine wertvolle Erweiterung für deutsche Online-Händler, die sicherstellen möchten, dass ihr Online-Shop den lokalen gesetzlichen Anforderungen entspricht.

**Woocommerce API**: un produkte zu loeschen auf dem Local host

**Wooocommerce info page from git**

Logo erlaubnis:

wordpress,

woocomerce,

**WooCommerce Germanized**

python,

visualstudio code

pixelding

putzmittelshop24

phpmyadmin

bfw

wordpress gui ist sooo langsam

<https://codex.wordpress.org/Database_Description>

A computer screen shot of a computer

Description automatically generated

By default, WordPress uses the MyISAM storage engine for database tables, which does not support foreign keys or enforce referential integrity. MyISAM tables are optimized for read-heavy workloads and offer better performance for certain types of queries.

However, while foreign keys are not enforced at the database level in WordPress, developers can still implement referential integrity and establish relationships between tables at the application level. This means that developers need to manually manage relationships between tables and ensure data consistency through their WordPress plugins, themes, or custom code.

For example, if you're creating a custom plugin that extends WordPress functionality and involves multiple database tables with relationships, you would need to write code to handle data validation, insertion, and updates in a way that maintains referential integrity.

Alternatively, developers can choose to use the InnoDB storage engine for their WordPress database tables, which does support foreign keys and referential integrity. InnoDB tables provide stronger data consistency guarantees compared to MyISAM tables, but they may come with a slight performance overhead, especially for write-heavy workloads.

Ultimately, the choice between MyISAM and InnoDB depends on the specific requirements and performance characteristics of your WordPress application, as well as your preference for data consistency versus performance.

WordPress installation doesn’t enforce referential integrity. WordPress uses the MyISAM storage engine, which gives more functionality and better performance but does not support foreign keys.

ChatGPT

**ChatGPT**

Yes, that's correct. By default, WordPress uses the MyISAM storage engine for database tables, which does not support foreign keys or enforce referential integrity. MyISAM tables are optimized for read-heavy workloads and offer better performance for certain types of queries.

However, while foreign keys are not enforced at the database level in WordPress, developers can still implement referential integrity and establish relationships between tables at the application level. This means that developers need to manually manage relationships between tables and ensure data consistency through their WordPress plugins, themes, or custom code.

For example, if you're creating a custom plugin that extends WordPress functionality and involves multiple database tables with relationships, you would need to write code to handle data validation, insertion, and updates in a way that maintains referential integrity.

Alternatively, developers can choose to use the InnoDB storage engine for their WordPress database tables, which does support foreign keys and referential integrity. InnoDB tables provide stronger data consistency guarantees compared to MyISAM tables, but they may come with a slight performance overhead, especially for write-heavy workloads.

Ultimately, the choice between MyISAM and InnoDB depends on the specific requirements and performance characteristics of your WordPress application, as well as your preference for data consistency versus performance.

https://webappick.com/woocommerce-database-schema-explained/

### Find WooCommerce Products in the WP Database

[WooCommerce product data](https://woocommerce.com/document/bundles/bundles-data-structures-storage/) can be found in the following tables:

* wc\_product\_meta\_lookup (product ID or product\_variation ID)
* wc\_order\_product\_lookup (product ID or product\_variation ID)
* posts (product ID or product\_variation ID)
* postmeta (product or product\_variation)
* posts (product ID or product\_variation ID)
* wc\_product\_meta\_lookup (product ID or product\_variation ID)

A screenshot of a computer

Description automatically generated

* wc\_order\_product\_lookup (product ID or product\_variation ID)
* A screenshot of a computer

  Description automatically generated
* posts (product ID or product\_variation ID)
* postmeta (product or product\_variation)
* posts (product ID or product\_variation ID)

https://www.webhat.in/article/woocommerce-tutorial/decoding-product-data-part-1-of-the-woocommerce-db-series/

A computer screen shot of a computer

Description automatically generated

Anhang