

# **Konzeption und Entwicklung einer JTL-Wawi Zusatzsoftware sowie Schnittstelle zur Übersetzung von Texten**

Bericht zur Projektarbeit zur Erlangung des Abschlusses als  
**Fachinformatiker Fachrichtung  
Anwendungsentwicklung**

Arbeit von  
**Quirin Langer**

Betrieb: RIS Web- & Software-Development GmbH & Co. KG

E-Mail: [langner@ris-development.de](mailto:langner@ris-development.de)

Telefon: +49 941 2000 1250

Prüfungsnummer: 95230

Ausführungszeit: 19.04.2021 - 21.05.2021

Projektbetreuer: Christoph Wagner

E-Mail: [wagner@ris-development.de](mailto:wagner@ris-development.de)

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
1.1	Projektumfeld . . . . .	4
1.2	Ist-Analyse . . . . .	4
1.3	Soll-Zustand . . . . .	4
1.4	Lösungsansätze . . . . .	5
1.5	Technologieplattform . . . . .	6
<b>2</b>	<b>Entwurfsphase</b>	<b>7</b>
2.1	Schnittstelle . . . . .	7
2.1.1	Datenbankmodell . . . . .	7
2.1.2	Aufbau . . . . .	7
2.2	Zusatzsoftware . . . . .	8
2.2.1	Datenbankmodell . . . . .	8
2.2.2	Aufbau . . . . .	8
<b>3</b>	<b>Implementierung</b>	<b>9</b>
3.1	Schnittstelle . . . . .	9
3.1.1	Datenbanktabellen/Migrations . . . . .	9
3.1.2	Entitätstypen/Models . . . . .	9
3.1.3	Controller . . . . .	9
3.1.4	Services . . . . .	10
3.2	Zusatzsoftware . . . . .	10
3.2.1	Konfiguration . . . . .	10
3.2.2	SqlHelper Klasse . . . . .	10
3.2.3	AntHelper Klasse . . . . .	11
3.2.4	CsvHelper und ApiHelper Klasse . . . . .	11
<b>4</b>	<b>Testphase</b>	<b>12</b>
4.1	Schnittstelle . . . . .	12
4.2	Zusatzsoftware . . . . .	12
<b>5</b>	<b>Schluss</b>	<b>13</b>
5.1	Soll-Ist-Vergleich . . . . .	13
5.2	Dokumentation und Abnahme . . . . .	14
5.3	Wirtschaftlichkeitsanalyse . . . . .	14
5.3.1	Kosten . . . . .	14
5.3.2	Einnahmen . . . . .	15

5.3.3	Fazit . . . . .	15
5.4	Ausblick . . . . .	16
<b>6</b>	<b>Glossar</b>	<b>17</b>
<b>7</b>	<b>Anhang</b>	<b>19</b>
A	Schnittstelle: ER-Diagramm . . . . .	19
B	Liste der HTTP-Endpunkte . . . . .	19
C	Tabelle 'translation_requests' . . . . .	19
D	Tabelle 'customers' . . . . .	20
E	Zusatzsoftware: ER-Diagramm . . . . .	20
F	Migration 'translation_request' . . . . .	21
G	Auslesen der Metriken . . . . .	22
H	Model 'TranslationRequest' . . . . .	22
I	Controller 'TranslationRequest' . . . . .	23
J	Service 'DeepLService' . . . . .	24
K	JTL-Ameise GUI: Export . . . . .	25
L	Konfigurationsdatei . . . . .	26
M	SqlHelper . . . . .	27
N	Tabelle 'hash' . . . . .	28
O	Tabelle 'hash_language' . . . . .	28
P	ApiHelper . . . . .	29
Q	Weichstelle der API für Tests . . . . .	29
<b>8</b>	<b>Benutzerhandbuch</b>	<b>30</b>
<b>9</b>	<b>Installationsanleitung</b>	<b>33</b>

# 1 Einleitung

## 1.1 Projektumfeld

Meine Ausbildung zum Fachinformatiker für Anwendungsentwicklung absolviere ich in der Firma RIS Web- & Software-Development GmbH & Co. KG in Regensburg. Der Betrieb beschäftigt derzeit elf Mitarbeiter. Als Servicepartner von JTL-Software werden die Warenwirtschaftssoftware (JTL-Wawi) und der E-Commerce-Shop (JTL-Shop) betreut und mit Plugins oder anderen kleinen Softwareerweiterungen zur Abdeckung bestimmter Anforderungen erweitert. Für den JTL-Shop werden auch eigene Frontend-Templates umgesetzt. Für die JTL-Wawi zählen Einrichtung, Betreuungen und Schulungen zu den Hauptbereichen. Neben der Tätigkeit als Servicepartner werden auch Individuallösungen für Kunden entwickelt und betreut. Dazu zählen Entwicklungen von Corporate Designs, Social-Media-Marketing, Neugestaltung/Betreuung von Webseiten, Erstellung von Web-Applikationen und nativer Software.

## 1.2 Ist-Analyse

Unserer Kunden nutzen die E-Commerce-Software JTL-Shop zusammen mit dem Warenwirtschaftsprogramm JTL-Wawi. Diese werden standardmäßig auf Deutsch installiert, mit der anschließenden Möglichkeit mehrere Sprachen zu nutzen. Bei unseren Kunden besteht eine rege Nachfrage nach Übersetzungen in andere Sprachen. Derzeit werden diese oft vom Kunden händisch durchgeführt, was bei vielen Artikeln zu sehr viel Arbeitsaufwand führt. Außerdem müssen bei einer Änderung eines bereits bestehenden Inhalts auch die anderssprachigen Gegenstücke neu übersetzt werden, um Inkonsistenzen zu vermeiden. Im JTL-Ökosystem gibt es zwei für uns relevante Teilbereiche. Zum einen den JTL-Shop, welcher als PHP-Web-Applikation betrieben wird und hauptsächlich Fließtexte, aber auch kurze Satzteile in einer Datenbank speichert. Zum anderen die JTL-Wawi, ein Warenwirtschaftsprogramm, welches auf einem Windows-System agiert, in dem die Artikel, Kategorien sowie Attribute, Beschreibungen und SEO-Inhalte erstellt, und in einer eigenen MSSQL-Datenbank gespeichert werden. Die JTL-Wawi lädt diese Texte außerdem über einen sogenannten Onlineshop-Abgleich in den Shop hoch. Für beide Softwaresysteme müssen die, gegebenenfalls genutzten, HTML-Tags berücksichtigt werden.

## 1.3 Soll-Zustand

Zu Entwickeln sind daher ein JTL-Shop Plugin, eine Zusatzsoftware zur JTL-Wawi sowie eine Schnittstelle für uns. Diese Schnittstelle wird die Kommunikation mit dem tatsächlichen

Übersetzungs-Service übernehmen. Dabei soll sie zu übersetzende Texte entgegennehmen, an einen Übersetzungs-Dienst übertragen und die übersetzten Inhalte zurück liefern, sowie Nutzungsmetriken pro Kunde erfassen, zur Erhebung von Übersetzungsgebühren.

Meine Aufgaben werden die Entwicklung und Konzeption der Schnittstellen-Software und die Umsetzung der Zusatzsoftware auf dem Windowssystem sein. Damit soll die Übersetzung automatisiert an einen Drittanbieter ausgelagert werden, um den Arbeitsaufwand und mögliche Fehler zu minimieren.

## 1.4 Lösungsansätze

Bei Automatisierungen in einem großen Ausmaß, ist es wichtig den richtigen Ausführungszeitpunkt zu wählen. Dabei gilt es verschiedene Möglichkeiten abzuwägen, beispielsweise feste Uhrzeiten, oder stündliche Intervalle sowie das Aufrufen, wenn es einen neuen übersetzbaren Wert gibt. Eine feste Uhrzeit bei all unseren Kunden würde zu extremer Belastung der Schnittstelle führen, welche als API umgesetzt wird. Mehrmals täglich zu übersetzen, bietet den großen Vorteil, dass Änderungen schneller übernommen werden können und die Auslastung der API würde sich besser verteilen. Zusätzlich wäre ein Option eines manuellen Anstoßes sicherlich von Vorteil.

Eine weitere Unklarheit, die vor der Umsetzung der Zusatzsoftware genauer betrachtet werden sollte, ist die Art der Datenbankzugriffe auf die Datenbank der JTL-Wawi. Es gibt zwei Arten auf die MSSQL-Datenbank zuzugreifen, direkt über SQL oder über die von JTL bereitgestellte Software JTL-Ameise. Sie ermöglicht einen überwachten und einwandfreien lesenden und schreibenden Zugriff auf die Datenbank. Dabei können große Mengen an Anpassungen, der gleichen Art, in einem Durchlauf durchgeführt werden, was für die Übersetzungen optimal ist. Diese Software ist jedoch erfahrungsgemäß sehr langsam (ein Export mit etwa 50.000 Artikeln braucht je nach Server etwa 4 Stunden). Der direkte Zugriff auf die Datenbank mit Hilfe von SQL ist deutlich schneller, jedoch sprechen die stetigen Datenbank-Schema-Veränderungen bei Updates, sowie fehlende Stored-Procedures dagegen. Kleinste Fehler können zu Inkonsistenzen in den Tabellen oder Datenverlust führen. Da die Datenbank mit sehr sensiblen Kundendaten arbeitet und die Zugriffe nicht für jede Wawi-Version exakt gleich sind, fällt die Wahl auf JTL-A

Die Auswahl der Technologien und Systeme für die (REST-)API ist groß, daher es gibt hier vor allem Entscheidungskriterien, der Wartbarkeit und Erweiterbarkeit. In meinem Betrieb ist PHP allgegenwärtig und daher im Vergleich zu JavaScript deutlich wartbarer für alle Kollegen. Die Entscheidung, die API in PHP zu programmieren, fiel daher recht schnell und einfach.

Für die Übersetzungssoftware wurden verschiedene Anbieter wie Amazon Translate, Google Translate, Microsoft Translator und DeepL Translator getestet. Dabei konnten die namenhaften Dienstleister leider entweder qualitativ nicht mithalten, oder bieten keine API, die aufgerufen werden kann. Daher soll der Übersetzungsdienst von DeepL benutzt werden. Er bietet nicht nur sehr gute Übersetzungen, sondern außerdem auch eine Vielzahl an verschiedenen Ein-/ und Ausgangssprachen sowie die Option, HTML zu berücksichtigen. Die Nutzung des Services via API ist zwar nicht kostenlos, jedoch wird für die sehr gute Übersetzungsqualität nur ein kleiner Betrag pro Zeichen fällig.

## 1.5 Technologieplattform

Die Nutzung eines Framework wie Laravel bietet sich bei der Arbeit mit PHP zusätzlich an, da die Entwicklung vereinfacht und beschleunigt werden kann. Dank der hervorragenden Dokumentation hat sich Laravel in der Vergangenheit als sicher, schnell, zuverlässig und auf Grund des Model-View-Controller-Entwurfsmusters (MVC) auch als sehr übersichtlich herausgestellt. Dabei repräsentieren *Models* Daten, die von einem *Controller* verarbeitet werden und dann mit Hilfe einer *View* als (optische) Ausgabe dargestellt werden können. Laravel bietet außerdem standardmäßig eine MySQL-Unterstützung, welche für das Projekt genutzt werden wird.

Da die JTL-Wawi das Microsoft .NET Framework voraussetzt, kann auch eine C#-Applikation problemlos ausgeführt werden. Daher bietet sich C# als Programmiersprache an und wird im Projekt für die Umsetzung der Zusatzsoftware benutzt.

Für die Datenbankzugriffe wird das von JTL bereitgestellte Programm JTL-Ameise verwendet, da es erlaubt alle Datenbanktabellen der JTL-Wawi zu ex-/ bzw. importieren. Das Programm arbeitet mit CSV-Dateien, was eine angenehme Dateiverarbeitung ermöglicht und mit C# leicht verarbeitet werden kann.

Zuletzt gilt es den Dienst zu erwähnen, welcher für die Übersetzung genutzt wird. DeepL hat sich als eine sehr hochwertige Übersetzungssoftware entpuppt, daher wird sie in diesem Projekt ihre Anwendung finden.

Da bereits die DeepL-API über das JSON-Format kommuniziert, bietet sich dieses auch für die Middleware an.

## 2 Entwurfsphase

### 2.1 Schnittstelle

#### 2.1.1 Datenbankmodell

Da die API Anfragen empfängt und diese den richtigen Kunden für die Abrechnung zuordnen soll, braucht es zwei Tabellen. Eine für den Kunden ('customers') und eine für dessen Übersetzungsanfragen ('translation\_requests'). In 'customers' wird der Kunde und sein API-Schlüssel zur Identifikation gespeichert, während in 'translation\_requests' der Inhalt, dessen Zeichenslänge, ein Zeitstempel so wie die Verknüpfung zum Kunden hinterlegt werden soll. Für die Abrechnung muss eine Übersetzungsanfrage immer genau einem Kunden zugewiesen werden können (siehe Anhang A).

#### 2.1.2 Aufbau

REST-API steht für *Representational State Transfer Application Programming Interface*. Der Hauptanwendungsbereich dieser Schnittstellen sind Web-Applikationen, welche mittels HTTP Anfragen, wie GET, PUT, POST oder DELETE, Daten zwischen Softwaresystemen austauschen können. Dabei werden in diesem Fall die Datenpakete im JSON-Format übertragen, um eine standardisierte Strukturierung zu ermöglichen.

Eine API hat oft mehrere Endpunkte, welche jeweils mit unterschiedlichen HTTP-Methoden genutzt werden können (siehe Anhang B). Die API wird mit dem PHP-Framework Laravel erstellt. Dabei folgt es dem MVC-Entwurfsmuster, welches sich durch die Trennung der Anwendungslogik von Darstellung und Benutzerinteraktionen auszeichnet.

Die *Models* der API beziehen sich auf die beiden Datenbanktabellen (siehe Anhang C und D), und können aus einer Tabellenzeile jeweils ein Objekt formen. Der *View* gibt übersetzte Inhalte in Form eines JSON-Objekts zurück.

Außerdem wird ein *Controller* für die Verarbeitung der *Requests* benutzt. Er wird Texte empfangen, eine Übersetzungsanfrage an DeepL stellen und anschließend den übersetzten Wert zurückgeben. Die gesamte Verarbeitungslogik passiert ebenfalls im Controller.

## **2.2 Zusatzsoftware**

### **2.2.1 Datenbankmodell**

Um einen vorliegenden Datensatz nicht zu übersetzen, obwohl dieser schon einmal übersetzt wurde, muss abgespeichert sein, dass es bereits eine Übersetzung in die jeweilige Sprache gibt. Ansonsten entstehen Kosten und Zeitaufwand/Auslastung am Server, die man auch vermeiden kann.

Um mögliche Änderungen in einem Text erkennen zu können, muss ein Indikator entworfen werden, der schnell auslesbar ist und wenn möglich Versionen, anhand der Länge oder eines Zeitstempels erkennt. Dieser wird anschließend, zusammen mit einem eindeutigen Identifikator des übersetzten Textes in einer Datenbanktabelle gespeichert (siehe Anhang E).

### **2.2.2 Aufbau**

Die Zusatzsoftware wird in C# und damit auch objektorientiert programmiert. Sie wird die Texte, mit Hilfe der JTL-Ameise, aus der Datenbank der JTL-Wawi auslesen und anschließend diese gelesenen Werte an die API schicken. Die Antworten des Übersetzungsdienstes werden dann als neue Datensätze in eine, für die JTL-Ameise lesbare CSV-Datei, geschrieben und importiert.

Um die Verarbeitungslogik von den Im-/ Exporten sowie den API-Aufrufen zu trennen, werden verschiedene Klassen entworfen. Dabei werden die Kommunikationswege zwischen der Zusatzsoftware und der JTL-Ameise bzw. API in jeweils eine eigene Klasse gespalten. Außerdem soll die Verarbeitung und Aufbereitung der CSV-Dateien ebenso vom restlichen Programmcode separiert werden.



## 3 Implementierung

### 3.1 Schnittstelle

#### 3.1.1 Datenbanktabellen/Migrations

Eine Datenbank-Tabelle kann in Laravel mit Hilfe von *Migrations* erzeugt und manipuliert werden. Für jede Tabelle wird also eine eigene Klasse erstellt, die von der *Migration*-Klasse erbt. Außerdem werden in jeder solchen Klasse zwei Methoden implementiert, welche für strukturelle Änderungen in der Datenbank sorgen können. Die `up()` Methode setzt die Änderungen um, während die `down()` Methode die Änderungen wieder rückgängig macht (siehe Anhang F). So entsteht eine Art Versionierungsprotokoll.

Da die API derzeit noch keine GUI besitzt, wurde intern beschlossen die Ausgabe der Nutzungsdaten der Kunden über einen SQL-Befehl auszulesen. So werden die Metriken derzeit monatlich über eine Datenbankverwaltungssoftware ausgelesen. Im Anhang G wird der Befehl für den Monat Mai aufgezeigt.

#### 3.1.2 Entitätstypen/Models

Für jede Datenbanktabelle wird außerdem eine Klasse benötigt, welche man als *Model* bezeichnet. Im Fall dieser API gibt es also die *Models Customer* und *TranslationRequest*. Diese beinhalten nicht nur die Attribute, die die Tabellenspalten repräsentieren, sondern ggf. auch kleinere Codeteile, beim Schreiben eines Datensatzes aufgerufen werden. So kann beispielsweise beim Setzen des Attributs `content` direkt die Länge der Zeichenkette berechnet und in einer weiteren Variablen gespeichert werden (siehe Anhang H Zeile 24ff.). Außerdem sind in der Klasse *Model* die Funktionalitäten für die Datenbankoperationen, wie zum Beispiel `save()`, `update()` oder `delete()` verankert.

#### 3.1.3 Controller

Die verschiedenen *Routen* werden von einer Methode im jeweiligen *Controller* bedient.

POST: `/api/translation_request/?[params]`

Die POST-Anfrage an diese *Route* wird vom *TranslationRequestController* mit der Methode `store()` bedient (siehe Anhang I). Dort werden zu Beginn die Daten der Anfrage validiert. Sollte dies fehlschlagen, wird automatisch ein Fehler zurückgegeben, ansonsten wird eine Verknüpfung zum Kunden, anhand des API-Keys, hergestellt. Es wird eine neue Instanz des

`DeepLServices` erstellt und die `translate()` Methode mit dem Text als Übergabeparameter aufgerufen. Nachdem die Antwort mit dem übersetzten Text angekommen ist, kann dieser zurückgegeben werden.

### 3.1.4 Services

Die Kommunikation mit der API von DeepL wird in einen *Service* gespalten, um die Verarbeitungslogiken sinnvoll voneinander zu trennen. Die `translate()` Methode (siehe Anhang J) liest eine Konfigurationsdatei aus, in der sowohl die DeepL-API-URL als auch ein API-Token für die Authentifizierung liegt. Im Anschluss wird eine HTTP-Anfrage an die DeepL-API, aus den übergebenen Daten formuliert. Dazu gehören ein Header, welcher einen *user-agent* verlangt, sowie der tatsächliche Inhalt, welcher aus einem Parameterobjekt besteht. Das Ergebnis ist der übersetzte Inhalt oder ein Fehler, welcher direkt weiter an den *Controller* zurückgegeben wird.

## 3.2 Zusatzsoftware

### 3.2.1 Konfiguration

Die in 1.4 angesprochenen langen Wartezeiten können mit Hilfe einer Filterung deutlich verkürzt werden. Die JTL-Wawi bietet Automatisierungsprozesse mit denen Artikel beim Verändern eine Markierung bekommen können, so werden die Exporte um ein vielfaches kleiner und somit schneller.

Um CSV-Dateien über die JTL-Ameise zu importieren/exportieren, müssen vorher Import- und Exportvorlagen erstellt werden. Das passiert händisch über die GUI der JTL-Ameise (siehe Anhang K).

Außerdem muss eine Konfigurationsdatei mit den Zugangsdaten zur JTL-Wawi Datenbank befüllt werden (siehe Anhang L).

### 3.2.2 SqlHelper Klasse

Zu Beginn des Programmstarts wird, mit Hilfe der Konfigurationsdatei, eine Verbindung zur JTL-Wawi Datenbank hergestellt und falls noch nicht geschehen, zwei neue Tabellen angelegt. Dazu wird die `SqlHelper` Klasse aufgerufen, welche eine `execute()` Methode zur Verfügung stellt, die einen übergebenen SQL-Befehl ausführen kann (siehe Anhang M). In den Tabellen wird gespeichert, welche Texte bereits übersetzt wurden. Um eine Prüfsumme der übersetzten

Datensätze zu erhalten, wird ein Hashwert des Originaltexts gebildet und zusammen mit einem, aus zwei Elementen bestehenden, Identifikator gespeichert (siehe Anhang N). Da ein Text in mehrere Sprachen übersetzt werden kann, entsteht eine '1 zu n' Beziehung, welche mit einer weiteren Tabelle aufgelöst wird (siehe Anhang O).

### 3.2.3 AntHelper Klasse

Sind die Tabellen vorhanden wird im nächsten Schritt die JTL-Ameise mit den vorher erstellten Exportvorlagen aufgerufen. Dafür wird ein Objekt der Klasse **AntHelper** erstellt und die **export()** Methode aufgerufen (siehe Anhang P). Wenn die Exporte vollständig abgearbeitet wurden, sind im dafür vorgesehenen Verzeichnis die CSV-Dateien mit den zu übersetzenden Daten gespeichert. Die Klasse bietet außerdem eine Methode **import()** um später übersetzte Inhalte wieder in die JTL-Wawi Datenbank zu schreiben, dazu wird eine CSV-Datei und ein passendes Importformat benötigt.

### 3.2.4 CsvHelper und ApiHelper Klasse

Zum Verarbeiten der CSV-Dateien wird eine .NET-Bibliothek namens **CsvHelper** verwendet. Sie ermöglicht es, die zu übersetzenden Texte, Feld für Feld aus der CSV-Datei auszulesen. Die gelesenen Datensätze werden anschließend an ein neues Objekt der Klasse **ApiHelper** als Übergabeparameter weitergegeben, um eine Übersetzung anzustoßen. Wird der Datensatz erfolgreich übersetzt, so wird der Hashwert davon in der Tabelle **hash** gespeichert, und je nach Sprachen entsprechende Verweise in der Tabelle **hash\_language** gespeichert. Anschließend wird erneut das Objekt der **CsvHelper** Bibliothek genutzt, um eine CSV-Datei für den Import zu schreiben.

Nach erfolgreichem Schreiben der Import-Datei wird der **AntHelper** erneut aufgerufen, diesmal mit der Methode **import()**. Der Vorgang wird mittels der Windows-Aufgabenplanung in regelmäßigen Abständen wiederholt, kann aber auch manuell gestartet werden.

## 4 Testphase

### 4.1 Schnittstelle

Die Datenbankanbindung wurde bereits während der Entwicklung getestet und mit Hilfe eines MySQL-Plugins direkt in der Entwicklungsumgebung überprüft. Da die Laravel-Controller mit Hilfe der *Models* bereits in der Lage sind, die Datenbank zu beschreiben, wenn ein passendes *Model* vorliegt, mussten hier keine weiteren Tests gemacht werden. Somit wurden die Datenbanktabellen lediglich auf ihre Richtigkeit sightgeprüft.

Die Schnittstelle dient als Middleware und gibt Fehler, die bei der Verbindung mit der DeepL-API auftreten, direkt weiter an den Endnutzer. Bei Fehlern in der Schnittstelle selbst werden Statuscodes in die Datenbank geschrieben.

Ansonsten wurden, mittels Testanfragen, vor allem überprüft, dass keine Routen für den Endnutzer verfügbar sind, mit denen Daten aus der Datenbank gelesen werden können.

Auch die Validierung in der `store()` Methode des `TranslationRequestController` wurde ausgiebig überprüft, indem verschiedene fehlerhafte Parameter an die Route übersendet wurden.

### 4.2 Zusatzsoftware

Für die Zusatzsoftware zur JTL-Wawi wurden vor allem die einzelnen Klassen an sich getestet.

Zuerst muss das Programm eine problemfreie Verbindung mit der Datenbank herstellen können, um auch schnell und zuverlässig eine Überprüfung durchführen zu können.

Anschließend wurde überprüft, ob der Aufruf der JTL-Ameise korrekt funktioniert. Das kann lediglich von einem Menschen kontrolliert werden, indem die CSV-Datei auf ihre Integrität sightgeprüft wird.

Um die Anbindung an die API zu testen, wurde im Controller ein `api_key` hinterlegt, für den kein Übersetzungsprozess angestoßen wird, sondern lediglich das Kürzel der Zielsprache mit dem Ausgangstext in Klammern ausgegeben wird (siehe Anhang Q). So können größere Tests durchgeführt werden, ohne Kosten zu verursachen, solange der Test-API-Schlüssel in der Anfrage angegeben wird.

## 5 Schluss

### 5.1 Soll-Ist-Vergleich

Projektphasen	Soll-Zeit [h]	Ist-Zeit [h]	Differenz [h]
<b>Projektplanung</b>			
Ist-Analyse	2	2	0
Soll-Konzept	2	2	0
Lösungsansätze vergleichen	2	2	0
<b>Entwurf</b>			
Datenbankmodell Schnittstelle	1	1	0
Datenbankmodell Zusatzsoftware	2	2	0
Aufbau Schnittstelle	4	3	-1
Aufbau Zusatzsoftware	4	6	+2
<b>Implementierung</b>			
<b>Schnittstelle</b>			
Erstellen der Datenbank	5	4	-1
Logik zur Verarbeitung	10	9	-1
Ausgabe der Metriken	3	2	-1
<b>Zusatzsoftware</b>			
Datenbankintegration	4	4	0
Erfassung von übersetzten Inhalten	2	3	+1
Im- und Exportformate	5	6	+1
Verarbeitung der Daten	5	6	+1
<b>Testen</b>			
Testen der Schnittstelle	3	3	0
Testen der Zusatzsoftware	5	4	-1
<b>Projektabschluss</b>			
Soll-Ist-Vergleich	1	1	0
Dokumentation	7	7	0
Abnahme	1	1	0
Wirtschaftlichkeitsanalyse	1	1	0
Gesamtstunden:	70	70	0

Der Aufbau der Schnittstelle konnte durch die Verwendung von Laravel deutlich schneller als erwartet festgelegt werden. Ebenso wurde durch die vorgegebene Struktur von Laravel viel Zeit bei der Erstellung von Datenbanken und Klassen eingespart.

Dafür war die Handhabung der JTL-Wawi Datenbankzugriffe ein größeres Problem als ursprünglich geschätzt. Auch die Implementierung dauerte dadurch etwas länger als zunächst erwartet.

## 5.2 Dokumentation und Abnahme

Es wurde eine API-Dokumentation (siehe 8.) für Endnutzer, sowie eine Installationsanleitung zur Zusatzsoftware (siehe 9.) erstellt. Die Software wurde anschließend zum Code-Review an ein Entwicklerteam abgegeben.

## 5.3 Wirtschaftlichkeitsanalyse

### 5.3.1 Kosten

Die Entwicklung des Projekts beläuft sich auf 70h. Hinzu kommen 70h die für die Entwicklung eines JTL-Shop-Plugins von einem Kollegen, wie in 1.3 beschrieben.

Aus buchhalterischer Sicht fallen für einen Auszubildenden pro Stunde etwa 20€ brutto an Kosten an. Außerdem ist noch ein Gemeinkostenzuschlagsatz von etwa 10% zu verrechnen. Somit betragen die gesamten Entwicklungskosten  $3080\text{€} = ((70\text{h} + 70\text{h}) \cdot 20\text{€}) + 10\%$ .

Die Einrichtung pro Kunde nimmt etwa 4h in Anspruch und muss ebenfalls mit Kosten von 20€/h berücksichtigt werden.

Die Kosten für die Übersetzung eines Shops ist abhängig von der Anzahl der zu übersetzenden Zeichen. Des weiteren sind die laufenden Kosten für Änderungen und neue Texte nur schwer einzuschätzen und stark abhängig von der Schnellebigkeit des Sortiments.

DeepL berechnet für 1.000.000 Zeichen exakt 20€ und eine monatliche Grundgebühr von 5€. Das bedeutet, dass pro Shop und pro Sprache initial mit etwa 60€ Kosten zu rechnen sind, und monatlich etwa 3€ anfallen.

#### Beispiel für einen Kunden:

Kosten einmalig	Betrag	Kosten monatlich	Betrag
1. Einrichtung	80€	1. Bereitstellungspauschale <sup>2</sup>	5€
2. initiale Übersetzung <sup>1</sup>	60€	2. monatl. Zeichenmenge <sup>1</sup>	3€
Gesamt:	140€	Gesamt:	8€

<sup>1</sup> Annahme: Ein Shop enthält 3 Mio. Zeichen sowie eine monatliche Erneuerungsrate von 5%.

<sup>2</sup> Wird nur einmalig fällig. Bei weiteren Kunden muss der Betrag nicht erneut bezahlt werden.

### 5.3.2 Einnahmen

Das Abrechnungskonzept von DeepL bleibt gegenüber dem Kunden unverändert und besteht aus einer Bereitstellungspauschale, sowie eines Tarifs für die Anzahl der übersetzten Zeichen. Die Bereitstellungspauschale soll pro Kunde im Jahr etwa 10% der Entwicklungskosten betragen.

$$\frac{3080\text{€} \cdot 0.1}{12} = 25.66\text{€} \approx 25\text{€/Monat}$$

Hinzukommend wird eine Einrichtungspauschale von 400€ einmalig fällig. Außerdem werden die Übersetzungskosten mit einer Gewinnmarge von 100% an den Kunden weitergegeben, daher verdoppeln sich alle Übersetzungskosten.

Kosten einmalig	Betrag	Kosten monatlich	Betrag
1. Einrichtungspauschale	400€	1. Bereitstellungspauschale	25€
2. initiale Übersetzung <sup>1</sup>	120€	2. monatl. Zeichenmenge <sup>1</sup>	6€
Gesamt:	520€	Gesamt:	31€

### 5.3.3 Fazit

Bei der Einrichtung ergibt sich somit ein Gewinn von 380€, sowie ein monatlicher Gewinn von etwa 18€ pro Kunde.

Somit kann berechnet werden, wie viel anschließend noch bis zu Amortisierung fehlt:

$$3080\text{€} - (3 \cdot 380\text{€}) = 1940\text{€}$$

Berechnung der Amortisierung mit 3 Kunden anhand der laufenden Kosten:

$$\frac{1940\text{€}}{3 \cdot 18\text{€}} = 35.93 \text{ Monate} \approx 36 \text{ Monate}$$

Mit drei Kunden und durchschnittlichen Nutzungswerten würde sich das Projekt nach 3 Jahren amortisieren. Das ist eine sehr weitsichtige Investition und damit sehr risikoreich. Nach sorgfältiger Rücksprache mit den drei auftraggebenden Kunden wurde beschlossen, die Entwicklungskosten zu gleichen Teilen auf sie aufzuteilen. Somit ist das Projekt von der Fertigstellung an amortisiert. Da die laufenden Kosten deutlich unter den laufenden Einnahmen liegen, können die Gewinne in die Instandhaltung, Wartung und Erweiterung der Software reinvestiert werden.

<sup>1</sup> Annahme: Ein Shop enthält 3 Mio. Zeichen sowie eine monatliche Erneuerungsrate von 5%.

## 5.4 Ausblick

Das Projekt war erfolgreich und wird vorraussichtlich mehr als nur die auftraggebenden Kunden erreichen.

Es bieten sich noch einige Erweiterungsmöglichkeiten an, welche nicht mehr im Rahmen des Projekts abwickelbar waren. Beispielsweise eine GUI für das Auslesen und Beschreiben der API-Datenbanken, um leichter Kunden hinzuzufügen oder deren Nutzungsdaten auszulesen. Auch eine Automatisierung der monatlichen Abrechnung wäre denkbar und sinnvoll.

Je mehr Kunden das Produkt erhalten, desto wichtiger wird die Robustheit der API, so könnte es in Zukunft relevant sein, ein mögliches Bottleneck in der API zu verhindern.



## 6 Glossar

**API** Application Programming Interface; Eine Programmierschnittstelle zur von Software zu Software.

**Bottleneck** ein Engpass in einem Prozess, in dem die Auslastung besonders hoch ist.

**CSV** Comma separated values; Dateiformat mit dem meist Tabellen dargestellt werden. Einzelne Spalten werden mit einem Komma [,] getrennt.

**Framework** Bereitstellung eines komponentenbasierten Grundgerüsts.

**GUI** Graphical User Interface; Eine grafische Bedienoberfläche einer Software.

**Hashwert** ist eine Art Fingerabdruck eines Datensatzes.

**HTTP** Hyper Text Transfer Protokol; Überwiegend in Web-Anwendungen verwendetes zustandsloses Datenübertragungsprotokoll zur Weitergabe von Daten an den Anwender.

**JSON** JavaScript Object Notation; Ist ein Programmiersprachenunabhängiges Datenformat, um Daten in einer festgelegten Struktur an den Anwender zu übergeben.

**Laravel** Framework im Entwurfsmuster Model-View-Controller, dass Bausteine zur Entwicklung von Web-Applikationen bereitstellt.

**Middleware** ist eine Software die zwischen zwei Anwendungen vermittelt.

**MSSQL** Microsoft SQL Datenbank.

**MVC** Model-View-Controller; Ist ein Entwurfsmuster zur Unterteilung der Software-Elemente in Model, View und Controller zur besseren Wiederverwendbarkeit der einzelnen Elemente.

**PHP** Hypertext Preprocessing; Weit verbreitete Skript-Sprache, überwiegend in der Web-Entwicklung genutzt.

**REST** Representational State Transfer; Ein Architekturstil für die Kommunikation zwischen zwei Softwaresystemen.

**SEO** Search Engine Optimization; Die Optimierung einer Webseite, zur besseren Sichtbarkeit für Suchmaschinen, wie zum Beispiel Google.

**SQL** Structured Query Language.

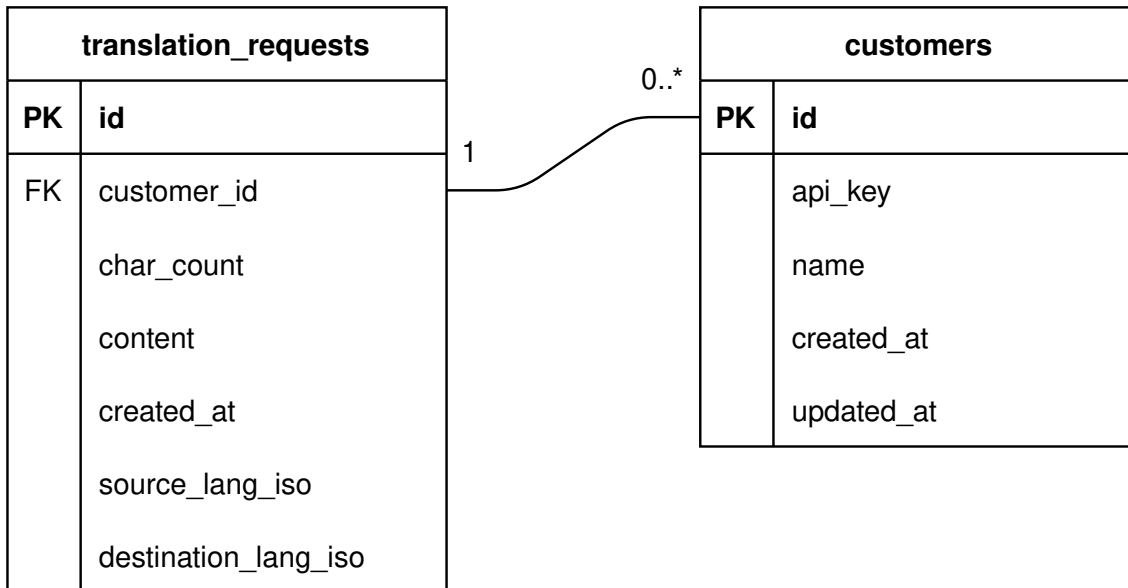
**Stored-Procedure** eigenständiger SQL-Befehl der für bestimmte Prozeduren gespeichert wurde.

**URL** Uniform Resource Locator; Dient der eindeutigen Adressierung von Dateien und Verzeichnissen und wird hauptsächlich im Internet genutzt.

**Windows-Aufgabenplanung** ist eine Windowsfunktion, die es erlaubt Programme zu festgelegten Zeitpunkten zu starten.

## 7 Anhang

### A Schnittstelle: ER-Diagramm



### B Liste der HTTP-Endpunkte

Methode	URI	Beschreibung
GET	api/ruok	Gibt den aktuellen Status der API zurück
GET	api/translation_request	Zeigt alle gespeicherte Datensätze
POST	api/translation_request	Schreibt einen Datensatz
GET	api/translation_request/{ id }	Zeigt einen Datensatz
PUT	api/translation_request/{ id }	Ändert einen Datensatz
DELETE	api/translation_request/{ id }	Löscht einen Datensatz

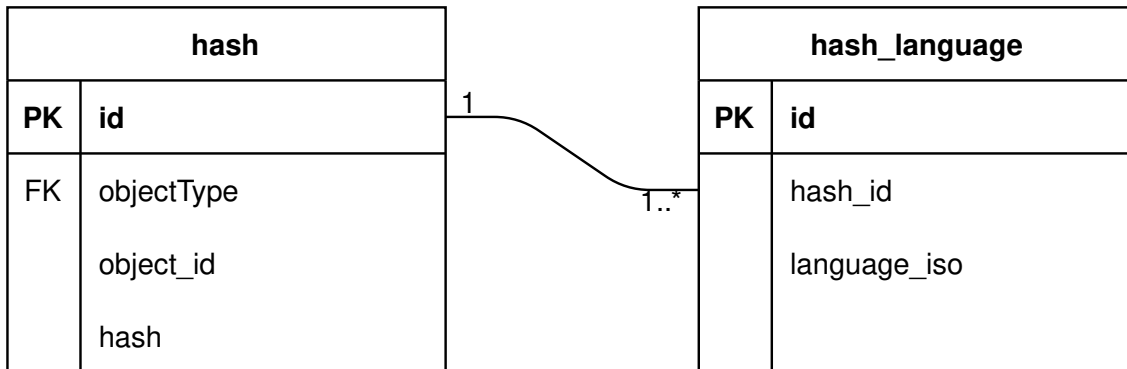
### C Tabelle 'translation\_requests'

ID	content	source	destination	count	customer_id	status	created_at
...	...	...	...	...	...	...	...
186	Tisch	DE	EN	5	1	200	2021-04-29 12:44:48
187	Straße	DE	EN	7	1	200	2021-04-29 12:45:38
188	Maske	DE	EN	5	2	200	2021-04-29 12:46:30
189	Uhr	DE	EN	3	1	200	2021-04-29 12:55:16
...	...	...	...	...	...	...	...

## D Tabelle 'customers'

ID	api_key	name	created_at	updated_at
1	d6c117cfa61375...	langer@[...].de	2021-04-29 12:47:56	2021-04-29 12:47:56
2	b6f8d434a847fb...	birnthaler@[...].de	2021-04-29 12:47:56	2021-04-29 12:47:56
...	...	...	...	...

## E Zusatzsoftware: ER-Diagramm



## F Migration 'translation\_request'

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateTranslationRequestsTable extends Migration
8 {
9     //
10    // Run the migrations.
11    // @return void
12    //
13    public function up()
14    {
15        Schema::create('translation_requests', function (Blueprint $table) {
16            $table->id();
17
18            $table->longText('content');
19
20            $table->string('source_language_iso', 2);
21            $table->string('destination_language_iso', 2);
22
23            $table->bigInteger('char_count');
24            $table->integer('status');
25            $table->foreignId('customer_id')->constrained('customers');
26
27            $table->timestamps();
28        });
29    }
30
31    //
32    // Reverse the migrations.
33    // @return void
34    //
35    public function down()
36    {
37        Schema::dropIfExists('translation_requests');
38    }
39 }
```

## G Auslesen der Metriken

```

1 SELECT c.name as "Kunde", SUM(char_count) as "Anzahl der Zeichen"
2 FROM translation_requests as r
3 JOIN customers c ON r.customer_id = c.id
4 WHERE r.created_at
5 BETWEEN "2021-05-01 00:00:00" AND "2021-05-31 23:59:59"
6 GROUP BY c.name

```

Ausgabe

Kunde	Anzahl der Zeichen
langer@ris-development.de	256187
birnthaler@ris-development.de	65536

## H Model 'TranslationRequest'

```

1 <?php
2 namespace App\Models;
3
4 use Illuminate\Database\Eloquent\Factories\HasFactory;
5 use Illuminate\Database\Eloquent\Model;
6
7 class TranslationRequest extends Model{
8     use HasFactory;
9     protected $fillable = [
10         'content',
11         'source_language_iso',
12         'destination_language_iso',
13         'char_count',
14         'status',
15         'customer_id',
16     ];
17
18     protected $attributes = [
19         'char_count' => 0,
20         'status' => 1,
21     ];
22
23     public function setContentAttribute($value){
24         $this->attributes['content'] = $value;
25         $this->attributes['char_count'] = strlen($value);
26     }
27 }

```

## I Controller 'TranslationRequest'

```
1
2 class TranslationRequestController extends Controller{
3
4     public function store(Request $request){
5
6         $data = $request->validate([
7             'content' => "required|min:1",
8             'source_language_iso' => "required|size:2",
9             'destination_language_iso' => "required|size:2",
10            'api_key' => "exists:".Customer::class.",api_key"
11        ]);
12
13        $customer = Customer::where('api_key', $data['api_key'])->firstOrFail();
14
15        $data['customer_id'] = $customer->id;
16        $translationRequest = TranslationRequest::create($data);
17
18        $service = new DeeplService;
19        $response = json_decode($service->translate($translationRequest));
20
21        //sent request
22        $translationRequest->status = 200;
23        $translationRequest->save();
24
25        return $response->translations[0]->text;
26    }
27 }
```

## J Service 'DeepLService'

```
1 class DeeplService {
2
3     public function translate(TranslationRequest $request){
4         $token = config('services.deepl.token');
5         $apiUrl = config('services.deepl.api_url');
6
7         $response = Http::asForm()->withHeaders([
8             'user-agent' => 'BilangApp',
9         ])->post($apiUrl, [
10             'auth_key' => $token,
11             'text' => $request->content,
12             'target_lang' => $request->destination_language_iso,
13             'source_lang' => $request->source_language_iso,
14             "tag_handling" => "xml"
15         ]);
16
17         return $response->body();
18     }
19 }
```



## K JTL-Ameise GUI: Export

JTL-Wawi Ameise: Daten aus JTL-Wawi exportieren - Mandant: Mandant\_19

Export von

- Artikel
  - Artikelbilder
  - Artikeldaten
  - Artikelskategorien
  - Attribute
  - Cross-Selling-Artikel
  - Eigene Felder
  - Gebinde
  - Lieferantenartikel
  - Mediendateien
  - Merkmale
  - Stücklisten
  - Variationen
  - Variationskombinationen
  - Hersteller
  - Recycling Registrierung für
- Aufträge
  - Aufträge
- JTL-WMS (Lagerverwaltung)
  - Lagerbestand
  - Lagerplätze
  - Nachschubkonfiguration
  - Inventurdifferenzliste
- Konfigurationsartikel
  - Konfigurationsartikel zuordn
  - Konfigurationsgruppen
  - Konfigurationskomponenten
- Kunden
  - Kundendaten
  - Kundenindividuelle Preise
- Lieferanten
  - Eingangsrechnungen
  - Lieferantenbestellungen
  - Lieferantendaten
- Marktplätze
  - Amazon-Angebotsupdate
  - eBay-Angebotsupdate
  - eBay-Attribue
- Buchungsdaten
  - Rechnungen (Neu)
  - Rechnungen DATEV (abgr

Exportdatei Format

☒ Kopfzeile erstellen Spaltenbegrenzer: Semikolon (;) Quote-Zeichen: " Escape-Zeichen: " Encoding: Utf-8

Artikelfilter

Filter: HasChanged Filter konfigurieren

☒ Auch inaktive exportieren (default)

Einstellungen für Kommazahlen

Dezimaltrenner: . Tausender-Trenner: keins Stellen: 2

Exportvorlage

Ausgewählte Exportvorlage: rs\_article\_bilang Vorlage wählen

Wenn Lieferantenartikel Felder ausgewählt sind Lieferantendaten nach dieser Regel exportieren: Alle Lieferanten

Datenfelder (zu exportierende Felder doppelklicken)

Alle Felder wählen Alle Felder abwählen

Feld in Datenbank

- Mindestabnahme: Händler
- Abnahmeintervall: Händler
- Mindestabnahme: Profisportler
- Abnahmeintervall: Profisportler
- Texte - weitere Plattformen -----
- Drucken/Mailen/Faxen: Artikelname
- Drucken/Mailen/Faxen: Kurzbeschreibung
- Drucken/Mailen/Faxen: Beschreibung
- eBay: Artikelname
- eBay: Kurzbeschreibung
- Amazon: Artikelname
- Amazon: Kurzbeschreibung
- Amazon: Beschreibung
- BiLang: Artikelname
- BiLang: Kurzbeschreibung
- BiLang: Beschreibung
- BiLang: SEO (Suchmaschinenname)
- BiLang: Titel-Tag
- BiLang: Meta-Keywords
- BiLang: Meta-Description
- Texte - weitere Plattformen - Georgisch -----

Diese Felder werden exportiert (rosa Felder können editiert werden)

Feld	Spalte in Exportdatei	Kopfzeilenbeschriftung
Artikelnummer	Spalte 1	Artikelnummer
Artikelname	Spalte 2	Artikelname
Kurzbeschreibung	Spalte 3	Kurzbeschreibung
Beschreibung	Spalte 4	Beschreibung
SEO Name (Suchmaschinenname)	Spalte 5	SEO Name (Suchmaschin...
SEO Titel-Tag	Spalte 6	SEO Titel-Tag
SEO Meta-Keywords	Spalte 7	SEO Meta-Keywords
SEO Meta-Description	Spalte 8	SEO Meta-Description
Drucken/Mailen/Faxen: Artikelname	Spalte 9	Drucken/Mailen/Faxen: Ar...
Drucken/Mailen/Faxen: Kurzbeschrei...	Spalte 10	Drucken/Mailen/Faxen: K...
Drucken/Mailen/Faxen: Beschreibung	Spalte 11	Drucken/Mailen/Faxen: B...
eBay: Artikelname	Spalte 12	eBay: Artikelname
eBay: Kurzbeschreibung	Spalte 13	eBay: Kurzbeschreibung
eBay: Beschreibung	Spalte 14	eBay: Beschreibung
Amazon: Artikelname	Spalte 15	Amazon: Artikelname
Amazon: Kurzbeschreibung	Spalte 16	Amazon: Kurzbeschreibung
Amazon: Beschreibung	Spalte 17	Amazon: Beschreibung

Nach oben Nach unten Freispalte hinzufügen Löschen

Abbrechen Hilfe Vorlage speichern... Export starten

## L Konfigurationsdatei

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <configuration>
3   <appSettings>
4     <!-- connection to wawi db -->
5     <add key="db_server" value="192.168.107.11\DEVWAWI" />
6     <add key="db_mandant" value="Mandant_19" />
7     <add key="db_pass" value="sa04jT14" />
8     <add key="db_user" value="sa" />
9     <add key="db_port" value="57981" />
10
11     <!-- Ameisendaten -->
12     <add key="jtlAmeisePath" value="C:\Programme\JTL-Software\JTL-wawi-ameise.exe" />
13   </appSettings>
14
15   <appLanguages>
16     <!-- add key="BG" value="" -->
17     <!-- add key="CS" value="" -->
18     <!-- add key="DA" value="" -->
19     <!-- add key="DE" value="" -->
20     <!-- add key="EL" value="" -->
21     <add key="EN" value="" />
22     <!-- add key="ES" value="" -->
23     <!-- add key="ET" value="" -->
24     <!-- add key="FI" value="" -->
25     <!-- add key="FR" value="" -->
26     <!-- add key="HU" value="" -->
27     <!-- add key="IT" value="" -->
28     <!-- add key="JA" value="" -->
29     <!-- add key="LT" value="" -->
30     <!-- add key="LV" value="" -->
31     <!-- add key="NL" value="" -->
32     <!-- add key="PL" value="" -->
33     <!-- add key="PT" value="" -->
34     <!-- add key="RO" value="" -->
35     <!-- add key="RU" value="" -->
36     <!-- add key="SK" value="" -->
37     <!-- add key="SL" value="" -->
38     <!-- add key="SV" value="" -->
39     <!-- add key="ZH" value="" -->
40   </appLanguages>
41 </configuration>
```

## M SqlHelper

```

1 using System;
2 using System.Configuration;
3 using System.Data.SqlClient;
4
5 namespace Bilanz.Classes {
6     public class SqlHelper {
7         public String DbServer { get; set; }
8         public String Db { get; set; }
9         public String DbUser { get; set; }
10        public String DbPass { get; set; }
11        public int DbPort { get; set; }
12        private String connectionString =>
13            "user id=" + DbUser + ";" +
14            "password=" + DbPass +
15            ";server=" + DbServer+ ((DbPort > 0) ? $",{DbPort}" : "") +
16            ";Trusted_Connection=no;Connect Timeout=10;" +
17            "Pooling=false;" +
18            "database=" + Db + "; " +
19            "connection timeout=30";
20
21        public SqlHelper() {
22            var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.
                None);
23            DbServer = config.AppSettings.Settings["db_server"].Value.Trim();
24            Db = config.AppSettings.Settings["db_mandant"].Value.Trim();
25            DbUser = config.AppSettings.Settings["db_user"].Value.Trim();
26            DbPass = config.AppSettings.Settings["db_pass"].Value.Trim();
27            DbPort = int.Parse(config.AppSettings.Settings["db_port"].Value.Trim());
28        }
29        public void execute(String query) {
30            using SqlConnection connection = new SqlConnection(connectionString);
31
32            connection.Open();
33
34            SqlCommand execute = new SqlCommand(query, connection);
35            execute.ExecuteNonQuery();
36            Console.WriteLine("Inserting Data Successfully");
37
38            connection.Close();
39        }
40    }
41 }

```

## N Tabelle 'hash'

ID	object_type	object_id	hash
...	...	...	...
255	kArtikel	45621	e3b0c44298fc1c149...
256	kArtikel	45621	9f86d081884c7d659...
257	kKategorie	19017	34ca495991b7852b8...
...	...	...	...

## O Tabelle 'hash\_language'

ID	hash_id	language_iso
...	...	...
419	255	EN
420	255	NL
410	256	EN
...	...	...

## P ApiHelper

```
1 public static string PostRequest(string url, string payload) {
2
3     HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
4     request.Method = "POST";
5     request.ContentType = "application/json";
6
7     using(var streamWriter = new StreamWriter(request.GetRequestStream())) {
8         streamWriter.Write(payload);
9     }
10
11     HttpWebResponse response = null;
12     string responseStr = "";
13
14     try {
15         response = (HttpWebResponse)request.GetResponse();
16         responseStr = ReadAllFromResponse(response);
17     }
18     catch(WebException e) {
19         Console.WriteLine("---\tError in request:\n" + e + "\n---");
20     }
21
22     using(var streamReader = new StreamReader(response.GetResponseStream())) {
23         var result = streamReader.ReadToEnd();
24     }
25
26     return responseStr;
27 }
```

## Q Weichstelle der API für Tests

```
1 if($data['api_key'] == "9876543210"){
2     return $data['destination_language_iso'].(".$data['content'].");
3 }
```

## 8 Benutzerhandbuch

# *BiLang-API: Technische Dokumentation*

Version 1.0.0 vom 19.05.2021

Autor: RIS Web- & Software-Development GmbH & Co. KG

Dies ist eine technische Dokumentation für die BiLang-Übersetzungs-API. Sie ist die Schnittstelle für die beiden anderen Komponenten der BiLang-Software. Sowohl das JTL-Shop-Plugin als auch die Zusatzsoftware für die JTL-Wawi sind auf diese API angewiesen.

Eine Anfrage an die API besteht grundsätzlich immer folgenden 4 Parametern:

Parameter	Beschreibung
content	Zu Übersetzender Text.
source_language_iso	Ausgangssprache des Parameters 'content'. . 'CS' - Tschechisch . 'DE' - Deutsch . 'EL' - Griechisch . 'EN' - Englisch . 'ES' - Spanisch . 'FR' - Französisch . 'HU' - Ungarisch . 'IT' - Italienisch . 'NL' - Niederländisch . 'PL' - Polnisch . 'PT' - Portugiesisch . 'RU' - Russisch . 'SV' - Schwedisch
destination_language_iso	Zielsprache in die 'content' übersetzt werden soll. . 'CS' - Tschechisch . 'DE' - Deutsch . 'EL' - Griechisch . 'EN' - Englisch . 'ES' - Spanisch . 'FR' - Französisch . 'HU' - Ungarisch . 'IT' - Italienisch . 'NL' - Niederländisch . 'PL' - Polnisch . 'PT' - Portugiesisch . 'RU' - Russisch . 'SV' - Schwedisch
api_key	Bei Vertragsschluss wird Ihnen ein API-Key übergeben. Bei Fragen wenden Sie sich bitte an den Support.

Dabei kann 'content' auch Inhalte mit XML-Format beinhalten, die XML-Tags werden dann nicht mit übersetzt, um HTML-Strukturen wie zum Beispiel Attribute oder Anker korrekt beizubehalten.

**Beispiele** HTTP-Methode: POST auf [bilang.ris-development.net/translation\\_request](http://bilang.ris-development.net/translation_request)

```
1 {  
2   "content": "Übersetzung",  
3   "source_language_iso": "DE",  
4   "destination_language_iso": "EN",  
5   "api_key": "9876543210"  
6 }
```

Response:

```
1 Translation
```

HTTP-Methode: POST auf [bilang.ris-development.net/translation\\_request](http://bilang.ris-development.net/translation_request)

```
1 {  
2   "content": "<a href=\"/übersetzung\"> Übersetzung </a>",  
3   "source_language_iso": "DE",  
4   "destination_language_iso": "EN",  
5   "api_key": "9876543210"  
6 }
```

Response:

```
1 <a href=\"/übersetzung\"> Translation </a>
```

**Limit** Die maximale Anfragensgröße darf 30kbytes nicht überschreiten,

**Formate und Sicherheit** Die Parameter werden benutzt, um Informationen an die API zu übergeben. Dabei wird vorausgesetzt, dass alle Parameter den UTF8-Zeichensatz benutzen. Die Antworten sind ebenfalls im UTF8-Zeichensatz. Ihr API-Key ist absolut einmalig und wird bei jeder Anfrage überprüft. Der gesamte Datenaustausch wird mit SSL abgesichert.



## 9 Installationsanleitung

# *BiLang-Zusatzsoftware: Installationsanleitung*

Version 1.0.0 vom 19.05.2021

Autor: RIS Web- & Software-Development GmbH & Co. KG

**1. Einleitung:** Um eine Installation erfolgreich abzuschließen, befolgen Sie bitte sämtliche Schritte, die in dieser Anleitung anschaulich erklärt werden. Sie benötigen:

- Zugangs- und Installationsrechte auf dem Windowssystem
- Zugangsdaten zu einem JTL-Wawi-Benutzer mit entsprechend hohen Rechten.
- Zugangsdaten zur JTL-Wawi-Datenbank
- Einen gültigen API-Schlüssel
- Die Software in der aktuellen Version zusammen mit den Ex- und Importvorlagen

**2. Vorbereiten der Installation:** Transferieren Sie die Software auf das Windowssystem auf dem auch die JTL-Wawi-Datenbank aktiv läuft.

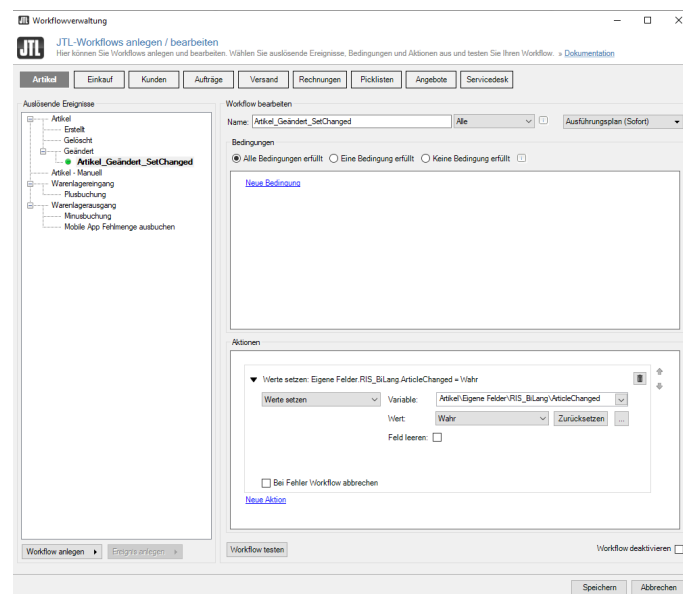
### 3. Eigenes Feld anlegen:

- Erstellen Sie ein 'Eigenes Feld' das als Indikator benutzt dafür verwendet werden kann, ob ein Artikel sich verändert hat oder nicht.
- Wählen Sie dazu zuerst den Menüpunkt 'Admin -> Eigene Felder'.
- Legen Sie nun die 'RIS\_BiLang' Feldgruppe an, in der das Feld 'ArticleChanged' eingetragen wird.
- Sie können außerdem unten links auswählen welche Warengruppen Sie übersetzen lassen möchten.
- Das Feld muss vom Datentyp 'Checkbox' sein, Anzeigeort und Beschreibung können frei gewählt werden.
- Bestätigen Sie Ihre Angaben, indem Sie die 'Speichern' Schaltfläche benutzen.

The screenshot shows a software window titled "Artikel" with a subtitle "Eigenes Feld anlegen / bearbeiten". Below the title bar, there is a small instruction: "Hier können Sie zusätzliche Artikel-Kategorieeigenschaften anlegen, um sie z.B. mit Feldern in Amazon zu verknüpfen. » Dokumentation". The window is divided into several sections. On the left, under "Eigenes Feld auswählen", there is a "Bereich:" dropdown set to "Artikel". Below it is a tree view showing a hierarchy: "Amazon-Allgemein", "Amazon-Sonstiges", "Komponenten", "RIS\_BiLang", and "ArticleChanged". At the bottom of this section are buttons for "Feld anlegen", "Löschen", "Gruppe anlegen", and "Gruppe bearbeiten". Below the tree view is a list of checkboxes for selecting product groups: "Warengruppe", "ohne Warengruppe", "Elektronik", "Ernährung", "Sportbekleidung", and "Sportgeräte". On the right, under "Eigenes Feld definieren", there is a yellow warning box with an exclamation mark icon and the text: "Dieses eigene Feld erscheint nicht als Eigenschaft für Artikel / Kategorien in Ihrem Shop. Es kann jedoch genau wie Funktionsattribute genutzt werden (max. 4000 Zeichen)". Below this, there are input fields for "Name:" (containing "ArticleChanged"), "Datentyp:" (a dropdown set to "Checkbox"), and "Anzeigeort:" (a dropdown set to "EigeneFelder"). There is also a "Beschreibung:" text area containing the text "Zeigt an ob ein Artikel seit dem letzten Übersetzungsvorgang verändert wurde". At the bottom right of the window are two buttons: "Speichern" and "Abbrechen".

**4. JTL-Wawi Workflow anlegen:** Nun muss das soeben erstellte Feld benutzt werden, dazu wählen Sie wieder 'Admin' und nun 'JTL-Workflows':

- Da das Feld immer dann auf Wahr gesetzt werden soll, wenn ein Artikel sich verändert hat, wird ein Workflow für das Ereignis 'Artikel - Geändert' erstellt.
- Die Bedingung kann verändert werden, es wird allerdings empfohlen keine Bedingung anzugeben.
- Erstellen Sie eine neue Aktion, indem Sie auf den entsprechenden Text klicken.
- Im Dropdown Menü wird die Option 'Werte setzen' ausgewählt und anschließend das 'ArticleChanged' als Variable gewählt.
- Bestätigen Sie Ihre Angaben, indem Sie die 'Speichern' Schaltfläche benutzen.



**5. Konfigurationsdatei anpassen:** Damit das Programm eine Verbindung zum JTL-Wawi Datenbankserver herstellen kann, muss die Konfigurationsdatei richtig beschrieben werden. Im Bereich `appSettings` finden Sie verschiedene Punkte.

`db_server` : Die IP-Adresse des Datenbankservers mit der Instanz der JTL-Wawi-Datenbank als Anhang, getrennt durch einen “\”.

`db_port` : Der Port unter dem die Datenbank angesprochen werden kann.

`db_mandant` : Der Name des Mandanten für den die Übersetzung aktiviert werden soll.

`db_user` : Der Benutzername zu einem Datenbankbenutzer (Standardwert 'sa').

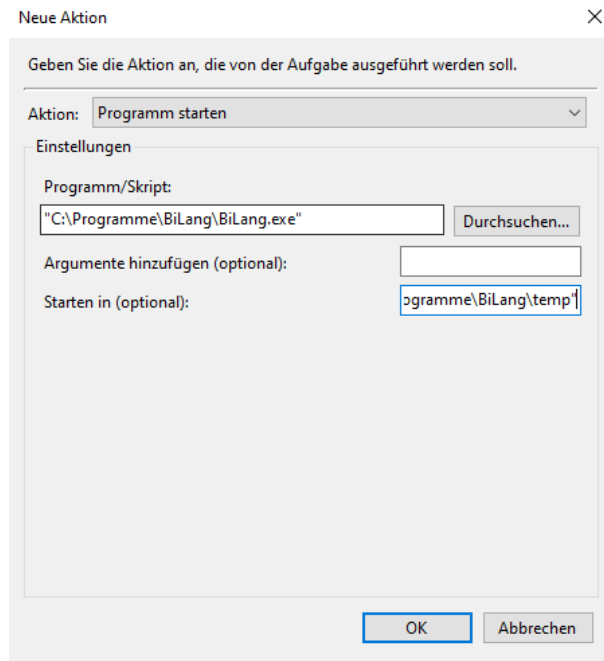
**db\_pass** : Das Passwort zum Datenbankbenutzernamen in db\_user (Standardwert 'sa04jT14').

**jtlAmeisePath** : Der absolute Pfad zur 'JTL-wawi-ameise.exe', welche im Installationsverzeichnis liegt (Standardwert 'C:\Programme\JTL-Software\JTL-wawi-ameise.exe').

Außerdem können die **appLanguages** nach Belieben ein- oder auskommentiert werden, um diese auch tatsächlich zu nutzen.

**6. Windows-Aufgabenplanung konfigurieren:** Um das Programm automatisiert starten zu lassen, verwenden Sie die Windows-Aufgabenplanung. Diese ermöglicht Programmstarts mit festen Intervallen.

- Dazu müssen Sie eine 'Aufgabe erstellen...'
- Name und Beschreibung ist frei wählbar
- Wechseln Sie in den Reiter 'Trigger' und legen einen neuen Trigger an



- Hier wird angegeben, wie oft das Programm eine Übersetzung anstoßen soll. Empfohlen wird eine Tägliche Ausführung, sowie manuelle Starts bei großen Veränderungen in den Stammdaten der JTL-Wawi.
- Nach eine Bestätigung mit OK, können Sie auf den Reiter 'Aktionen' wechseln.
- Hier erstellen Sie eine neue Aktion vom Typ 'Programm starten' und wählen die Bi-Lang.exe aus Ihrem Verzeichnis aus.
- Empfohlen wird in dem Feld 'Starten in' den absoluten Pfad des Verzeichnisses 'Bi-Lang/temp' auszuwählen.

Anschrift Prüfling:

Langer Quirin  
Baumannstraße 2  
92224 Amberg

Anschrift Ausbildungsbetrieb:

RIS-Development GmbH & Co. KG  
Siemensstraße 9  
93055 Regensburg**Protokoll der durchgeführten Projektarbeit**

Prüf-Nr.:

95230**Ausbildungsberuf:**Fachinformatiker für Anwendungsentwicklung**1. Arbeitszeit:**

1.1 Die vom Prüfungsteilnehmer kalkulierte Zeit entspricht der betrieblichen Kalkulation.

☒ ja ☐ neinWenn nein: Sie ist um 0 % höher 0 % niedriger

1.2 Das Projekt wurde vom Prüfungsteilnehmer in der kalkulierten Zeit komplett fertiggestellt (einschließlich eventueller Nacharbeit):

☒ ja ☐ neinWenn nein: Um 0 Stunden früher fertig geworden,0 Stunden länger gebraucht.**2. Ausführung:**

2.1 Wurde das Projekt entsprechend dem eingereichten Konzept ausgeführt?

☒ ja ☐ nein

Wenn nein: Welche Änderungen ergaben sich?

-

-

-

bitte weiter auf der Rückseite!

### Prüfungsteilnehmer:

Langer Quirin

Name / Vorname

RIS-Development GmbH & Co. KG

Firma

#### 2.2 Wurde das Projekt selbständig und ohne fremde Hilfe ausgeführt?

☒ ja

☐ nein

Wenn nein: Begründung und Umfang der Hilfestellung.

-  
-  
-

#### 2.3 Das Projekt konnte ohne Nacharbeit in einem einwandfreien Zustand übergeben werden.

☒ ja

☐ nein

Wenn nein: Begründung und Umfang der Nacharbeit.

-  
-  
-

### 3. Dokumentation:

#### 3.1 Die Dokumentation wurde vom Prüfungsteilnehmer selbständig und ohne fremde Hilfe erstellt.

☒ ja

☐ nein

Wenn nein: Welche Hilfestellung wurde gegeben?

-  
-  
-

#### 3.2 Die Dokumentation entspricht den betrieblichen Anforderungen.

☒ ja

☐ nein

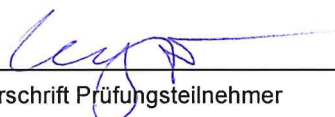
Wenn nein: Worin besteht die Abweichung?

-  
-  
-

Datum

27.05.2021

Unterschrift Prüfungsteilnehmer





RIS

Regensburger IT-Solutions

Web- & Software-Development GmbH & Co. KG

Siemensstraße 9, D-93049 Regensburg

Web: [www.ris-development.de](http://www.ris-development.de) Telefon: +49 941 2000 1250

E-Mail: [info@ris-development.de](mailto:info@ris-development.de) Fax: +49 941 2000 1251