

**Abschlussprüfung Sommer 2017**

Informatikkaufmann

Dokumentation zur betrieblichen Projektarbeit

# **AMAZON VERSANDBESTÄTIGUNGEN AUTOMATISIEREN**

**Webanwendung für den automatischen Versand von  
Versandbestätigungen für Amazon-Bestellungen**

**Auszubildender:**

Tobias Dalhof  
Rispenweg 22  
73479 Ellwangen



**Ausbildungsbetrieb:**

Alfa GmbH  
Dr.-Rudolf-Schieber-Str. 11-15  
73463 Westhausen

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis.....</b>	<b>III</b>
<b>1 Einleitung.....</b>	<b>1</b>
1.1 Projektbeschreibung .....	1
1.2 Projektziele .....	1
1.3 Projektumfeld .....	2
1.4 Prozessschnittstellen .....	2
<b>2 Projektplanung .....</b>	<b>3</b>
2.1 Zeit- und Ablaufplanung.....	3
2.2 Ressourcenplanung.....	4
<b>3 Analyse .....</b>	<b>5</b>
3.1 Ist-Analyse.....	5
3.2 Projektkosten.....	5
3.3 Kosten-Nutzen-Rechnung .....	6
3.4 Lastenheft.....	7
<b>4 Entwurf.....</b>	<b>7</b>
4.1 Architektur .....	7
4.2 Geschäftslogik.....	8
4.3 Datenbankentwurf .....	9
4.4 Pflichtenheft .....	10
<b>5 Umsetzung .....</b>	<b>10</b>
5.1 Datenbankmigrationen erstellen.....	10
5.2 Bestellinformationen speichern .....	11
5.3 Versandbestätigungen zu Amazon senden.....	11
5.4 Log-Datei per E-Mail senden.....	13
5.5 Benutzeroberfläche .....	14
5.6 Tests .....	14
<b>6 Abnahme und Einführung .....</b>	<b>15</b>
6.1 Abnahme.....	15

6.2	Serverinstallation.....	15
6.3	Produktivsetzung.....	15
<b>7</b>	<b>Kundendokumentation.....</b>	<b>16</b>
<b>8</b>	<b>Projektergebnisse.....</b>	<b>16</b>
8.1	Soll-Ist-Vergleich.....	16
8.2	Gemachte Erfahrungen .....	17
8.3	Ausblick .....	17
<b>A</b>	<b>Anlagen .....</b>	<b>i</b>
A.1	Flussdiagramm zur Ist-Analyse.....	i
A.2	Auszug aus dem Lastenheft.....	ii
A.3	Auszug aus dem Pflichtenheft.....	iii
A.4	Flussdiagramm zum Ablauf.....	v
A.5	Datenbankmigrationen .....	vi
A.5.1	Tabelle für Benutzer .....	vi
A.5.2	Tabelle für Bestellungen .....	vii
A.5.3	Tabelle für Versandbestätigungen .....	vii
A.6	Controller: Bestellinformationen speichern.....	viii
A.7	Job: Daten zu Amazon senden.....	x
A.8	Order Fulfillment XML Template .....	xii
A.9	Controller: Benutzeroberfläche.....	xii
A.10	Screenshots der Benutzeroberfläche.....	xiii
A.11	Inhaltsverzeichnis der Benutzerdokumentation.....	xvi
A.12	Auszug der Entwicklerdokumentation.....	xvii
A.13	Kopie des genehmigten Projektantrages.....	xviii
A.14	Persönliche Erklärung .....	xxi

## Abkürzungsverzeichnis

<b>Amazon MWS</b>	Amazon Marketplace Web Service
<b>API</b>	Application Programming Interface
<b>CSS</b>	Cascading Style Sheets
<b>ERP-System</b>	Enterprise Resource Planning System
<b>HTML</b>	Hypertext Markup Language
<b>MVC</b>	Model-View-Controller
<b>SMTP</b>	Simple Mail Transfer Protocol
<b>XML</b>	Extensible Markup Language

# 1 Einleitung

In der folgenden Projektdokumentation schildere ich, Tobias Dalhof, den Ablauf des IHK-Abschlussprojektes, welches ich im Rahmen meiner Ausbildung zum Informatikkaufmann durchgeführt habe. Der Ausbildungsbetrieb ist die Alfa GmbH, ein Großhändler für Baustoffe. Zu den Kunden gehören hauptsächlich Handwerksfirmen und Industriebetriebe, aber auch zunehmend Privatpersonen. Derzeit sind bei der Alfa GmbH am Standort in Westhausen 29 Mitarbeiter beschäftigt.

## 1.1 Projektbeschreibung

Der Amazon Marktplatz ist bei der Alfa GmbH ein stark wachsender Vertriebskanal. Derzeit ist es so, dass ein Mitarbeiter die Versandbestätigungen für die Kunden manuell über das Amazon Seller Central versendet. Bei diesem Prozess notiert sich der Mitarbeiter von jeder Bestellung die Sendungsverfolgungsnummer aus dem ERP-System und trägt diese dann von Hand in die Versandbestätigung im Amazon Seller Central ein.

Da dies fehleranfällig und zeitaufwendig ist, soll dieser Arbeitsschritt automatisiert werden.

Die Alfa GmbH benötigt eine Webanwendung, welche die Versandbestätigungen von Amazon-Bestellungen mit Hilfe des Amazon MWS (Amazon Marketplace Web Services) automatisch verarbeitet. Die Sendeverfolgungsnummern müssen aus dem ERP-System der Alfa GmbH automatisch in die Versandbestätigungen von Amazon eingetragen werden.

Es soll ermittelt werden, wie viel Zeit und Kosten durch die Automatisierung eingespart werden können. Die Webanwendung soll in PHP programmiert werden und erweiterbar sein. Nach erfolgreichen Tests soll die Webanwendung auf einem Server installiert und in Betrieb genommen werden.

## 1.2 Projektziele

Für die Alfa GmbH soll eine erweiterbare Webanwendung entwickelt werden, welche die Versandbestätigungen von Amazon-Bestellungen automatisch mit der dazugehörigen Sendungsverfolgungsnummer an Amazon übermittelt.

Durch die Automatisierung sollen die Mitarbeiter der Alfa GmbH entlastet und Fehler, die bei der manuellen Eingabe von Sendungsverfolgungsnummern entstehen können, vermieden werden.

Die dadurch gewonnene Zeit sollen die Mitarbeiter für andere Aufgaben einsetzen, wie zum Beispiel die Optimierung von Amazon-Produktseiten oder die Verbesserung des Kundenservice, um das Wachstum auf dem Amazon-Marktplatz weiter voranzutreiben.

## 1.3 Projektumfeld

Der Auftraggeber des Projekts ist die IT-Abteilung der Alfa GmbH.

Für die erfolgreiche Durchführung dieses Projekts ist die regelmäßige Kommunikation mit der IT-Leitung sowie dem verantwortlichen Mitarbeiter, der die Amazon-Bestellungen bearbeitet, erforderlich.

Die IT-Abteilung der Alfa GmbH beschäftigt sich hauptsächlich mit E-Commerce-Themen (u. a. Weiterentwicklung und Optimierung von Online-Shops) und mit dem ERP-System (GDI<sup>1</sup>).

Zu den Aufgaben des verantwortlichen Mitarbeiters aus dem internen Amazon-Team der Alfa GmbH gehören die Bearbeitung von Bestellungen, die Bearbeitung von Kundenanfragen, die Erstellung und Optimierung von Produktseiten auf Amazon sowie die Planung und Verwaltung von Werbekampagnen auf Amazon.

## 1.4 Prozessschnittstellen

Die Amazon- Bestellinformationen kommen aus dem **ERP-System** der Alfa GmbH und sollen in einer Datenbank der **Webanwendung** zwischengespeichert werden. Die Webanwendung soll die Daten weiter zum **Amazon Merchant Web Service** senden.

Herr Michael Beyer, mein Ausbilder und Leiter der IT-Abteilung, ist mein Ansprechpartner in Sachen ERP-System. Er legt außerdem die Anforderungen an die Webanwendung fest und führt das Code-Review durch.

Herr Volker Herzog, der verantwortliche Mitarbeiter für den Verkauf auf dem Amazon Marktplatz, hilft mir bei Fragen zum Ablauf bei der Kommissionierung von Amazon-Bestellungen weiter.

---

<sup>1</sup> Vgl. <http://www.gdi.de/startseite.html>

## 2 Projektplanung

### 2.1 Zeit- und Ablaufplanung

Zur Projekterstellung stehen mir insgesamt 35 Stunden zur Verfügung. Die Zeit soll optimal verteilt werden.

<b>Analyse</b>		<b>5 Stunden</b>
1. Ist-Analyse		1 Stunde
2. Projektkosten		1 Stunde
3. Kosten-Nutzen-Rechnung		3 Stunden
<b>Entwurf</b>		<b>6 Stunden</b>
1. Datenbankentwurf		1 Stunde
2. Planung der Geschäftslogik		3 Stunden
3. Pflichtenheft erstellen		2 Stunden
<b>Umsetzung</b>		<b>12 Stunden</b>
1. Datenbankmigrationen erstellen		0,5 Stunden
2. Bestellinformationen speichern		1,5 Stunden
3. Versandbestätigungen zu Amazon senden		8 Stunden
4. Log-Datei per Mail senden		1 Stunde
5. Benutzeroberfläche erstellen		1 Stunde
<b>Abnahme</b>		<b>2 Stunden</b>
1. Abnahme durch die IT-Leitung		0,5 Stunden
2. Serverinstallation und Produktivsetzung		1,5 Stunden
<b>Dokumentationen erstellen</b>		<b>10 Stunden</b>
1. Projektdokumentation erstellen		8 Stunden
2. Kundendokumentation erstellen		2 Stunden
<b>Gesamtzeit</b>		<b>35 Stunden</b>

Tabelle 1: Zeit- und Ablaufplanung

## 2.2 Ressourcenplanung

Das Projekt wird an meinem Arbeitsplatzrechner bei der Alfa GmbH umgesetzt. Auf dem PC läuft Windows 10 Pro als Betriebssystem. Eine lokale Entwicklungsumgebung (Ubuntu 16.04 Server und diverser Software) wurde mit Virtualbox und Vagrant<sup>2</sup> bereits im Vorfeld installiert.

Zum Schreiben von PHP-Code wird die PHP Entwicklungsumgebung PhpStorm 2017 eingesetzt – PhpStorm 2017 war ebenfalls vorinstalliert.

Die Versionierung des Codes erfolgt mit Git und Bitbucket.

Für die Erstellung des Datenbankmodells wird MySQL Workbench verwendet.

### Hardware

- Desktop-PC

### Software

- Windows 10 Pro – Betriebssystem
- PhpStorm 2017 – PHP Entwicklungsumgebung
- MySQL Workbench – Zum Erstellen des Datenbankmodells
- Laravel Homestead – Lokale Entwicklungsumgebung
- Laravel 5.4 – PHP Web-Framework
- Git und Bitbucket – Versionsverwaltung

### Personal

- Abteilungsleiter IT – Festlegung der Anforderungen und Code-Review
- Verantwortlicher Mitarbeiter für Amazon-Bestellungen – Steht bei Fragen zur Verfügung
- Auszubildender Informatikkaufmann – Projektumsetzung

---

<sup>2</sup> Vgl. <https://laravel.com/docs/5.4/homestead>



## 3 Analyse

### 3.1 Ist-Analyse

Zum Tagesbeginn werden alle offenen Amazon-Bestellungen in der Warenwirtschaft erfasst und die Kommissionsscheine ausgedruckt. Alle Bestellungen, die im Verlauf des Tages eingehen, müssen ebenfalls in der Warenwirtschaft erfasst und abgearbeitet werden.

Sobald die Ware im Lager gepickt und gepackt wurde, werden am Lager-PC die Belege (Versandetikett, Lieferschein und Rechnung) ausgedruckt.

Die Sendungsverfolgungsnummern, welche auf den jeweiligen Versandetiketten stehen, werden bei jeder Bestellung von Hand auf den Kommissionsschein geschrieben.

Alle fertigen Bestellungen werden im Versandbereich des Lagers abgestellt und am frühen Abend vom Logistikpartner der Alfa GmbH abgeholt.

Der letzte Schritt besteht darin, die zuvor notierten Sendungsverfolgungsnummern im Amazon Seller Central bei jeder Bestellung einzutragen und den Versand zu bestätigen. Dieser Schritt wird immer am Ende des Tages erledigt.

Zur Verdeutlichung des Bearbeitungsprozesses von Amazon-Bestellungen befindet sich in Anlage 1 ein Flussdiagramm.

### 3.2 Projektkosten

Die aufgeführten **Personalkosten** sind frei gewählt. Ein Auszubildender erhält somit eine Stundenpauschale von 15 € und ein Mitarbeiter 30 € (brutto). Für die **Ressourcennutzung** wurde eine Stundenpauschale von 10 € angesetzt.

Die Kosten werden in der Tabelle 2 aufgeführt.

Vorgang	Mitarbeiter	Zeit	Personal <sup>3</sup>	Ressourcen <sup>4</sup>	Gesamt
Entwicklungskosten	1 Auszubildender	35 h	525,00 €	350,00 €	<b>875,00 €</b>
Fachgespräche	2 Mitarbeiter	2 h	60,00 €	40,00 €	<b>100,00 €</b>
Code-Review	1 Mitarbeiter	1 h	30,00 €	10,00 €	<b>40,00 €</b>
Abnahme	1 Auszubildender 1 Mitarbeiter	1 h	45,00 €	20,00 €	<b>65,00 €</b>
Gesamtkosten					<b>1.080,00 €</b>

Tabelle 2: Personalkosten

Im ersten Jahr fallen einmalig die Kosten für die **Projektumsetzung** in Höhe von 1.080,00 € an. Die jährlichen **Serverkosten** belaufen sich auf 179,88 €<sup>5</sup>.

Zusätzlich werden pro Jahr 6 Stunden **Wartungszeit** eingeplant. Die Wartungsarbeiten sollen von einem Mitarbeiter der Alfa GmbH durchgeführt werden.

### 3.3 Kosten-Nutzen-Rechnung

Die **Zeitersparnis** durch die Automatisierung beläuft sich **pro Bestellung auf ca. eine Minute**.

Die durchschnittliche Anzahl der Bestellungen des letzten Jahres belief sich auf **300 Bestellungen pro Monat**. Die Geschäftsleitung erwartet eine **jährliche Steigerung** der Verkäufe auf dem Amazon-Marktplatz von **mindestens 10 %**.

Jährliche Lohnsteigerungen werden **nicht** berücksichtigt. Kosten und Nutzen werden in der Tabelle 3 gegenübergestellt.

<sup>3</sup> Personalkosten pro Vorgang = Mitarbeiteranzahl \* Bruttostundenlohn \* Zeit

<sup>4</sup> Ressourcen = Anzahl Mitarbeiter \* Zeit \* Stundenpauschale Ressourcennutzung

<sup>5</sup> Jährliche Serverkosten = Monatliche Kosten (14,99 €) \* 12

Kategorie	2017	2018	2019	2020
<b>Nutzen</b>				
<b>Erwartete Bestellungen<sup>6</sup></b>	3600	3960	4356	4792
<b>Zeitersparnis<sup>7</sup> (gerundet)</b>	60 h	66 h	73 h	80 h
<b>Stundenbruttolohn</b>	30,00 €	30,00 €	30,00 €	30,00 €
<b>Einsparungen<sup>8</sup></b>	1.800,00 €	1.980,00 €	2.190,00 €	2.400,00 €
<b>Kosten</b>				
<b>Serverkosten<sup>9</sup></b>	179,88 €	179,88 €	179,88 €	179,88 €
<b>Projektumsetzung</b>	1.080,00 €	-	-	-
<b>Wartungskosten<sup>10</sup></b>	180,00 €	180,00 €	180,00 €	180,00 €
<b>Gesamtkosten</b>	1.439,88 €	359,88 €	359,88 €	359,88 €
<b>Gesamteinsparungen</b>	<b>360,12 €</b>	<b>1.620,12 €</b>	<b>1.830,12 €</b>	<b>2.040,12 €</b>

Tabelle 3: Gegenüberstellung von Kosten und Nutzen

### 3.4 Lastenheft

Es wurde zusammen mit der IT-Leitung ein Lastenheft erstellt. Im Lastenheft wurden die Anforderungen an die Webanwendung festgelegt. Ein Auszug des Lastenhefts befindet sich in Anlage 2.

## 4 Entwurf

### 4.1 Architektur

Als solide Grundlage der Webanwendung soll das PHP Webframework **Laravel** verwendet werden. Laravel ist ein weit verbreitetes und modernes Framework, welches das **MVC-Muster** (Model-View-Controller) verwendet. Da ich bereits mit diesem Framework gearbeitet habe, muss ich mich nicht einarbeiten.

<sup>6</sup> Erwartete Bestellungen = Durchschnitt d. Bestellung im Vorjahr + 10 % erwarteter Wachstum

<sup>7</sup> Zeitersparnis = Anzahl der Bestellungen \* Zeitersparnis pro Bestellung (1 Minute)

<sup>8</sup> Einsparungen = Zeitersparnis \* Stundenbruttolohn

<sup>9</sup> Jährliche Serverkosten = Monatliche Serverkosten (14,99 €) \* 12

<sup>10</sup> Wartungskosten = Geplante Wartungszeit pro Jahr (6 h) \* Stundenbruttolohn (30 €)

Das MVC-Muster ist ein Architekturmuster zur Trennung von Software in drei Komponenten: Model, View und Controller. Durch eine Trennung der drei Komponenten werden **Änderungen** und **Erweiterungen erleichtert** und die **Wiederverwendbarkeit** der einzelnen **Komponenten** ermöglicht.<sup>11</sup>

### Model

Ein Model kann als Set von Objekten gesehen werden, denen es erlaubt ist mit der Datenbank zu kommunizieren. In einem Model kann Geschäftslogik definiert werden.

### View

Views sind verantwortlich für das Rendern von Inhalten auf Benutzerebene. Views bestehen nicht aus Geschäftslogik und kommunizieren nicht direkt mit der Datenbank!

### Controller

In einem Controller wird die Geschäftslogik von einer Anwendung definiert. Ein Controller holt beispielsweise auch Daten von einem Model und gibt diese dann weiter an den View.

## 4.2 Geschäftslogik

Wie in Abschnitt 1.4 (Prozessschnittstellen) erklärt wurde, kommen die Bestellinformationen aus dem ERP-System der Alfa GmbH. An allen Wochentagen um jeweils 19 Uhr führt das ERP-System einen „**Tagesabschluss**“ durch. Beim Tagesabschluss werden verschiedene Bestellungsverarbeitungsprozesse durchlaufen. An dieser Stelle kann auch für jede Amazon-Bestellung eine GET-Anfrage mit Query-String an die Webanwendung gestartet werden.

Die Webanwendung stellt eine URL zur Verfügung, auf der die Anfragen des ERP-Systems gesendet werden können. Bei jeder Anfrage zur Webanwendung müssen die Amazon-Bestellungsnummer (`amazon_order_id`), der Versanddienstleister (`carrier_name`) sowie die Sendungsverfolgungsnummer (`shipper_tracking_number`) in Form eines Query-Strings mitgesendet werden.

Aus dem Query-String des Requests wird dann eine Bestellung mit einer dazugehörigen Versandbestätigung erstellt und in der Datenbank gespeichert.

Aus der Perspektive des MVC-Musters stellt eine Bestellung (`Order`) und eine Versandbestätigung (`DeliveryConfirmation`) jeweils ein Model dar.

---

<sup>11</sup> Vgl. Model View Controller [https://de.wikipedia.org/wiki/Model\\_View\\_Controller](https://de.wikipedia.org/wiki/Model_View_Controller)

Die Logik für das Speichern von Bestellungen und Versandbestätigungen wird in einem Controller (`OrderFulfillmentFeedController`) definiert.

Die Webanwendung soll ebenfalls an allen Wochentagen um 19:30 Uhr einen Job ausführen, der alle Versandbestätigungen in Form eines XML-Feeds<sup>12</sup> zusammenfasst und zu Amazon sendet.

Nachdem die Daten zu Amazon gesendet wurden soll nun eine Log-Datei an `log@alfa-direkt.de` gesendet werden.

Die Benutzeroberfläche der Webanwendung soll manuelle Eingriffe in das System ermöglichen.

### 4.3 Datenbankentwurf

Für die Webanwendung werden drei Tabellen benötigt. Die Tabelle **users** dient zum Speichern der Benutzer. In der Tabelle **orders** sollen die Bestellinformationen gespeichert werden. In der Tabelle **delivery\_confirmations** sollen die Versandbestätigungen gespeichert werden.

Eine Bestellung hat demnach eine Versandbestätigung und eine Versandbestätigung gehört zu einer Bestellung (1:1 Beziehung).

Jede Versandbestätigung soll einen Status bekommen, um zum Beispiel besser mit fehlenden oder fehlerhaften Informationen umgehen zu können.

Die Abbildung 1 zeigt das Tabellenmodell.

---

<sup>12</sup> Vgl. „Ship and Confirm Shipment (and get paid) - Order Fulfillment“, Seite 43 [https://images-na.ssl-images-amazon.com/images/G/02/rainier/help/XML\\_Documentation\\_Intl\\_V158772716\\_.pdf](https://images-na.ssl-images-amazon.com/images/G/02/rainier/help/XML_Documentation_Intl_V158772716_.pdf)

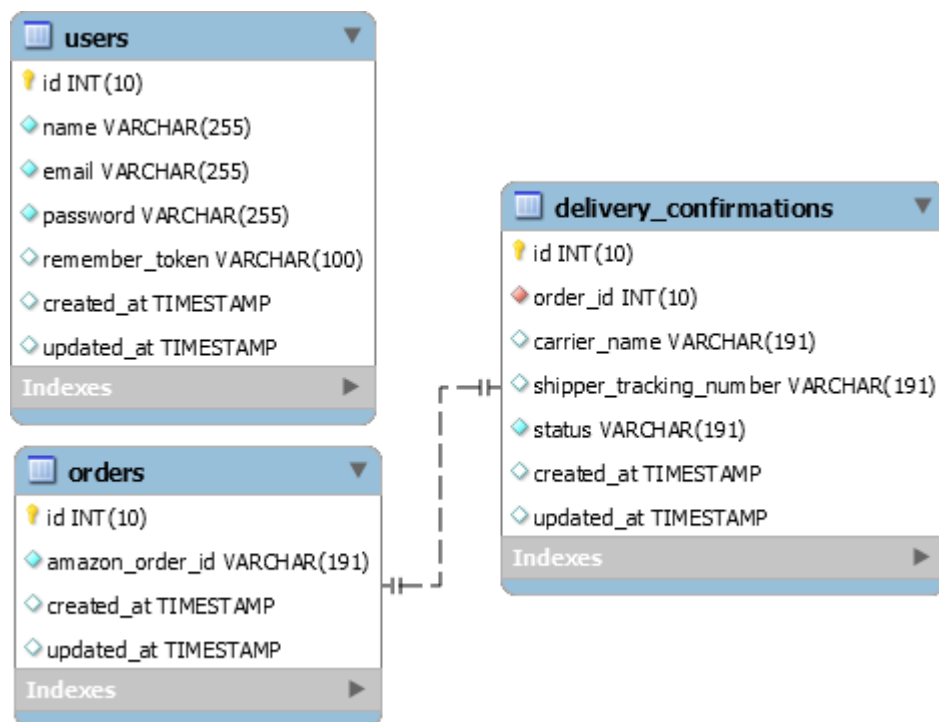


Abbildung 1: Tabellenmodell

## 4.4 Pflichtenheft

Es wurde ein Pflichtenheft erstellt, das die Umsetzung der Anforderungen aus dem Lastenheft umfassend beschreibt. Ein Auszug des Pflichtenheftes befindet sich in Anlage 3.

# 5 Umsetzung

Um den Ablauf der Webanwendung besser darzustellen wurde ein Flussdiagramm (siehe Anlage 4) erstellt.

## 5.1 Datenbankmigrationen erstellen

In Laravel werden neue Datenbanktabellen mit Migrationsdateien erstellt. Auch für jede Änderung an einer Tabelle wird eine neue Migration erstellt. Datenbankmigrationen können als eine Art von Versionskontrolle für die Datenbank gesehen werden.

Für die Webanwendung wurden drei Migrationen (siehe Anlage 5) erstellt: `create_users_table`, `create_orders_table` und `create_delivery_confirmations_table`. Um die Migrationen in Datenbanktabellen umzuwandeln, muss auf dem Server der Befehl `php artisan migrate` ausgeführt werden.

## 5.2 Bestellinformationen speichern

Zuerst wurde eine Route<sup>13</sup> erstellt (siehe Abbildung 2). Mit einer Route legt man den Controller mit einer Methode fest, der auf eine bestimmte URL antworten soll. Auf diese Route wird das ERP-System der Alfa GmbH die GET-Requests ausführen. Der Route wurde ein Controller (OrderFulfillmentFeedController, siehe Anlage 6) und eine Methode (save) zugewiesen.

Die URL zum Übermitteln von Bestellinformationen lautet also wie folgt:

[http://beispiel.de/feed/order-fulfillment/save?amazon\\_order\\_id=12345&carrier\\_name=GLS&shipper\\_tracking\\_number=12345](http://beispiel.de/feed/order-fulfillment/save?amazon_order_id=12345&carrier_name=GLS&shipper_tracking_number=12345)

```
// Feed Routes
Route::group(['prefix' => 'feed', 'as' => 'feed.'], function () {

    // Feed Status
    Route::get('status', 'FeedStatusController@index')->name('status');
    Route::get('status/show/{feedId}', 'FeedStatusController@show')->name('status.get');

    // Order Fulfillment
    Route::group(['prefix' => 'order-fulfillment', 'as' => 'order-fulfillment.'], function () {
        Route::get('save', 'OrderFulfillmentFeedController@save')->name('save');
    });

});
```

Abbildung 2: Route zum Speichern der Versandbestätigungen

Die Methode save liest die Daten aus dem Query-String (amazon\_order\_id, carrier\_name, shipper\_tracking\_number) und erstellt daraus eine neue Bestellung (Order) sowie eine dazugehörige Versandbestätigung (DeliveryConfirmation).

Sollte der Versanddienstleister (carrier\_name) oder die Sendungsverfolgungsnummer (shipper\_tracking\_number) fehlen, so erhält die Versandbestätigung den Status data\_incomplete. Wenn die Daten vollständig sind lautet der Status data\_complete.

## 5.3 Versandbestätigungen zu Amazon senden

Um mit dem Amazon Merchant Web Service kommunizieren zu können wird das Laravel Package amazon-mws-laravel<sup>14</sup> verwendet. Bevor das Package konfiguriert werden konnte, musste von Amazon ein API-Zugang für das Händlerkonto der Alfa GmbH beantragt werden. API-Zugänge müssen im Amazon Seller Central beantragt werden.

<sup>13</sup> Vgl. <https://laravel.com/docs/5.4/routing>

<sup>14</sup> Vgl. <https://github.com/sonnenglas/amazon-mws-laravel>

Zum allgemeinen Verständnis von Amazon MWS musste ich mich zuerst in die Entwicklerdokumentation<sup>15</sup> einlesen. Amazon beschreibt seinen Marketplace Web Service mit folgenden Worten:

„Amazon Marketplace Web Service (Amazon MWS) ist ein integriertes Web Service-API, das Amazon-Verkäufer unter anderem beim programmatischen Datenaustausch zur Katalog- und Bestellverwaltung unterstützt. Ein programmatischer Datenaustausch ermöglicht es Verkäufern eine voll automatisierte Integration des eigenen Systems mit den Amazon Systemen zu realisieren. Somit lassen sich Verbesserungen in der Verkaufseffizienz realisieren und die Abwicklungsdauer für Kunden verbessern. Die XML-Datenintegration mit Amazon ermöglicht eine Verkaufsautomatisierung, die Verkäufern hilft, ihr Geschäft zu erweitern.“<sup>16</sup>

Um die Versandbestätigungen an Amazon senden zu können, müssen die Daten im XML-Format<sup>17</sup> vorliegen.

Es wurde ein Job<sup>18</sup> (`HandleOrderFulfillmentFeed`, siehe Anlage 7) erstellt, der folgende Aufgaben erledigt:

1. Hole alle Versandbestätigungen mit dem Status `data_complete`.
2. Erstelle und speichere eine XML-Datei (XML-Template siehe Anlage 8) mit den Daten der Versandbestätigungen.
3. Sende die XML-Datei zu Amazon.
4. Setze den Status der Versandbestätigungen auf `submitted`.
5. Sende eine Log-Datei per E-Mail.

Der Job soll an Wochentagen um 19:30 Uhr ausgeführt werden. Dazu musste im Laravel Task Scheduler<sup>19</sup> eine neue Schedule eingetragen werden (siehe Abbildung 6).

---

<sup>15</sup> Entwicklerdokumentation unter <https://developer.amazonservices.de/gp/mws/docs.html>

<sup>16</sup> Vgl. <https://developer.amazonservices.de/>

<sup>17</sup> Vgl. „Selling on Amazon – Guide to XML“ – Seite 43 („Ship and Confirm Shipment (and get paid“) - Order Fulfillment) [https://images-na.ssl-images-amazon.com/images/G/02/rainier/help/XML\\_Documentation\\_Intl\\_V158772716\\_.pdf](https://images-na.ssl-images-amazon.com/images/G/02/rainier/help/XML_Documentation_Intl_V158772716_.pdf)

<sup>18</sup> Vgl. <https://laravel.com/docs/5.4/queues#creating-jobs>

<sup>19</sup> Mehr zum Scheduler unter <https://laravel.com/docs/5.4/scheduling>



```
/**
 * Define the application's command schedule.
 *
 * @param \Illuminate\Console\Scheduling\Schedule $schedule
 * @return void
 */
protected function schedule(Schedule $schedule)
{
    // Log::info('Test from kernel!');
    $schedule->call(function () {
        Log::info('HandleOrderFulfillmentFeed Job started.');
```

```
        dispatch(new HandleOrderFulfillmentFeed);
        Log::info('HandleOrderFulfillmentFeed Job finished.');
```

```
    }->weekdays()->dailyAt('19:30');
```

```
    }
```

Abbildung 3: Schedule zum Ausführen des Jobs

## 5.4 Log-Datei per E-Mail senden

Der Versand der Log-Dateien wurde mit Hilfe der Laravel-Mailing-Funktionen<sup>20</sup> erstellt. Zur Konfiguration wurden die SMTP-Daten des Mailservers der Alfa GmbH benötigt. In Abbildung 5 ist ein Screenshot der E-Mail zu sehen. In Abbildung 6 ist der Inhalt einer fehlerfreien Log-Datei abgebildet.

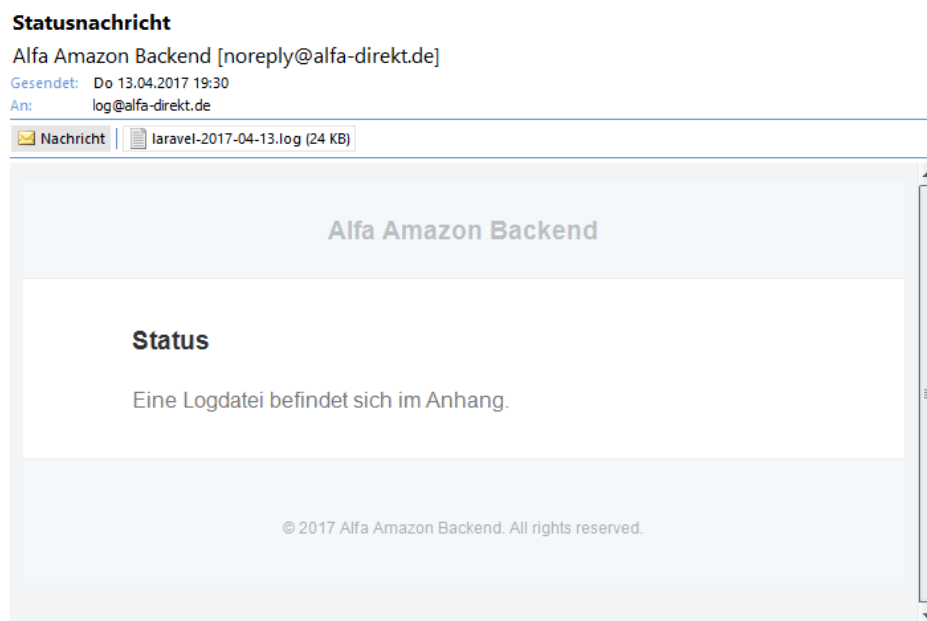


Abbildung 4: Statusnachricht im E-Mail-Postfach

<sup>20</sup> Laravel Mail Dokumentation unter <https://laravel.com/docs/5.4/mail>

```
[2017-04-13 19:30:02] local.INFO: HandleOrderFulfillmentFeed Job started.
[2017-04-13 19:30:02] local.INFO: Processing Order #304-9351583-0901164
[2017-04-13 19:30:02] local.INFO: Processing Order #306-2696289-4866767
[2017-04-13 19:30:02] local.INFO: Processing Order #305-3768450-7389161
[2017-04-13 19:30:02] local.INFO: Processing Order #306-8088809-9312354
[2017-04-13 19:30:02] local.INFO: Processing Order #306-4909202-0090718
[2017-04-13 19:30:02] local.INFO: Processing Order #306-3079730-4643539
[2017-04-13 19:30:02] local.INFO: Processing Order #303-1003850-7868368
[2017-04-13 19:30:02] local.INFO: Processing Order #302-8233487-7939568
[2017-04-13 19:30:02] local.INFO: Processing Order #306-4067372-2280322
[2017-04-13 19:30:02] local.INFO: Processing Order #303-3318594-9429169
[2017-04-13 19:30:02] local.INFO: Processing Order #302-0993552-9442755
[2017-04-13 19:30:02] local.INFO: Processing Order #306-4988083-4164335
[2017-04-13 19:30:02] local.INFO: Processing Order #304-7532107-6701942
[2017-04-13 19:30:02] local.INFO: Processing Order #303-6852521-1979526
[2017-04-13 19:30:02] local.INFO: Processing Order #306-4254856-8971527
[2017-04-13 19:30:02] local.INFO: Processing Order #306-7419492-6120335
[2017-04-13 19:30:02] local.INFO: Processing Order #028-1057459-8961957
[2017-04-13 19:30:02] local.INFO: Processing Order #306-2220557-1551539
[2017-04-13 19:30:02] local.INFO: Processing Order #306-2964489-2907527
[2017-04-13 19:30:02] local.INFO: Processing Order #303-8591886-6855509
[2017-04-13 19:30:02] local.INFO: Making request to Amazon: SubmitFeed
[2017-04-13 19:30:02] local.INFO: Response OK!
[2017-04-13 19:30:02] local.INFO: Successfully submitted feed #51015017269 (_POST_ORDER_FULFILLMENT_DATA_)
[2017-04-13 19:30:02] local.INFO: 20 orders processed.
```

Abbildung 5: Inhalt von einer Log-Datei ohne Fehlermeldungen

## 5.5 Benutzeroberfläche

Die Benutzeroberfläche wurde mit Hilfe von Bootstrap<sup>21</sup> (Version 3) erstellt. Bootstrap ist ein HTML, CSS und JavaScript Framework. Eine Einarbeitung in Bootstrap war wegen fundierten Vorkenntnissen nicht erforderlich.

Die Benutzeroberfläche bietet folgende Funktionalitäten:

- Login und Logout
- Übersichtsseite der Versandbestätigungen
- Versandbestätigungen erstellen, ändern und löschen
- Versandbestätigungen manuell zu Amazon senden

Screenshots zur Benutzeroberfläche sind in der Anlage 10 zu finden.

## 5.6 Tests

Um sicherzustellen, dass die Versandbestätigungen korrekt zu Amazon gesendet und verarbeitet werden, habe ich einzelne Bestellungen über die Benutzeroberfläche manuell eingetragen und abgesendet. Im Amazon Seller Central konnte ich dann überprüfen, ob alles korrekt funktioniert.

---

<sup>21</sup> Bootstrap Framework <http://getbootstrap.com/>

## 6 Abnahme und Einführung

### 6.1 Abnahme

Am Ende des Projekts präsentierte ich der IT-Leitung die Projektergebnisse. Alle gestellten Anforderungen (siehe Anlage 10 Lastenheft) an die Webanwendung konnten zur vollsten Zufriedenheit der Alfa GmbH erfüllt werden. Es ist noch anzumerken, dass die IT-Leitung bei jedem Schritt in der Umsetzungsphase über den gegenwärtigen Stand der Entwicklung informiert wurde. Somit verkürzte sich die Dauer der Abnahme.

Abschließend wurde noch ein Code-Review durch Herr Beyer durchgeführt.

### 6.2 Serverinstallation

Die Webanwendung wurde auf einem kleinen Server von DomainFactory mit Ubuntu 16.04 als Betriebssystem installiert. Eine Installationsanleitung mit Serveranforderungen wurde in der Entwicklerdokumentation (siehe Anlage 12) hinterlegt.

Es wurden noch einmal alle Funktionen getestet, um einen reibungslosen Produktivbetrieb zu gewährleisten.

**Hinweis:** Das Projekt wird mit Git und einem privaten Repository von Bitbucket.com verwaltet. Zur Bereitstellung des Codes muss das Repository kopiert werden. Wenn an der Webanwendung Änderungen vorgenommen wurden, dann müssen diese folglich vom Repository geladen werden.

### 6.3 Produktivsetzung

Sobald im ERP-System eine Abfrage zum Übergeben der Bestellinformationen an die Webanwendung (siehe 5.2) eingefügt wird, befindet sich die Webanwendung im Produktivzustand.

Die Abfrage wurde von Herrn Beyer eingebaut. Am selben Tag erfolgte ein Test im Produktivsystem. Das ERP-System übermittelte um 19 Uhr alle Bestellinformationen an die Webanwendung. Um 19:30 wurden alle Versandbestätigungen zu Amazon gesendet und korrekt verarbeitet.

Im Verlauf der nächsten Wochen soll genau beobachtet werden, ob die Webanwendung weiterhin korrekt funktioniert. Die gesendeten Log-Dateien sollen uns dabei unterstützen.

## 7 Kundendokumentation

Es wurde eine Benutzer- und Entwicklerdokumentation erstellt.

### Benutzerdokumentation

Die Benutzerdokumentation wurde im Intranet der Alfa GmbH veröffentlicht. In dieser Dokumentation wird die Verwendung der Benutzeroberfläche beschrieben. Ein Screenshot des Inhaltsverzeichnisses der Benutzerdokumentation ist in der Anlage 11 zu finden.

### Entwicklerdokumentation

Die Entwicklerdokumentation wurde in einer Readme-Datei im Bitbucket-Repository gespeichert. Dort findet man technische Informationen über die Webanwendung sowie eine Installationsanleitung. Ein Screenshot der Entwicklerdokumentation ist in der Anlage 12 zu finden.

## 8 Projektergebnisse

### 8.1 Soll-Ist-Vergleich

Im Soll-Ist-Vergleich wird die geplante Zeit von Abschnitt 2.1 der tatsächlich verbrauchten Zeit gegenübergestellt.

In der Tabelle 4 ist festzustellen, dass es beim Entwurf und bei der Umsetzung kleine Abweichungen gegeben hat, die aber wieder untereinander kompensiert werden konnten. Die Einarbeitung in Amazon MWS erforderte aufgrund der großen Menge an Informationen etwas mehr Zeit als erwartet.

Kategorie	Soll	Ist	Differenz
Analyse	5 Stunden	5 Stunden	0 Stunden
Entwurf	6 Stunden	5 Stunden	- 1 Stunde
Umsetzung	12 Stunden	13 Stunden	+ 1 Stunde
Abnahme	2 Stunden	2 Stunden	0 Stunden
Dokumentationen	10 Stunden	10 Stunden	0 Stunden
<b>Gesamt</b>	<b>35 Stunden</b>	<b>35 Stunden</b>	<b>0 Stunden</b>

Tabelle 4: Soll-Ist-Vergleich

## **8.2 Gemachte Erfahrungen**

Bei der Umsetzung des Projekts konnte ich wertvolle Erfahrungen mit der Amazon Merchant Web Service API sammeln. Neue Themen waren für mich die Automatisierung mit Hilfe des Laravel Schedulers sowie das Versenden von E-Mails über eine Webanwendung. Ich konnte mein allgemeines Wissen über die Extensible Markup Language (XML) verbessern.

## **8.3 Ausblick**

Die Webanwendung soll in den kommenden Monaten verbessert und erweitert werden. Weil die Kommissionierung von Amazon-Bestellungen noch immer ein hohes Maß an manueller Arbeit erfordert, wird bereits über den automatischen Versand von Rechnungen nachgedacht.

Wie bereits in Abschnitt 6.3 erwähnt, soll die Webanwendung in den kommenden Wochen weiter im Produktivbetrieb getestet und nach möglichen Fehlern gesucht werden.

## A Anlagen

### A.1 Flussdiagramm zur Ist-Analyse

**Hinweis:** Die in Rot markierte Prozesse sollen durch dieses Projekt eliminiert werden.

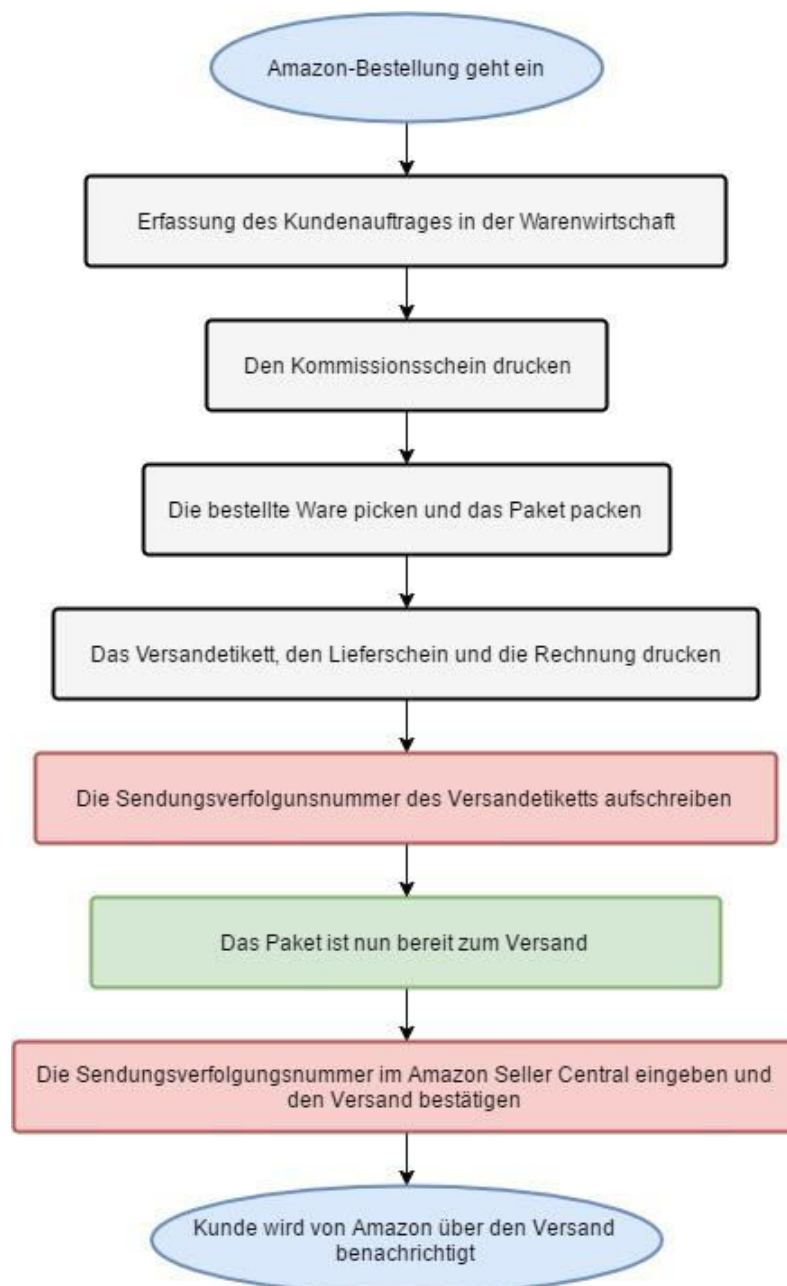


Abbildung 6: Bearbeitungsprozess von Amazon-Bestellungen

## **A.2 Auszug aus dem Lastenheft**

Folgende Anforderungen müssen erfüllt werden:

### **Erweiterbarkeit**

Die Webanwendung soll in der Zukunft um weitere Funktionen erweitert werden. Es muss also eine Basis geschaffen werden, die erweiterbar ist.

### **Bestellinformationen zur Webanwendung senden und speichern**

In der Webanwendung muss eine Schnittstelle geschaffen werden, mit der es möglich ist Bestellinformationen zu empfangen, verarbeiten und in einer Datenbank zu speichern. Zum Speichern der Versandbestätigungen muss eine URL zum Senden von HTTP-GET-Requests bereitgestellt werden. Aus den mitgelieferten Query-Strings der Requests, welche die relevanten Bestellinformationen enthalten, sollen in einer Datenbank der Webanwendung die Versandbestätigungen gespeichert werden.

### **Daten automatisch zu Amazon senden**

Die Versandbestätigungen sollen von Montag bis Freitag um jeweils 19:30 Uhr automatisch an Amazon übermittelt werden.

Versandbestätigungen mit unvollständigen Daten sollen nicht zu Amazon gesendet werden.

### **Log-Datei per E-Mail senden**

Zur Identifizierung von möglichen Fehlern, muss die Webanwendung nach der Übermittlung der Daten zu Amazon eine E-Mail mit einer Log-Datei im Anhang an log@alfa-direkt.de senden.

### **Benutzeroberfläche**

Um einen manuellen Eingriff zu ermöglichen, muss eine einfache Benutzeroberfläche mit folgenden Funktionen bereitgestellt werden:

- Login und Logout
- Versandbestätigungen erstellen, bearbeiten und löschen
- Liste mit Versandbestätigungen anzeigen
  - Versandbestätigungen mit unvollständigen Daten
  - Versandbestätigungen die bald versendet werden
  - Die letzten 50 gesendeten Versandbestätigungen
- Versandbestätigungen manuell an Amazon senden

## A.3 Auszug aus dem Pflichtenheft

Umsetzung der Anforderungen aus dem Lastenheft:

### Erweiterbarkeit

Um eine Erweiterbarkeit der Webanwendung zu gewährleisten, soll das Projekt auf dem PHP Webframework „Laravel“ basieren. Laravel ist ein weit verbreitetes und modernes Framework, welches das **MVC-Muster** (Model-View-Controller) verwendet.

Das MVC-Muster ist ein Architekturmuster zur Trennung von Software in drei Komponenten: Model, View und Controller. Durch eine Trennung der drei Komponenten werden Änderungen und Erweiterungen erleichtert und die Wiederverwendbarkeit der einzelnen Komponenten ermöglicht.

### Bestellinformationen zur Webanwendung senden und speichern

Das Speichern der Versandbestätigungen wird über die URL `http://beispiel.de/feed/order-fulfillment/save?amazon_order_id=12345&carrier_name=GLS&shipper_tracking_number=12345` erfolgen. Im Query-String müssen also folgende Daten enthalten sein:

- `amazon_order_id` = Bestellnummer der Amazon-Bestellung
- `carrier_name` = Name des Versanddienstleisters
- `shipper_tracking_number` = Sendungsverfolgsnummer

Bei jedem GET-Request soll aus dem Query-String eine neue Bestellung sowie eine dazugehörige Versandbestätigung in der Datenbank gespeichert werden.

### Daten automatisch zu Amazon senden

Die Schnittstelle zum Amazon Merchant Web Service soll mit dem Laravel Package `amazon-mws-laravel` realisiert werden. Dieses Package kümmert sich im Hintergrund um die Authentifizierung (Token Management) zur Amazon Merchant Web Service API und stellt nützliche Methoden zur Verfügung, um mit der API zu kommunizieren. Hierzu werden die Amazon MWS API-Zugangsdaten des Händlerkontos benötigt.



Der automatische Versand der Versandbestätigungen soll durch einen Cronjob („ein Cronjob ist eine Jobsteuerung von Betriebssystemen wie Linux, der wiederkehrende Aufgaben zu einer bestimmten Zeit ausführen kann.“<sup>22</sup>) bzw. dem Laravel Scheduler<sup>23</sup> ausgelöst werden.

Wenn der Name des Versanddienstleisters oder die Sendungsverfolgungsnummer fehlt, soll der Status der Versandbestätigung dementsprechend geändert werden.

### **Log-Datei per E-Mail senden**

Die E-Mail mit der Log-Datei im Anhang soll mit Hilfe von Laravel versendet werden. Hierzu werden die SMTP-Daten des E-Mail-Kontos benötigt. Die E-Mail soll direkt nach der Datenübertragung zu Amazon gesendet werden.

### **Benutzeroberfläche**

Für die Benutzeroberfläche soll das HTML, CSS und JavaScript Framework Bootstrap (Version 3) eingesetzt werden. Auf der Startseite sollen nach erfolgreichem Login die Versandbestätigungen in Tabellenform aufgelistet werden. Links zum Bearbeiten von Versandbestätigungen sollen in den Tabellen angezeigt werden. Im Bearbeitungsformular von einer Versandbestätigung soll auch ein Button zum Löschen platziert werden. Ein Button zum Erstellen einer Versandbestätigung soll ganz oben über den Tabellen platziert werden. Ein Button zum manuellen Senden der Versandbestätigung soll unter der Tabelle der vollständigen Versandbestätigungen platziert werden.

---

<sup>22</sup> Vgl. <https://www.checkdomain.de/support/faq/webhosting-und-homepage/allgemeine-fragen-hosting-hp-ftp/was-ist-ein-cronjob.php>

<sup>23</sup> <https://laravel.com/docs/5.4/scheduling#introduction>

## A.4 Flussdiagramm zum Ablauf

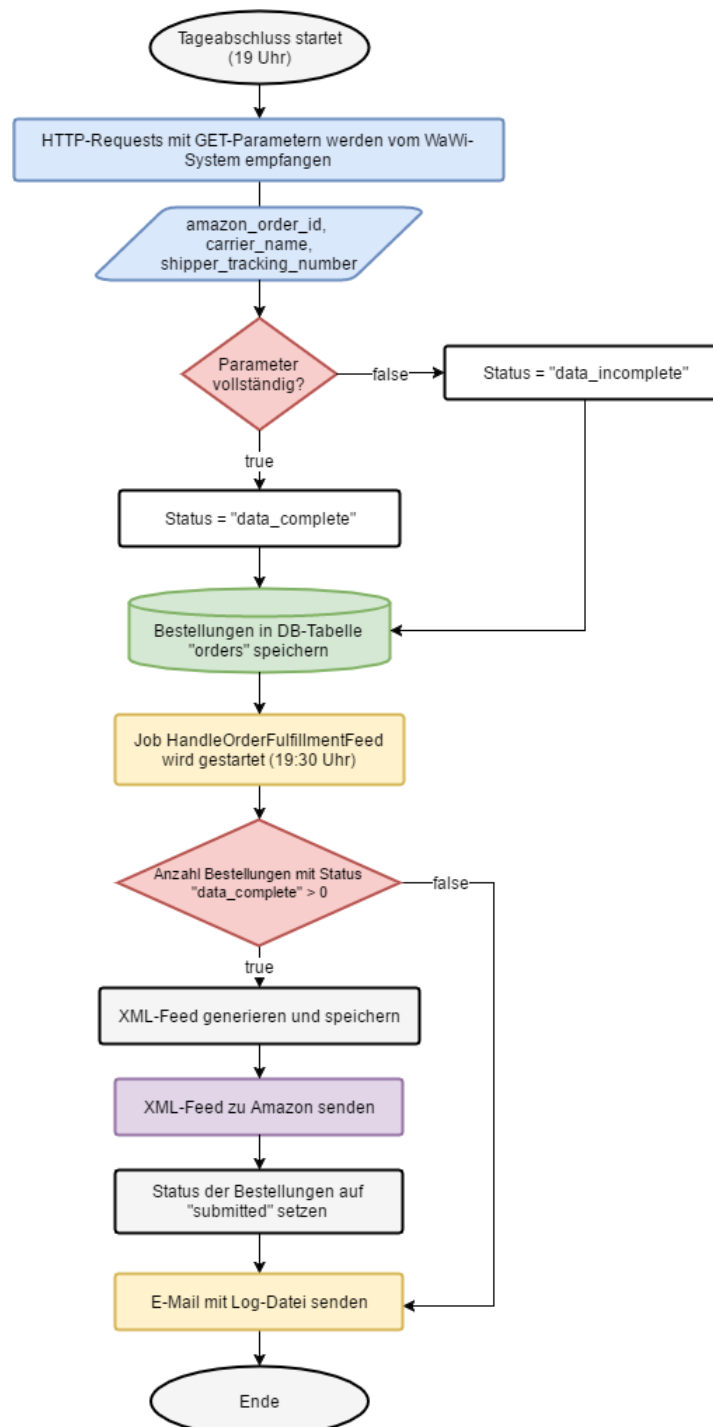


Abbildung 7: Flussdiagramm zum Ablauf

## A.5 Datenbankmigrationen

### A.5.1 Tabelle für Benutzer

```
1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateUsersTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->increments('id');
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->string('password');
21             $table->rememberToken();
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('users');
34     }
35 }
```

## A.5.2 Tabelle für Bestellungen

```
1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateOrdersTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('orders', function (Blueprint $table) {
17             $table->increments('id');
18             $table->string('amazon_order_id');
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      *
26      * @return void
27      */
28     public function down()
29     {
30         Schema::dropIfExists('orders');
31     }
32 }
```

## A.5.3 Tabelle für Versandbestätigungen

```
1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateDeliveryConfirmationsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('delivery_confirmations', function (Blueprint $table) {
17             $table->increments('id');
18             $table->integer('order_id')->unsigned();
19             $table->string('carrier_name')->nullable();
20             $table->string('shipper_tracking_number')->nullable();
21             $table->string('status');
```

```
22         $table->timestamps();
23
24         $table->foreign('order_id')
25             ->references('id')
26             ->on('orders')
27             ->onDelete('cascade');
28     });
29 }
30
31 /**
32  * Reverse the migrations.
33  *
34  * @return void
35  */
36 public function down()
37 {
38     Schema::dropIfExists('delivery_confirmations');
39 }
40 }
```

## A.6 Controller: Bestellinformationen speichern

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\DeliveryConfirmation;
6  use App\Order;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Log;
9
10
11 class OrderFulfillmentFeedController extends Controller
12 {
13     /**
14      * This method stores all incoming orders to the database.
15      *
16      * @param Request $request
17      * @return Order
18      */
19     public function save(Request $request)
20     {
21         // Get data from request.
22         $requestData = $request->all();
23         $amazonOrderID = $requestData['amazon_order_id'] ?
24 $requestData['amazon_order_id'] : null;
25         $carrierName = $requestData['carrier_name'] ? $requestData['carrier_name'] :
26 null;
27         $shipperTrackingNumber = $requestData['shipper_tracking_number'] ? $this-
28 >getFirstTrackingNumber($requestData['shipper_tracking_number']) : null;
29
30         // Create a new order.
31         $order = new Order();
32         $order->amazon_order_id = $amazonOrderID;
33         $order->save();
34
35         // Create a new delivery confirmation.
36         $deliveryConfirmation = new DeliveryConfirmation();
```

```

37     $deliveryConfirmation->order_id = $order->id;
38     $deliveryConfirmation->carrier_name = $carrierName;
39     $deliveryConfirmation->shipper_tracking_number = $shipperTrackingNumber;
40
41     /*
42      * Set the $deliveryConfirmation->status to 'data_incomplete' when carrier_name
43 or
44      * shipper_tracking_number are null - otherwise set $deliveryConfirmation-
45 >status to 'data_complete'.
46      * All delivery confirmations with the status 'data_incomplete' won't be sent
47 to Amazon!
48      */
49     if (
50         null === $deliveryConfirmation->carrier_name ||
51         null === $deliveryConfirmation->shipper_tracking_number
52     ) {
53         $deliveryConfirmation->status = 'data_incomplete';
54         Log::warning('Order #' . $order->amazon_order_id . ' has incomplete data.
55 This order will not be sent.');
```

```

56     }
57     else
58     {
59         $deliveryConfirmation->status = 'data_complete';
60     }
61
62     // Save and assign the delivery confirmation to the order.
63     $order->deliveryConfirmation()->save($deliveryConfirmation);
64
65     return $order;
66 }
67
68 /**
69  * This returns the shipper_tracking_number. There may be delivery confirmations
70  * with multiple tracking numbers. For Amazon, we only pick the first number.
71  *
72  * @param $shipperTrackingNumber
73  * @return String
74  */
75 function getFirstTrackingNumber($shipperTrackingNumber)
76 {
77     $shipperTrackingNumber = explode(',', $shipperTrackingNumber);
78
79     // Remove white spaces.
80     $shipperTrackingNumber = str_replace(' ', '', $shipperTrackingNumber);
81
82     return $shipperTrackingNumber[0];
83 }
84 }
```

## A.7 Job: Daten zu Amazon senden

```
1 <?php
2
3 namespace App\Jobs;
4
5 [...]
6
7 class HandleOrderFulfillmentFeed implements ShouldQueue
8 {
9     use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
10
11     /**
12      * Create a new job instance.
13      *
14      * @return void
15      */
16     public function __construct()
17     {
18         //
19     }
20
21     /**
22      * This job sends a _POST_ORDER_FULFILLMENT_DATA_ feed to Amazon.
23      *
24      * @return void
25      */
26     public function handle()
27     {
28         // Get all delivery confirmations with status 'data_complete'.
29         $deliveryConfirmations = DeliveryConfirmation::where('status',
30 'data_complete')->get();
31
32         if (count($deliveryConfirmations) > 0)
33         {
34             $merchantId = Config::get('amazon-mws.store.hwd_germany.merchantId');
35             $fulfillmentDate = Carbon::now()->toAtomString();
36
37             // Prepare the XML template.
38             $template = View::make('feeds.order-fulfillment')
39                 ->with('deliveryConfirmations', $deliveryConfirmations)
40                 ->with('merchantId', $merchantId)
41                 ->with('fulfillmentDate', $fulfillmentDate);
42
43             // Make a response using the XML template.
44             $response = Response::make($template, '200')->header('Content-Type',
45 'text/xml');
46
47             // Save the response content.
48             $responseContent = $response->getContent();
49
50             // Set the file name. For e.g. order-fulfillment_2017-04-12.xml
51             $fileName = 'order-fulfillment_' . Carbon::now()->toDateString() . '.xml';
52
53             // Create the XML feed.
54             Storage::disk('xml')->put($fileName, $responseContent);
55
56             // Get the XML feed.
57             $feed = Storage::disk('xml')->get($fileName);
58 }
```

```
59         // Create new AmazonFeed.
60         $amazon = new AmazonFeed('hwd_germany');
61         $amazon->setFeedType('_POST_ORDER_FULFILLMENT_DATA_');
62         $amazon->setFeedContent($feed);
63
64         foreach ($deliveryConfirmations as $confirmation)
65         {
66             Log::info('Processing Order #' . $confirmation->order-
67 >amazon_order_id);
68         }
69
70         // This call actually sends the feed to Amazon.
71         $amazon->submitFeed();
72
73         Log::info(count($deliveryConfirmations) . ' delivery confirmations
74 processed.');
```

```
75
76         // Send status mail.
77         Mail::to(env('MAIL_LOG_ADDRESS', 'log@alfa-direkt.de'))->send(new
78 OrdersSubmitted());
79
80         // Set $confirmation->status to 'submitted'.
81         foreach ($deliveryConfirmations as $confirmation)
82         {
83             $confirmation->status = 'submitted';
84             $confirmation->save();
85         }
86     }
87     else {
88         Log::info('No orders to process.');
```

```
89
90         // Send status mail.
91         Mail::to(env('MAIL_LOG_ADDRESS', 'log@alfa-direkt.de'))->send(new
92 OrdersSubmitted());
93     }
94 }
95 }
```



## A.8 Order Fulfillment XML Template

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 xsi:noNamespaceSchemaLocation="amznenvelope.xsd">
4   <Header>
5     <DocumentVersion>1.01</DocumentVersion>
6     <MerchantIdentifier>{{ $merchantId }}</MerchantIdentifier>
7   </Header>
8   <MessageType>OrderFulfillment</MessageType>
9   @foreach ($deliveryConfirmations as $key => $confirmation)
10  <Message>
11    <MessageID>{{ $key+1 }}</MessageID>
12    <OrderFulfillment>
13      <AmazonOrderID>{{ $confirmation->order->amazon_order_id }}</AmazonOrderID>
14      <FulfillmentDate>{{ $fulfillmentDate }}</FulfillmentDate>
15      <FulfillmentData>
16        <CarrierCode>{{ $confirmation->carrier_name }}</CarrierCode>
17        <ShipperTrackingNumber>{{ $confirmation->shipper_tracking_number
18      }}</ShipperTrackingNumber>
19      </FulfillmentData>
20    </OrderFulfillment>
21  </Message>
22  @endforeach
23 </AmazonEnvelope>
```

## A.9 Controller: Benutzeroberfläche

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\DeliveryConfirmation;
6 use App\Order;
7 use Illuminate\Http\Request;
8
9 class HomeController extends Controller
10 {
11     /**
12      * Show the application dashboard.
13      *
14      * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
15      */
16     public function index()
17     {
18         $incompleteDeliveryConfirmations = DeliveryConfirmation::where('status',
19 'data_incomplete')
20         ->with('order')
21         ->get();
22
23         $completeDeliveryConfirmations = DeliveryConfirmation::where('status',
24 'data_complete')
25         ->with('order')
26         ->get();
27
28         $submittedDeliveryConfirmations = DeliveryConfirmation::where('status',
```

```
29     'submitted')
30         ->with('order')
31         ->orderBy('updated_at', 'desc')
32         ->take(50)
33         ->get();
34
35     return view('home')
36         ->with('incompleteDeliveryConfirmations', $incompleteDeliveryConfirmations)
37         ->with('completeDeliveryConfirmations', $completeDeliveryConfirmations)
38         ->with('submittedDeliveryConfirmations', $submittedDeliveryConfirmations);
39 }
40 }
```

## A.10 Screenshots der Benutzeroberfläche

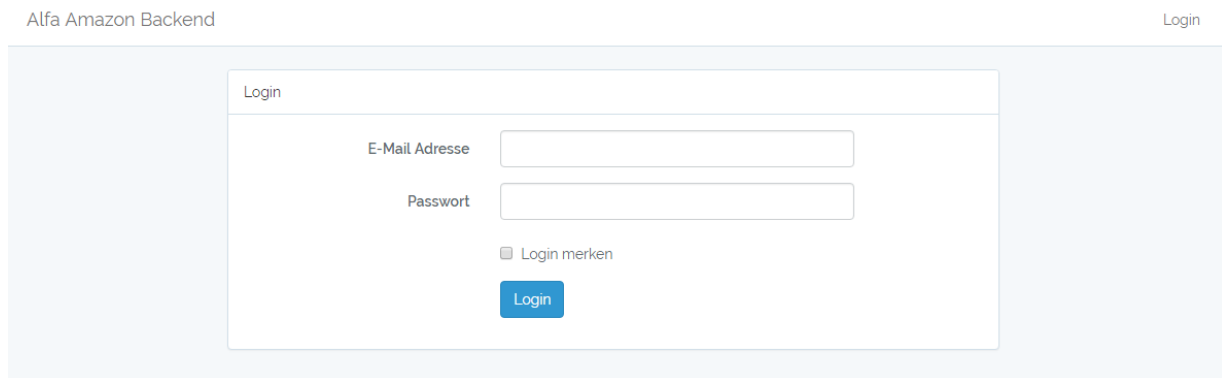


Abbildung 8: Login-Formular

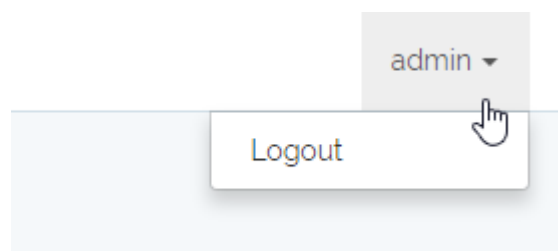


Abbildung 9: Logout-Link

Alfa Amazon Backendadmin ▾

### Versandbestätigungen

Neue Versandbestätigung

Unvollständige Versandbestätigungen (werden nicht gesendet)

ID	Amazon-Bestellnummer	Versand durch	Verfolgungsnummer	Erstellt am	Aktion
43	TEST3	GLS	fehlt	2017-04-18 18:22:55	<a href="#">Bearbeiten</a>
44	TEST4	fehlt	TEST4	2017-04-18 18:23:02	<a href="#">Bearbeiten</a>

Wartende Versandbestätigungen (werden um 19:30 Uhr gesendet)

ID	Amazon-Bestellnummer	Versand durch	Verfolgungsnummer	Erstellt am	Aktion
41	TEST1	GLS	TEST1	2017-04-18 18:22:42	<a href="#">Bearbeiten</a>
42	TEST2	GLS	TEST2	2017-04-18 18:22:51	<a href="#">Bearbeiten</a>

Versandbestätigungen manuell senden

Gesendete Versandbestätigungen (die letzten 50)

ID	Amazon-Bestellnummer	Versand durch	Verfolgungsnummer	Gesendet am	Aktion
15	304-9351583-0901164	GLS	833194009288	2017-04-13 19:01:24	<a href="#">Bearbeiten</a>
16	306-2696289-4866767	GLS	833194009431	2017-04-13 19:01:24	<a href="#">Bearbeiten</a>
17	305-3768450-7389161	GLS	833194009479	2017-04-13 19:01:24	<a href="#">Bearbeiten</a>
18	306-8088809-9312354	GLS	833194009356	2017-04-13 19:01:24	<a href="#">Bearbeiten</a>
19	306-4909202-0090718	GLS	833194009424	2017-04-13 19:01:24	<a href="#">Bearbeiten</a>

Abbildung 10: Versandbestätigungen in der Übersicht

Neue Versandbestätigung

Amazon-Bestellnummer

Versanddienstleister

GLS

Sendungsverfolgungsnummer

Speichern

Abbildung 11: Neue Versandbestätigung erstellen

Versandbestätigung ändern

Amazon-Bestellnummer

TEST-304-9351583-0901164

Versanddienstleister

GLS

Sendungsverfolgungsnummer

TEST-833194009288

Status

data\_complete

Speichern

Löschen

Abbildung 12: Versandbestätigung bearbeiten oder löschen

Diese Versandbestätigung wurde bereits zu Amazon gesendet!  
Wenn Änderungen gespeichert werden, dann wird diese Versandbestätigung erneut übermittelt.

Versandbestätigung ändern

Amazon-Bestellnummer

302-0641782-5639554

Versanddienstleister

GLS

Sendungsverfolgungsnummer

833194010918

Status

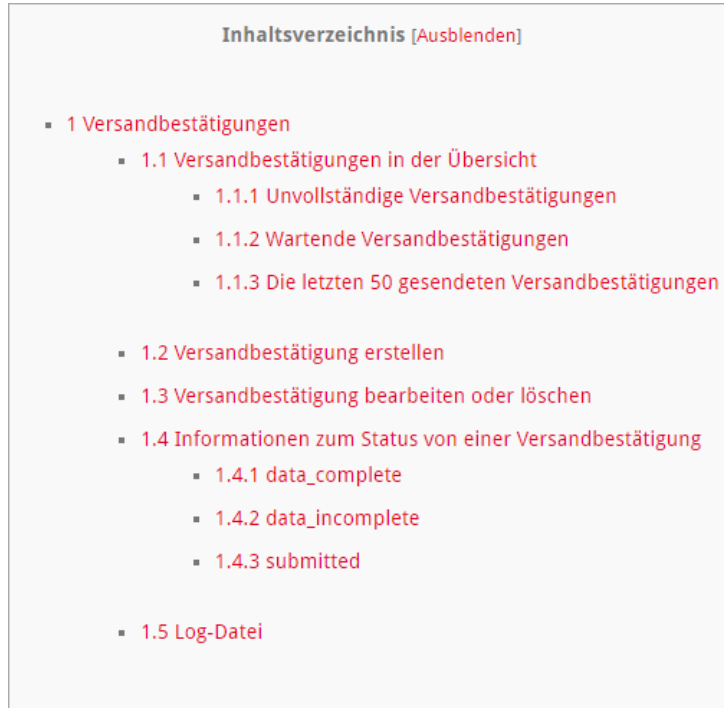
submitted

Speichern

Löschen

Abbildung 13: Eine bereits gesendete Versandbestätigung mit Hinweis

## A.11 Inhaltsverzeichnis der Benutzerdokumentation



The screenshot shows a web interface for the user documentation table of contents. At the top, the title 'Inhaltsverzeichnis' is followed by a link '[Ausblenden]'. Below this is a hierarchical list of topics, each preceded by a square bullet point. The topics are organized into five main sections: 1. Versandbestätigungen, 1.1 Versandbestätigungen in der Übersicht, 1.2 Versandbestätigung erstellen, 1.3 Versandbestätigung bearbeiten oder löschen, and 1.4 Informationen zum Status von einer Versandbestätigung. Section 1.1 has three sub-items: 1.1.1 Unvollständige Versandbestätigungen, 1.1.2 Wartende Versandbestätigungen, and 1.1.3 Die letzten 50 gesendeten Versandbestätigungen. Section 1.4 has three sub-items: 1.4.1 data\_complete, 1.4.2 data\_incomplete, and 1.4.3 submitted. Section 1.5 Log-Datei is listed at the bottom.

- **Inhaltsverzeichnis** [\[Ausblenden\]](#)
- **1 Versandbestätigungen**
  - **1.1 Versandbestätigungen in der Übersicht**
    - 1.1.1 Unvollständige Versandbestätigungen
    - 1.1.2 Wartende Versandbestätigungen
    - 1.1.3 Die letzten 50 gesendeten Versandbestätigungen
  - **1.2 Versandbestätigung erstellen**
  - **1.3 Versandbestätigung bearbeiten oder löschen**
  - **1.4 Informationen zum Status von einer Versandbestätigung**
    - 1.4.1 data\_complete
    - 1.4.2 data\_incomplete
    - 1.4.3 submitted
  - **1.5 Log-Datei**

Abbildung 14: Inhaltsverzeichnis der Benutzerdokumentation im Intranet

## A.12 Auszug der Entwicklerdokumentation

 [Edit README](#)

### Alfa Amazon Backend - Entwicklerdokumentation

Das Alfa Amazon Backend beinhaltet nützliche Tools, welche die Arbeit für Amazon-Händler erleichtern.

#### Laravel 5.4

Dieses Projekt basiert auf dem PHP Framework [Laravel 5.4](#). [Hier geht's zur Laravel Dokumentation](#)

#### Verwendete Packages

- [amazon-mws-laravel](#): "A library to connect to Amazon's MWS web services in an object oriented manner"
- [Forms & HTML](#): "HTML and Form Builders for the Laravel Framework"

#### Entwicklungsumgebung

Als Entwicklungsumgebung wird [Laravel Homestead](#) empfohlen.

#### Serveranforderungen

- Nginx (HTTP Server)
- MySQL (Datenbank)
- PHP 7
- Git (Versionierungssystem)
- Composer (Dependency-Manager für PHP)

#### Datenbank

Eine Datenbank mit dem Namen `alfa_amazon_backend` wird benötigt.

#### Nginx Konfiguration

Folgende Nginx-Konfiguration wird empfohlen:

```
server {
    listen 80;
    server_name amazonbackend.alfa-direkt.de;


    root "/var/www/amazonbackend/public";
    index index.php index.html index.htm;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }
}
```

Abbildung 15: Auszug aus der Entwicklerdokumentation

## A.13 Kopie des genehmigten Projektantrages

 <b>IHK</b> Industrie- und Handelskammer Ostwürttemberg							
<b>Antrag für die betriebliche Projektarbeit</b>							
Berufsbezeichnung / Fachrichtung / Einsatzgebiet / Fachbereich: <b>Informatikkaufmann</b>							
Antragsteller(in):  <b>Tobias Dalhof</b> <b>Rispenweg 22</b>  <b>73479 Ellwangen-Neunheim</b>	Ausbildungsbetrieb / Praktikumsbetrieb  <b>Alfa GmbH</b> <b>Dr.-Rudolf-Schieber-Str. 11-15</b>  <b>73463 Westhausen</b>						
Abschlussprüfung: <b>Sommer 2017</b>							
Projektbezeichnung / Projektziel (Auftrag / Teilauftrag): <b>Versandbestätigungen mit Sendungsverfolgungsnummern automatisch an das Amazon Seller Central senden</b>							
Kurze Projektbeschreibung (Anzahl der Endgeräte, verwendete Software, Plattform(en)): <b>Situation:</b> Der Amazon Marktplatz ist in unserem Unternehmen ein stark wachsender Vertriebskanal. Derzeit ist es so, dass ein Mitarbeiter die Versandbestätigungen für die Kunden manuell über das Amazon Seller Central versendet. Bei diesem Vorgang notiert sich ein Mitarbeiter von jeder Bestellung die Sendungsverfolgungsnummer aus dem Warenwirtschaftssystem und trägt diese dann von Hand in die Versandbestätigung bei Amazon ein. Da dies fehleranfällig und zeitaufwendig ist, soll dieser Arbeitsschritt automatisiert werden.  <b>Projekt:</b> Die Alfa GmbH benötigt eine Software, welche die Versandbestätigungen von Amazon-Bestellungen mit Hilfe vom Amazon Marketplace Web Service (Amazon MWS) automatisch versendet. Besonders wichtig ist, dass die Sendungsverfolgungsnummern aus dem System der Alfa GmbH in die Versandbestätigungen von Amazon eingetragen werden. Es soll ermittelt werden, wie viel Zeit und Kosten durch die Automatisierung eingespart werden. Die Software soll in PHP programmiert werden. Nach erfolgreichen Tests soll die Software auf einem Linux-Server installiert und in Betrieb genommen werden.							
Projektumfeld: <ul style="list-style-type: none"> <li>Alfa GmbH in Westhausen</li> </ul>							
*)Durchführungszeitraum:  <b>06.03.17 bis 24.03.17</b>	Projektverantwortlicher im Ausbildungsbetrieb / Praktikumbetrieb: <table border="1"> <tr> <th>Vorname</th> <th>Name</th> <th>Telefon</th> </tr> <tr> <td><b>Michael</b></td> <td><b>Beyer</b></td> <td><b>07363 954460</b></td> </tr> </table>	Vorname	Name	Telefon	<b>Michael</b>	<b>Beyer</b>	<b>07363 954460</b>
Vorname	Name	Telefon					
<b>Michael</b>	<b>Beyer</b>	<b>07363 954460</b>					



Industrie- und Handelskammer  
Ostwürttemberg

### Antrag für die betriebliche Projektarbeit

Berufsbezeichnung / Fachrichtung / Einsatzgebiet / Fachbereich  
**Informatikkaufmann**

Projektphasen mit Zeitplanung in Stunden:

1. Aufnahme der Anforderungen und Ziele	1 Std.
2. IST-Analyse	1 Std.
3. SOLL-Analyse	1 Std.
4. Kosten-Nutzen-Analyse	4 Std.
5. Pflichtenheft erstellen	4 Std.
<b>6. Software planen, erstellen und installieren</b>	
6.1. Einarbeitung in Amazon MWS	3 Std.
6.2. Programmierung der Software	10 Std.
6.3. Funktionsprüfung, Qualitätskontrolle	4 Std.
6.4. Installation und Inbetriebnahme	2 Std.
7. Einweisung und Abnahme	1 Std.
8. Dokumentation der Projektarbeit	4 Std.
<b>Summe: 35 Std.</b>	

Dokumentation zur Projektarbeit (Nicht selbstständig erstellte Dokumente sind zu unterstreichen!):

- Dokumentation
- Präsentation
- Lastenheft
- Pflichtenheft

Geplante Präsentationsmittel (Zutreffendes bitte ankreuzen):

Flipchart ☐ Tageslichtprojektor ☐ Pinnwand ☐ \*) Beamer ☒  
\*) Andere Präsentationsmittel: Notebook ☒

\*) Sind vom Prüfungsteilnehmer funktionsfähig mitzubringen

Einverständniserklärung des Auszubildenden zur Durchführung des Projekts

E-Mail: **m.beyer@alfa-direkt.de**  
Ort, Datum: **Westhausen, 02.02.17**  
Stempel und Unterschrift:  
  
*[Handwritten Signature]*

Antragsteller

E-Mail: **dalhof.tobias@gmail.com**  
Ort, Datum: **Westhausen, 02.02.17**  
*[Handwritten Signature: T. Dalhof]*





Industrie- und Handelskammer  
Ostwürttemberg

### Antrag für die betriebliche Projektarbeit

Berufsbezeichnung / Fachrichtung / Einsatzgebiet / Fachbereich  
**Informatikkaufmann**

Antragsteller(in):

**Tobias Dalhof**  
**Rispenweg 22**

**73479 Ellwangen-Neunheim**

Ausbildungsbetrieb / Praktikumbetrieb:

**Alfa GmbH**  
**Dr.-Rudolf-Schieber-Str. 11-15**

**73463 Westhausen**

Abschlussprüfung: **Sommer 2017**

Projektbezeichnung / Projektziel (Auftrag / Teilauftrag):  
**Versandbestätigungen mit Sendungsverfolgungsnummern automatisch an das Amazon Seller Central senden**

**Stellungnahme des Prüfungsausschusses:** (Bitte leserlich ausfüllen!)

\*) Nach Genehmigung darf mit der Durchführung der Projektarbeit begonnen werden. Änderungen dürfen nur mit Zustimmung des Prüfungsausschusses vorgenommen werden. Die Begründung bitte schriftlich bei der IHK einreichen.

☒ **genehmigt**

(Erneute Einreichung nicht notwendig)

☐ **genehmigt, Auflagen sind zu berücksichtigen**

☐ **abgelehnt, neuen Projektantrag vorlegen**  
bis zum .....

(Erneute Einreichung ist nur einmal möglich)

☐ **abgelehnt, Projektantrag unter Berücksichtigung der Auflagen erneut einreichen bis zum**  
.....

Unterschriften des Prüfungsausschusses:

Vorsitzender

*[Signature]*

Mitglieder

*[Signature]*

Ort, Datum

*Heidenheim, 20.02.2017*

## A.14 Persönliche Erklärung

### zur Projektarbeit und Dokumentation im Rahmen der Abschlussprüfung in den IT-Berufen

Ich versichere durch meine Unterschrift, dass ich die Durchführung der betrieblichen Projektarbeit als auch die dazugehörige Dokumentation selbständig in der vorgegebenen Zeit erarbeitet habe. Alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, wurden von mir als solche kenntlich gemacht.

Ebenso bestätige ich, dass ich bei der Erstellung der Dokumentation meiner Projektarbeit weder teilweise noch vollständig Passagen aus Projektarbeiten übernommen habe, die bei der prüfenden oder einer anderen Kammer eingereicht wurden.

Ich bestätige, dass die Dokumentation keine Betriebsgeheimnisse bzw. schutzwürdige Betriebs- oder Kundendaten enthält, das Urheberrecht beachtet wurde und es keine datenschutzrechtlichen Bedenken gibt.

Ort, Datum:

Unterschrift des Prüfungsteilnehmers:

---

---

Ich habe die obige Erklärung zur Kenntnis genommen und bestätige, dass die betriebliche Projektarbeit einschließlich der Dokumentation in der vorgegebenen Zeit in unserem Betrieb durch den Prüfungsteilnehmer angefertigt wurde.

Ort, Datum:

Unterschrift des Ausbilders:

---

---