

# Technische Dokumentation

## GNU Taler Plugin

## Verwendete Technologien

### 1.1. Programmiersprachen

Hier finden Sie eine Auflistung der Programmiersprachen, die wir für das Projekt verwendet haben.

- PHP 7.2+
- JavaScript ES5+
- WordPress PHP API
- WooCommerce PHP API

### 1.2. Software

Unterhalb sehen Sie eine Auflistung aller Software, die wir im Rahmen dieses Projektes verwendet haben.

- Programmierumgebung: JetBrains PhpStorm 2019.1
- Webserver: Apache Webserver 2.4.39
- Datenbank: MySQL 5.0.12

## 2. Installationsanleitung

### 2.1. Einleitung

Mit dieser Anleitung möchten wir veranschaulichen, was der einfachste und schnellste Weg ist das GNU Taler Plugin, auf einem Webserver mit WordPress, zu installieren. Ausserdem möchten wir die Voraussetzungen klar definieren, die erfüllt sein müssen, um das Plugin erfolgreich installieren und in Betrieb nehmen zu können.

### 2.2. Technische Voraussetzungen

Es müssen die nachfolgenden technischen Voraussetzungen erfüllt sein, damit das Plugin installiert und in Betrieb genommen werden kann.

Software:

Name Software	Version
WordPress	Mindestens 5.2
WooCommerce	Mindestens 3.6.2
GNU Taler Wallet	Mindestens 0.5.2 für Google Chrome und Opera Mindestens 0.6.65 für Mozilla Firefox

Hardware:

- Der Webserver (Apache HTTP Server, Microsoft IIS, etc), auf dem das Plugin laufen soll, muss die Programmiersprache PHP mindestens in der Version 7.2 unterstützen

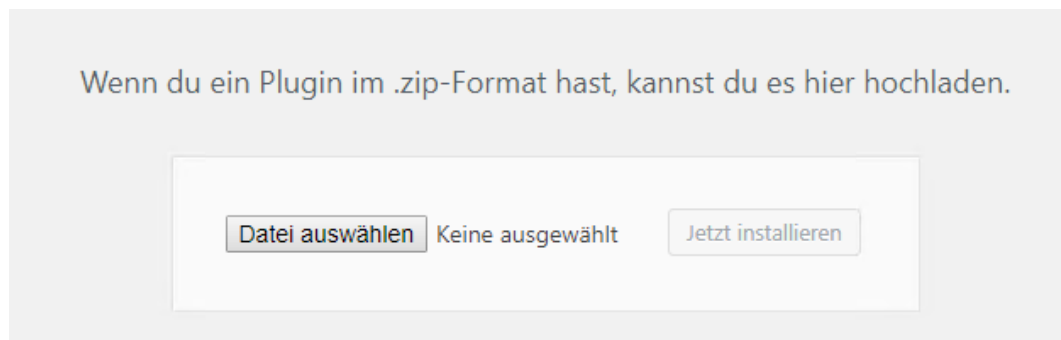
### 2.3. Installation Plugin

Da wir die technischen Voraussetzungen geklärt haben, kann man sich nun der schrittweisen Installation des Plugins widmen.

1. Als ersten Schritt muss man sich das GNU Taler Plugin vom WordPress Plugin Store (<https://wordpress.org/plugins/>) herunterladen.  
Nachdem heruntergeladen sollte man das Plugin als ZIP-Datei vorhanden haben, wie man im Bild unterhalb sehen kann:

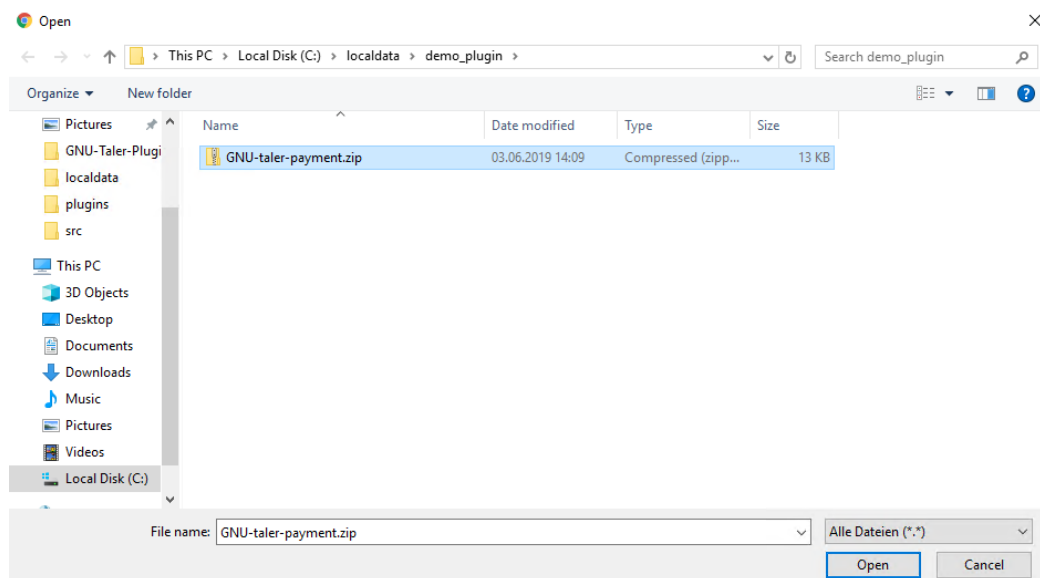


2. Danach muss das Plugin nun der WordPress-Seite hinzugefügt werden. Um das zu tun muss man im WordPress Admin-Dashboard nach Plugins->installierte Plugins->Installieren navigieren. Dort muss man dann auf den Knopf 'Plugin hochladen' klicken.
3. Auf dieser neuen Seite muss man auf 'Datei auswählen' klicken.



## GNU Taler Plugin

Von dort aus wird man dann auf ein Auswahlfenster weitergeleitet, wo man das Plugin auswählen kann.

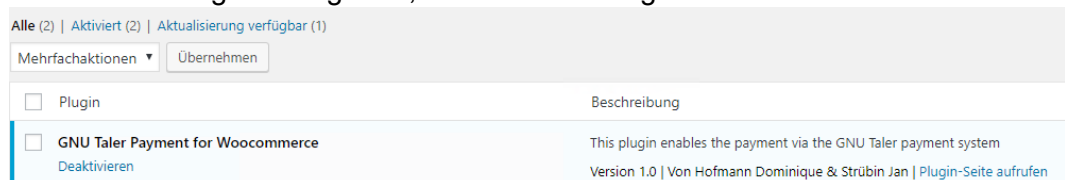


4. Anschliessend kann man auf 'Jetzt installieren' klicken und das Plugin wird von WordPress automatisch installiert. Der Installationsprozess sollte wie folgt aussehen.



Wenn der Installationsvorgang erfolgreich abgeschlossen worden ist, kann das Plugin über 'Plugin aktivieren' gerade noch aktiviert werden.

5. Als letzter Schritt sollte nun noch überprüft werden, ob das Plugin aktiviert wurde und auf der Plugin-Seite sichtbar ist. Für das muss man auf die Seite Plugins->Installierte Plugins navigieren, wo man dann folgendes sehen sollte.













Damit ist die Grundlegende Installation des Plugins vollendet und man könnte es bereits in Betrieb nehmen. Im nächsten Kapitel gibt es noch einen genaueren Blick auf die Einstellungen, die man beim Plugin vornehmen kann.

### 2.4. Einstellungen

In diesem Kapitel geht es darum einen detaillierten Einblick in die Einstellungsmöglichkeiten des Plugins zu werfen.

Als erstes, wo man die Einstellungen überhaupt finden kann. Um die Einstellungen verändern zu können, muss man zuerst im Admin-Dashboard zu der Seite WooCommerce->Einstellungen->Zahlungen navigieren. Dort befinden sich alle Bezahlungsmöglichkeiten, die man bei seinem Webshop finden kann.

**Zahlungsmethoden**  
Die installierten Zahlungsmethoden sind unten aufgeführt und die Reihenfolge, in der diese im Frontend angezeigt werden, kann per Drag-and-drop geändert werden.

Methode	Aktiviert	Beschreibung	
  <b>Direkte Banküberweisung</b> – Direct bank transfer	<input checked="" type="checkbox"/>	Nimm persönlich Zahlungen per BACS entgegen. Eher unter der Bezeichnung „Banküberweisung“ bekannt	<a href="#">Verwalten</a>
  <b>Scheckzahlungen</b> – Check payments	<input checked="" type="checkbox"/>	Nimm persönlich Zahlungen per Schecks entgegen. Dieser Offline-Zahlungsweg kann auch hilfreich sein, um Käufe zu testen.	<a href="#">Verwalten</a>
  <b>Per Nachnahme</b> – Cash on delivery	<input checked="" type="checkbox"/>	Lasse deine Kunden bei der Lieferung mit Bargeld (oder auf andere Weise) bezahlen.	<a href="#">Verwalten</a>
  <b>PayPal</b>	<input type="checkbox"/>	PayPal Standard leitet Kunden an PayPal weiter, damit sie ihre Zahlungsinformationen eingeben können.	<a href="#">Konfiguration</a>
  <b>GNU Taler Gateway</b> – GNU Taler	<input checked="" type="checkbox"/>	This plugin enables the payment via the GNU Taler payment system	<a href="#">Verwalten</a>

[Änderungen speichern](#)

Um zu den Einstellungen des GNU Taler Plugins zu gelangen, muss man nun bei der Bezahlungsmethode ‘GNU Taler Gateway’ auf den Knopf ‘Verwalten’ drücken.

Als nächstes sieht man eine neue Seite mit allen möglichen Einstellungen zum Plugin.

**GNU Taler Gateway** ⓘ  
This plugin enables the payment via the GNU Taler payment system

**Enable/Disable** ☒ Enable GNU Taler Gateway

**Title** ⓘ GNU Taler

**Description**  
Pay with the new Payment system GNU Taler.  
*This controls the description which the customer sees during checkout.*

**GNU Taler Backend URL**  
https://backend.demo.taler.net  
*Set the URL of the GNU Taler Backend.*

**GNU Taler Backend API Key**  
ApiKey sandbox  
*Set the API-key for the Authorization with the GNU Taler Backend.*

**GNU Taler Fulfillment URL**  
http://gnutaler.hofmd.ch  
*Set the URL where the customer should return after finishing the payment process.*

**Summarytext of the order**  
Order  
*Set the text the customer should see as a summary when he confirms the payment.*

**Enable/Disable** ☒ Enable sending your merchant information to the GNU Taler Backend  
*Do you want to send your merchant information to the GNU Taler Backend via the transaction*

**Name of the webshop**  
GNU Taler Webshop  
*Set the name of the webshop that the customer will see during the payment transaction.*

[Änderungen speichern](#)

## GNU Taler Plugin

Nachfolgend finden Sie eine Auflistung aller Einstellungen und was deren Bedeutung ist:

Einstellung	Bedeutung
Enable / Disable Plugin	Mit dieser Einstellung kann man die Bezahlmethode aktivieren oder deaktivieren.
Title	Der Titel der Bezahloption, die der Kunde bei der Kasse sehen kann.
Description	Die Beschreibung der Bezahloption bei der Auswahl der Bezahlungsmöglichkeiten.
GNU Taler Backend URL	Die URL zum GNU Taler Backend, mit der das Plugin kommuniziert. Wird bei jeder neuen Bestellung vorher verifiziert.
GNU Taler Backend API Key	API Key für die Authentifizierung beim GNU Taler Backend.
Summarytext of the order	Zusammenfassung, die der Kunde beim Bestätigen der Bestellung sieht.
Enable / Disable Merchant Information	Hier kann man festlegen, ob man seine Daten, mit jeder Bestellung an das Backend mitsenden will oder nicht.
Name of the Webshop	Der Name des Webshops, der bei der Bestätigung der Bestellung dargestellt wird.

Schlussendlich kann man die Änderungen mit dem Knopf 'Änderungen speichern' abspeichern.

### 3. Gebrauchsanleitung

#### 3.1. Einleitung

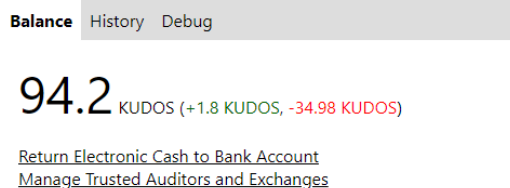
Bei dieser Gebrauchsanleitung geht es darum aufzuzeigen, wie der Kaufprozess, aus der Sicht des Kunden und der Rückerstattungsprozess, aus der Sicht des Systemadministrator aussieht.

#### 3.2. Kaufprozess

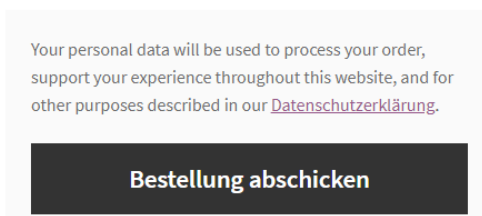
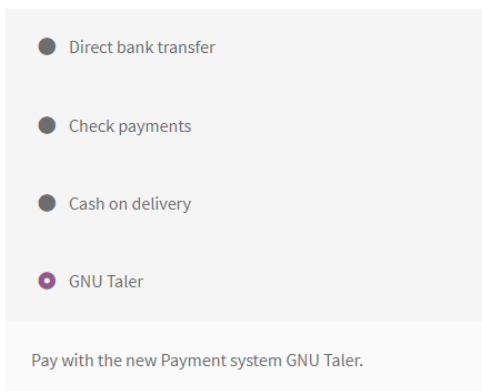
Nachfolgend sehen Sie wie der Kaufprozess, Schritt für Schritt, auf einem WordPress Webserver mit einem WooCommerce Webshop aussieht.

Für dieses Szenario gehen wir davon aus, dass der Kunde bereits ein oder mehrere Produkte zu seinem Warenkorb hinzugefügt und sich anschliessend zur Kasse begeben hat.

1. Als erstes sollte der Kunde überprüfen, ob er die GNU Taler Wallet als Browser Extension installiert hat und über genügend Geldmittel verfügt, um den Einkauf zu tätigen. Wenn alles korrekt installiert wurde, sollte es wie folgt aussehen.



2. Wenn der Kunde alle notwendigen Kontaktdaten für die Transaktion ausgefüllt hat, ist der nächste Schritt das GNU Taler Bezahlungssystem als Bezahlmethode auszuwählen.



Wenn man dies getan hat, kann man abschliessend die Bestellung abschicken.

3. Wenn nun keine Fehler während der Abwicklung der Transaktion auftreten, wird der Kunde schlussendlich auf seine GNU Taler Wallet weitergeleitet.

### GNU Taler Wallet

The merchant **GNU Taler Webshop** offers you to purchase:

**Order of the following items:**

The following items are included:

- Order of product: Beanie: 1 KUDOS

The total price is **5 KUDOS** (plus 0.2 KUDOS fees).

[Confirm payment](#)

[show more details](#)

Bei diesem Fenster kann der Kunde noch einmal überprüfen, ob er die korrekten Produkte bestellt hat und falls alles stimmt, kann er die Transaktion mit 'Confirm payment' bestätigen.

4. Anschliessend wird der Kunde zum Webshop zurückgeleitet und kann auf seiner GNU Taler Wallet, die soeben getätigte Transaktion anschauen.

Balance History Debug

89 KUDOS (+1.8 KUDOS, -34.98 KUDOS)

[Return Electronic Cash to Bank Account](#)  
[Manage Trusted Auditors and Exchanges](#)

Alle getätigten Transaktionen sind auf unter 'History' sichtbar.

Balance History Debug

Tue Jun 11 2019 16:31:38 GMT+0200 (Central European Summer Time)

Paid 5 KUDOS to merchant *GNU Taler Websh...* ([view product](#))

Tue Jun 11 2019 16:28:09 GMT+0200 (Central European Summer Time)

Merchant *GNU Taler Websh...* offered contract FAHV5...

Wed Jun 05 2019 17:43:31 GMT+0200 (Central European Summer Time)

Paid 1 KUDOS to merchant *GNU Taler Websh...* ([view product](#))

Wed Jun 05 2019 17:24:15 GMT+0200 (Central European

Somit ist der Kaufprozess beendet und der Kunde kann einen von neu an anfangen.

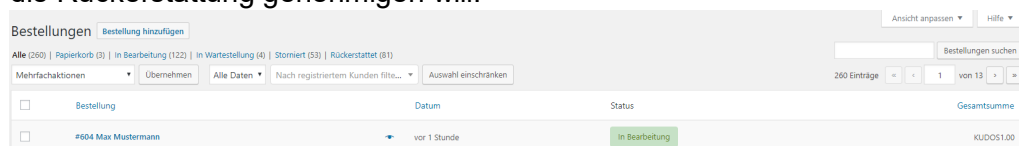


### 3.3. Rückerstattungsprozess

In diesem Kapitel möchten wir veranschaulichen, wie der Prozess für eine Rückerstattung aus der Sicht des Systemadministrators abläuft.

Für dieses Szenario nehmen wir an, dass bereits eine Bestellung getätigt wurde und der Kunde eine Rückerstattung auf ein bestimmtes Produkt anfordert.

1. Um eine Rückerstattung gewähren zu können, muss der Systemadministrator sich auf dem WordPress Dashboard befinden. Von dort aus muss man dann zu WooCommerce->Bestellungen, wo man alle im System befindlichen Bestellungen sehen kann.
2. Von dort aus muss der Systemadministrator auf die Bestellung klicken, wo man die Rückerstattung genehmigen will.



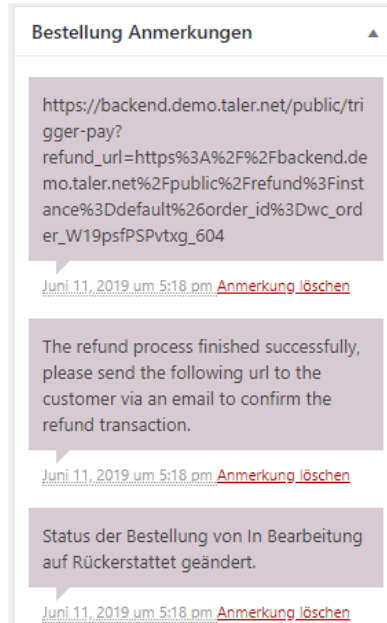
3. Die Bestellungen-Seite sollte dann wie folgt aussehen.

Wie man unten links, beim Pfeil, sehen kann hat jede Bestellung einen Rückerstattungs-Knopf. Um die Rückerstattung erstellen zu können muss man dementsprechend auf den Knopf drücken.

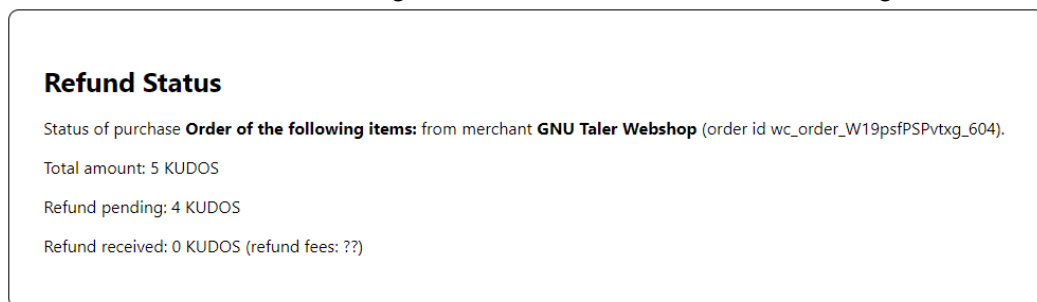
4. Daraufhin sollte man dann folgende Erweiterung sehen.

Man hat nun die Möglichkeit eine Rückerstattung, über einen beliebigen Betrag, entweder manuell oder über das GNU Taler Bezahlungssystem zu initiieren. Wenn man den Betrag und einen Grund für die Rückerstattung angegeben hat, kann man den Knopf 'Rückerstattung über den GNU Taler Gateway' drücken.

5. Nachdem der Rückerstattungs-Prozess vom Plugin beendet worden ist, erhält man einen Weiterleitungslink zur GNU Taler Wallet des Kunden.

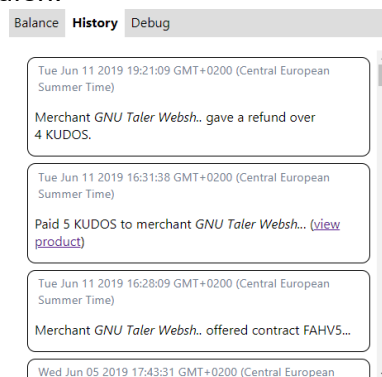


6. Diesen kann man dann als Systemadministrator dem Kunden senden, der über diesen Link zur Rückerstattungs-Seite des GNU Taler Wallet weitergeleitet wird.



Nach Aktivierung der Rückerstattung durch den Kunden, wird der Betrag auf die Wallet des Kunden überwiesen und der Prozess ist beendet.

Der Kunde kann die Überweisung, wie im vorherigen Kapitel gezeigt, in seiner GNU Taler Wallet nachprüfen.



## 4. Testfälle

### 4.1. Anmerkung


Da das Plugin auf der Basis der WordPress API und der WooCommerce API programmiert wurde, kam es zu unvorhergesehenen Schwierigkeiten beim Erstellen der Testfälle. Zum Schreiben der Testfälle haben wir ein Testing-Framework namens 'PHPUnit' verwendet. Jedoch konnte das Framework, aufgrund einer Problematik mit den Coding-Standards von WooCommerce, nicht auf die nötigen Informationen zugreifen und es uns somit ermöglichen die Testfälle für das Plugin zu schreiben. Deswegen mussten wir den Teil des Codes, der testbar und nicht Teil der WooCommerce oder WordPress API war, in eine separate Datei auslagern. Danach konnten wir auf dem Code basierend unsere Testfälle erstellen. Infolge dessen ist die Situation ein bisschen komplexer geworden. Wir konnten das Problem jedoch zu unserer Zufriedenheit lösen.

### 4.2. Übersicht

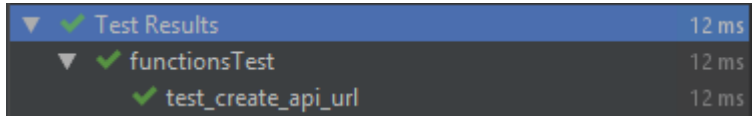
Unterhalb sehen Sie eine kurze Übersicht über alle durchgeführten Testfälle:

Testfall Nr.	Beschreibung
1	Dieser Testfall behandelt die Funktion <code>call_api</code> , welche das Senden von Requests und das zurückgeben von Responses übernimmt. Im Rahmen des Testfalls wird die Methode in mehreren Szenarien angewendet, welche aus dem Umfeld des Plugins stammen. Dazu gehören untereinander eine Bestellung mit Bestätigung und eine Rückerstattung die an das GNU Taler Backend gesendet und validiert wird.
2	In diesem Testfall wird die Funktion <code>create_api_url</code> überprüft, welche benutzt wird, um eine gegebene URL anzupassen. Mit dem Ziel, dass spätere Requests an die richtige Adresse gehen. Deswegen werden verschiedene URL übergeben und anschliessend überprüft, ob die zurückgegebenen URL valid sind oder nicht.
3	Bei diesem und den zwei folgenden Testfällen, wird die Funktion <code>curl_error_handling</code> behandelt. Welche die Funktion hat HTTP Status Codes abzufangen und diese zu validieren. In diesem spezifischen Testfall geht es darum die Status Codes mit dem Wert 200 abzufangen und der Funktion zu übergeben. Anschliessend wird überprüft, ob der zurückgegebene Wert mit dem Erwartungswert übereinstimmt oder nicht.
4	Wie im vorherigen Testfall beschrieben, geht es in diesem Fall auch wieder um das Abfangen und Validieren von HTTP Status Codes. Hier geht es spezifischer um die Status Codes mit dem Wert 400, welche einen Fehler beim Kunden vermuten lassen. Wie beim oberen Testfall geht es darum mehrere Requests abzusenden und die Antwort der Funktion zu übergeben. Daher wird überprüft ob die Funktion den korrekten Wert zurückliefert oder nicht.
5	Beim letzten Testfall geht es darum die Codes mit dem Wert 500 abzufangen. Dieser Wert lässt einen Fehler beim Server vermuten bei welchem die Requests landen sollten. Zur Überprüfung werden dann mehrere Requests abgesendet und die Antwort wieder der Funktion übergeben. Im Anschluss wird noch einmal zwischen dem erwarteten und dem zurückgegebenen Wert der Funktion verglichen.

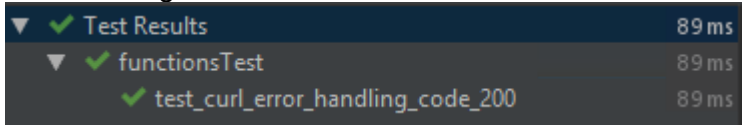
### 4.3. Testfall 1

Testfall Nr.	Datum
1	05.04.2019
<b>Beschreibung</b>	
In diesem Testfall geht es darum die Funktion <code>call_api()</code> zu testen, welche die Aufgabe hat einen Request via einer gewünschten Methode (POST, GET, PUT, etc) an eine URL zu senden und die Antwort entsprechend zu verarbeiten. Nachdem der Request versendet wurde, wird ein Wert zurückgegeben, der den Erfolg oder Misserfolg darstellt.	
<b>Vorarbeit</b>	
<p>Als Vorbereitung für diesen Testfall mussten Testdaten erstellt werden, damit verschiedene Szenarien abgedeckt werden konnten.</p> <p>Die Szenarien waren wie folgt:</p> <ol style="list-style-type: none"> <li>1. Eine Bestellung beim GNU Taler Backend zu erstellen und eine vom Backend validierte Bestellungs-ID zurückzuerhalten.</li> <li>2. Eine vom Backend validierte Bestellung per Request von der Seite des Verkäufers bestätigen, um den Kunden zur Zahlungsbestätigung weiterleiten zu können.</li> <li>3. Eine Verifizierung einer URL, die zum GNU Taler Backend führen sollte.</li> <li>4. Eine Anfrage an eine valide URL über eine nicht vorhandene Methode (Nicht POST, GET oder PUT).</li> </ol>	
<b>Erwartetes Ergebnis</b>	
<p>Szenarien:</p> <ol style="list-style-type: none"> <li>1. Bei diesem Szenario war das erwartete Ergebnis zwei verschiedene Werte. Zum einen ein Wert 'true' vom Typ Boolean und zum anderen ein Array mit den Einzelheiten der Bestellung, wozu auch die gesuchte Bestellungs-ID dazugehört.</li> <li>2. Beim zweiten wurde, wie oben, ein Wert 'true' vom Typ Boolean und ein Array mit Informationen zur Bestellung erwartet. Teil des Arrays ist die gesuchte Weiterleitungs-URL und auch, ob die Bestellung bereits bezahlt wurde.</li> <li>3. Bei diesem Szenario wurde wieder ein Wert 'true' vom Typ Boolean und eine Nachricht vom Typ String erwartet.</li> <li>4. Beim letzten Szenario wurde ein Wert 'false' vom Typ Boolean und eine Nachricht 'Bad Request' vom Typ String erwartet.</li> </ol>	
<b>Eingetroffenes Ergebnis</b>	
<p>Alle zurückgegebene Werte stimmten mit den erwarteten Werten überein. Unten ist der erfolgreiche Test als Screenshot zu sehen.</p>  <p>The screenshot shows a list of test results in a dark-themed interface. It includes a dropdown arrow, a green checkmark, and the text 'Test Results' with a duration of '821 ms'. Below this is another entry with a dropdown arrow, a green checkmark, and the text 'functionsTest' with a duration of '821 ms'. The bottom entry, which is highlighted in blue, shows a green checkmark, the text 'test_call_api', and a duration of '821 ms'.</p>	

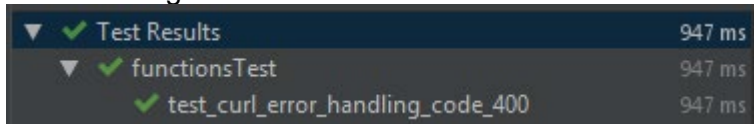
#### 4.4. Testfall 2

Testfall Nr.	Datum
2	05.04.2019
Beschreibung	
In diesem Testfall geht es darum die Funktion <code>create_api_url()</code> zu testen. Es hat die Funktion eine gegebene URL zu modifizieren, damit später ein Request an die richtige Adresse geht. Der Wert, der zurückgegeben wird, ist die URL mit einem modifizierten Zusatz.	
Vorarbeit	
Als Vorarbeit mussten Werte für die verschiedenen Szenarien vorbereitet werden: <ol style="list-style-type: none"> <li>1. URL, wenn eine Bestellung erstellt werden soll.</li> <li>2. URL, Wenn eine Bezahlung für eine Bestellung bestätigt werden soll.</li> <li>3. URL, Wenn eine Rückerstattung beantragt und bestätigt werden soll.</li> </ol>	
Erwartetes Ergebnis	
Szenarien: <ol style="list-style-type: none"> <li>1. Bei diesem Szenario wird erwartet, dass der URL der entsprechende Zusatz hinzugefügt wurde, um beim Backend eine Bestellung beantragen zu können: Bsp. 'https://backend.demo.taler.net/order'</li> <li>2. Bei diesem Szenario ist der Wert, der erwartet wird, die URL mit dem hinzugefügten Teil, der es einem ermöglicht die Bestellung zu bestätigen: Bsp. 'https://backend.demo.taler.net/order_id?wc_order1234'</li> <li>3. Beim letzten Szenario ist der zu erwartende Wert, eine URL mit dem angefügten Ende, um eine Rückerstattung beantragen zu können: Bsp. 'https://backend.demo.taler.net/refund'</li> </ol>	
Eingetroffenes Ergebnis	
Alle Werte der Szenarien, die zurückgegeben wurden, stimmten mit den erwarteten Werten überein. Der erfolgreiche Test ist unterhalb als Screenshot einsehbar:	
	

#### 4.5. Testfall 3

Testfall Nr.	Datum
3	05.04.2019
Beschreibung	
In diesem Testfall geht es darum einen Teil der Funktion <code>curl_error_handling()</code> zu testen. Die Aufgabe der Funktion ist es HTTP Status Codes, von fertig abgeschlossenen Requests, abzufangen und zu validieren. Dieser Testfall behandelt die Status Codes mit dem Wert 200, was einen erfolgreichen Abschluss bedeutet.	
Vorarbeit	
Als Vorarbeit musste nur ein Szenario erstellt werden, da es sich nur um einen Request handelt: 1. Überprüfung einer URL auf Validität	
Erwartetes Ergebnis	
Bei diesem Szenario ist das Ergebnis, das erwartet wird, zwei Werte. Der Erste ist ein Wert 'true' vom Typ Boolean und der Zweite der HTTP Status Code.	
Eingetroffenes Ergebnis	
Alle zurückgegebene Werte stimmten mit den erwarteten Werten überein, dass bedeutet, dass der Request erfolgreich abgearbeitet wurde. Der Screenshot des erfolgreichen Tests ist unterhalb sichtbar:	
	

#### 4.6. Testfall 4

Testfall Nr.	Datum
4	05.04.2019
Beschreibung	
In diesem Testfall geht es darum den anderen Teil der Funktion <code>curl_error_handling()</code> zu testen. Dieser Testfall behandelt die Status Codes mit dem Wert 400, was einen Fehler auf der Seite des Anwenders bedeutet.	
Vorarbeit	
Als Vorarbeit mussten Werte für die verschiedenen Szenarien vorbereitet werden: 1. Eine Bestellung per Request bestätigen, jedoch ist die JSON-Datei die mitgeschickt wird fehlerhaft. 2. Eine URL soll validiert werden, jedoch ist der Authentifizierungsschlüssel falsch 3. Eine URL soll validiert werden, jedoch erfolgt der Request auf eine URL, wo der Anwender keine Rechte hat. 4. Eine URL soll validiert werden, jedoch führt die URL auf eine Seite, die nicht mehr vorhanden ist.	
Erwartetes Ergebnis	
Bei allen Szenarien ist das zu erwartende Ergebnis zwei Werte. Zum einen ein Wert 'false' vom Typ Boolean und ein entsprechender HTTP Status Code.	
Eingetroffenes Ergebnis	
Alle zurückgegebene Werte stimmten mit den erwarteten Werten überein, dass bedeutet, dass die Fehler erfolgreich abgefangen wurden. Der Screenshot des erfolgreichen Tests ist unterhalb sichtbar:	
	

#### 4.7. Testfall 5

Testfall Nr.	Datum
5	05.04.2019
Beschreibung	
In diesem Testfall geht es darum den letzten Teil der Funktion <code>curl_error_handling()</code> zu testen. Dieser Testfall behandelt die Status Codes mit dem Wert 500, was einen Fehler auf der Seite des Servers bedeutet.	
Vorarbeit	
<p>Als Vorarbeit mussten Werte für die verschiedenen Szenarien vorbereitet werden:</p> <ol style="list-style-type: none"> <li>1. Eine URL soll validiert werden, jedoch erfolgt der Request auf eine URL, wo der Server einen Fehler hat.</li> <li>2. Eine URL soll validiert werden, jedoch gibt es einen Netzwerkfehler zwischen dem Anwender und dem Server</li> <li>3. Eine URL soll validiert werden, jedoch erfolgt der Request auf eine URL, wo auf dem Server Wartungsarbeiten stattfinden.</li> <li>4. Eine URL soll validiert werden, jedoch gibt es einen Timeout Fehler auf dem Weg zum Server</li> </ol>	
Erwartetes Ergebnis	
Bei allen Szenarien ist das zu erwartende Ergebnis zwei Werte. Zum einen ein Wert 'false' vom Typ Boolean und ein entsprechender HTTP Status Code.	
Eingetroffenes Ergebnis	
<p>Alle zurückgegebene Werte stimmten mit den erwarteten Werten überein, dass bedeutet, dass die Fehler erfolgreich abgefangen wurden.</p> <p>Der Screenshot des erfolgreichen Tests ist unterhalb sichtbar:</p> 