# Inhaltsverzeichnis

Version history

This document will be versioned with the server, hence this is document version 0.7 from 2023-01-19. It is the first release of this document so no version history is available yet.

Server access

Access to the web interface is possible via https://api.usegroup.de/. Terms of service are listed in the chapter Terms of service on page 23.

To register please access https://api.usegroup.de:9443/devportal/services/configs
and click "Create Account" link on the bottom left. Select a username, "Proceed to self register", enter the rest of the data and have your email verified.
Afterwards you can login on https://api.usegroup.de .

If you select the desired Mustangserver version and click on the blue "try out" button (not the link in the navigation) on the next page you should be able to click a "get test key" button.
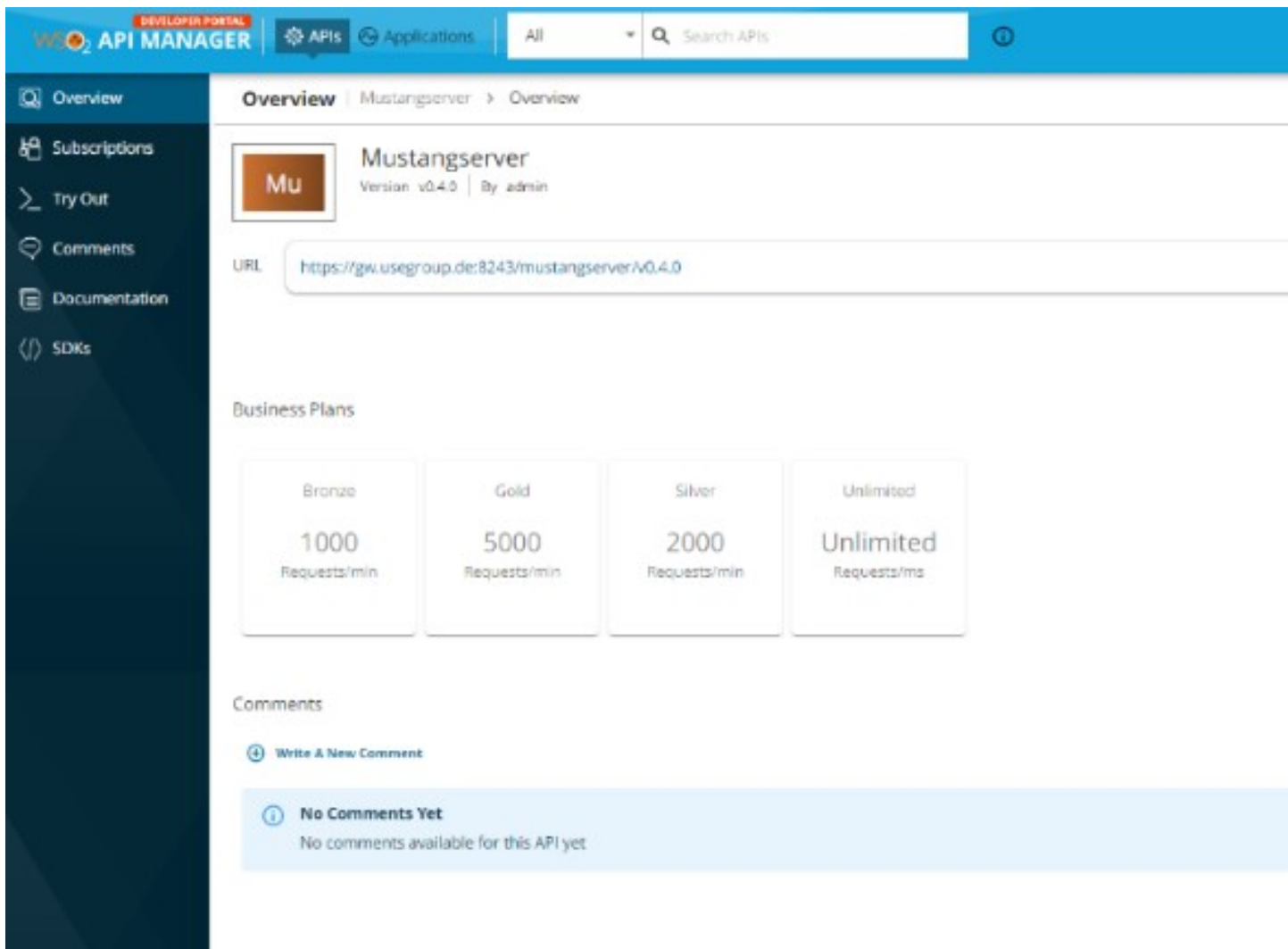
e.g. when you open the "ping" operation and click "try it out" and "execute" you should get a "pong" response.

Operations

The available operations are

- ping: Just a test, always just returns „pong". The only operation acessible via HTTP GET, the rest is POST.

- validate: Validate a Factur-X/ZUGFeRD or XRechnung CII or Order-X-CIO File using Mustang's validator. Requires a file and returns a XML report. The format to be validated against will be read from it's guideline ID.

- phive: Validate a CII or UBL file using https://github.com/phax/phive . Requires a VES ID and a file and returns JSON.

  Available VES IDs are

de.xrechnung:cii:1.2.0
de.xrechnung:cii:1.2.1
de.xrechnung:cii:1.2.2
de.xrechnung:cii:2.0.0
de.xrechnung:cii:2.0.1
de.xrechnung:cii:2.1.1
de.xrechnung:cii:2.2.0
de.xrechnung:ubl-creditnote:1.2.0
de.xrechnung:ubl-creditnote:1.2.1
de.xrechnung:ubl-creditnote:1.2.2
de.xrechnung:ubl-creditnote:2.0.0
de.xrechnung:ubl-creditnote:2.0.1
de.xrechnung:ubl-creditnote:2.1.1
de.xrechnung:ubl-creditnote:2.2.0
de.xrechnung:ubl-invoice:1.2.0
de.xrechnung:ubl-invoice:1.2.1
de.xrechnung:ubl-invoice:1.2.2
de.xrechnung:ubl-invoice:2.0.0
de.xrechnung:ubl-invoice:2.0.1
de.xrechnung:ubl-invoice:2.1.1
de.xrechnung:ubl-invoice:2.2.0
eu.cen.en16931:cii:1.0.0
eu.cen.en16931:cii:1.1.0
eu.cen.en16931:cii:1.2.0
eu.cen.en16931:cii:1.2.1
eu.cen.en16931:cii:1.2.3
eu.cen.en16931:cii:1.3.0
eu.cen.en16931:cii:1.3.1
eu.cen.en16931:cii:1.3.2
eu.cen.en16931:cii:1.3.3
eu.cen.en16931:cii:1.3.4
eu.cen.en16931:cii:1.3.5
eu.cen.en16931:cii:1.3.6
eu.cen.en16931:cii:1.3.6a
eu.cen.en16931:cii:1.3.7
eu.cen.en16931:cii:1.3.8
eu.cen.en16931:ubl-creditnote:1.0.0
eu.cen.en16931:ubl-creditnote:1.1.0
eu.cen.en16931:ubl-creditnote:1.2.0
eu.cen.en16931:ubl-creditnote:1.2.1
eu.cen.en16931:ubl-creditnote:1.2.3
eu.cen.en16931:ubl-creditnote:1.3.0
eu.cen.en16931:ubl-creditnote:1.3.1
eu.cen.en16931:ubl-creditnote:1.3.2
eu.cen.en16931:ubl-creditnote:1.3.3
eu.cen.en16931:ubl-creditnote:1.3.4
eu.cen.en16931:ubl-creditnote:1.3.5
eu.cen.en16931:ubl-creditnote:1.3.6
eu.cen.en16931:ubl-creditnote:1.3.6a
eu.cen.en16931:ubl-creditnote:1.3.7
eu.cen.en16931:ubl-creditnote:1.3.8
eu.cen.en16931:ubl:1.0.0
eu.cen.en16931:ubl:1.1.0

eu.cen.en16931:ubl:1.2.0
eu.cen.en16931:ubl:1.2.1
eu.cen.en16931:ubl:1.2.3
eu.cen.en16931:ubl:1.3.0
eu.cen.en16931:ubl:1.3.1
eu.cen.en16931:ubl:1.3.2
eu.cen.en16931:ubl:1.3.3
eu.cen.en16931:ubl:1.3.4
eu.cen.en16931:ubl:1.3.5
eu.cen.en16931:ubl:1.3.6
eu.cen.en16931:ubl:1.3.6a
eu.cen.en16931:ubl:1.3.7
eu.cen.en16931:ubl:1.3.8
eu.peppol.bis3.aunz.ubl:creditnote-self-billing:1.0.6
eu.peppol.bis3.aunz.ubl:creditnote-self-billing:1.0.7
eu.peppol.bis3.aunz.ubl:creditnote:1.0.6
eu.peppol.bis3.aunz.ubl:creditnote:1.0.7
eu.peppol.bis3.aunz.ubl:invoice-self-billing:1.0.6
eu.peppol.bis3.aunz.ubl:invoice-self-billing:1.0.7
eu.peppol.bis3.aunz.ubl:invoice:1.0.6
eu.peppol.bis3.aunz.ubl:invoice:1.0.7
eu.peppol.bis3.sg.ubl:creditnote:1.0.2
eu.peppol.bis3.sg.ubl:creditnote:1.0.3
eu.peppol.bis3.sg.ubl:invoice:1.0.2
eu.peppol.bis3.sg.ubl:invoice:1.0.3
eu.peppol.bis3:catalogue-response:3.13.0
eu.peppol.bis3:catalogue-response:3.14.0
eu.peppol.bis3:catalogue:3.13.0
eu.peppol.bis3:catalogue:3.14.0
eu.peppol.bis3:creditnote:3.13.0
eu.peppol.bis3:creditnote:3.14.0
eu.peppol.bis3:despatch-advice:3.13.0
eu.peppol.bis3:despatch-advice:3.14.0
eu.peppol.bis3:invoice-cii:3.14.0
eu.peppol.bis3:invoice-message-response:3.13.0
eu.peppol.bis3:invoice-message-response:3.14.0
eu.peppol.bis3:invoice:3.13.0
eu.peppol.bis3:invoice:3.14.0
eu.peppol.bis3:mlr:3.13.0
eu.peppol.bis3:mlr:3.14.0
eu.peppol.bis3:order-agreement:3.13.0
eu.peppol.bis3:order-agreement:3.14.0
eu.peppol.bis3:order-response:3.13.0
eu.peppol.bis3:order-response:3.14.0
eu.peppol.bis3:order:3.13.0
eu.peppol.bis3:order:3.14.0
eu.peppol.bis3:punch-out:3.13.0
eu.peppol.bis3:punch-out:3.14.0
eu.peppol.directory:businesscard:1.0.0
eu.peppol.directory:businesscard:2.0.0
eu.peppol.directory:businesscard:3.0.0

- pdf: Create a PDF/A file from any input PDF. Requires a PDF file (plain PDF, PDF A/1, PDF/A-3 or PDF/X). This operation will remove all non-PDF/A features as well as any embedded files, including potentially available Factur-X/ZUGFeRD files, and embed only available fonts.

- parse: Read a Factur-X/ZUGFeRD/XRechnung and create a JSON representation. Requires a file.

- Invoice2xml: Convert a Factur-X/ZUGFeRD/XRechnung JSON representation to XML. Requires a input JSON string, a standard (ZUGFeRD = zf, XRechnung = xr, or Factur-X = fx), a version (usually 2 for ZUGFeRD and 1 for Factur-X) and a profile ("MINIMUM","BASICWL","BASIC","EN16931","EXTENDED" or "XRECHNUNG" for Factur-X, for ZUGFeRD 1 "BASIC","COMFORT" or "EXTENDED"). For XRechnung only "XRECHNUNG".

- extract: Extracts just the XML (not as JSON like parse) from a Factur-X/ZUGFeRD/Order-X file.

- combineXML: Combines CII XML and a PDF/A-1 document to a Factur-X/ZUGFeRD PDF/A-3 document. Requires a input PDF/A-1 file and a standard (ZUGFeRD = zf or Factur-X = fx), a version (usually 2 for ZUGFeRD and 1 for Factur-X) and a profile ("MINIMUM","BASICWL","BASIC","EN16931","EXTENDED" or "XRECHNUNG" for Factur-X, for ZUGFeRD 1 "BASIC","COMFORT" or "EXTENDED").

- cii2ubl: transforms XML from the UN/CEFACT Cross Industry Invoice (CII) XML format, the basis of factur-x/ZUGFeRD and the CII version of the XRechnung, to the Universal Business Language format, UBL. Requires a CII string.

Mediation, Transformation and Orchestration

The http://api.usegroup.de/ uses WSO² as API management which in turn uses Apache Synapse (https://synapse.apache.org/) for mediation/transformation/orchestration. This means that mediation and orchestration can be developed e.g. in WSO²'s Integration Studio (https://wso2.com/integration/integration-studio/) and uploaded as XML file. Apart from acting as a load balancer and central authentication this allow to

- override certain states in the process, e.g. implement a timeout after a certain number of seconds

- invoke a chain of operations in only one virtual endpoint, e.g. conversion from plain PDF, parallely converting invoice data to XML, merging PDF/A and XML and validation thereof and/or

- map any custom specific input- or output parameter to the values used by Mustangserver internally

Client development

For all clients outlined in this document Oauth's client credentials will be used. Before first use you will have to enable client credentials in the applications tab,

https://api.usegroup.de:9443/devportal/applications Default Application, Oauth2 tokens. The procedure is described more detailed in the PHP Client chapter „Allowing Client Credentials" on page 7 but is generic to all examples and does not apply for PHP only.

Please note that most clients will use the Mustangserver version which had been *selected* in the backend when downloading the OpenAPI definition. New versions may be retired as soon as six month after release of the successor. It is possible to always use the latest (more precisely: recommended) version by removing the version from the endpoint:

```
-                "https://gw.usegroup.de:8243/mustang/v0.5.0/mustang/validate",
+                "https://gw.usegroup.de:8243/mustang/mustang/validate",
```

Example PHP Client

This example operates in a PHP context but https://editor.swagger.io/ also allows

C#, Dart, HTML, Go, Java, Javascript, Kotlin, Python, R, Ruby, Scala, Swift and Typescript.

Mustangserver Hello World

Preparations

Screenshot 1:
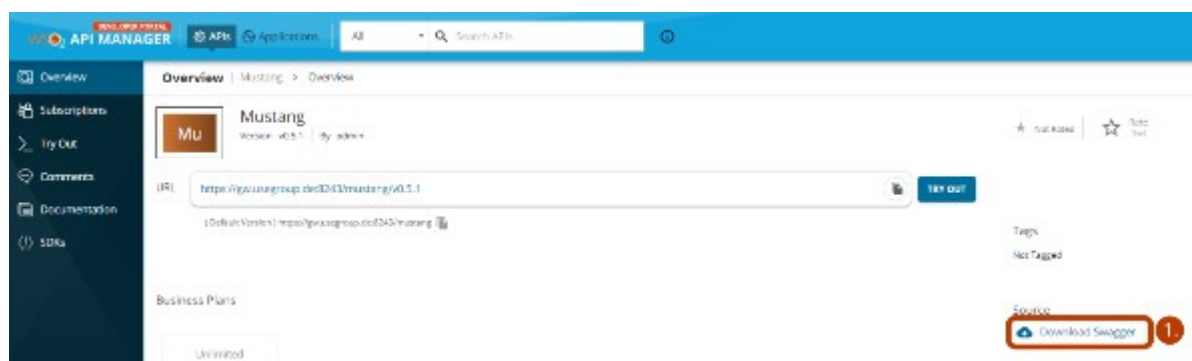
#1. Log in on https://api.usegroup.de/devportal/ , select the latest Mustangserver API and download the OpenAPI (=Swagger) definition of the API (->1.)
#2. Open the file in a text editor, select all and copy
#3. Go to editor.swagger.io, paste the definition and confirm conversion to yaml. Select Generate Client|PHP (3.). The API is public so usually there is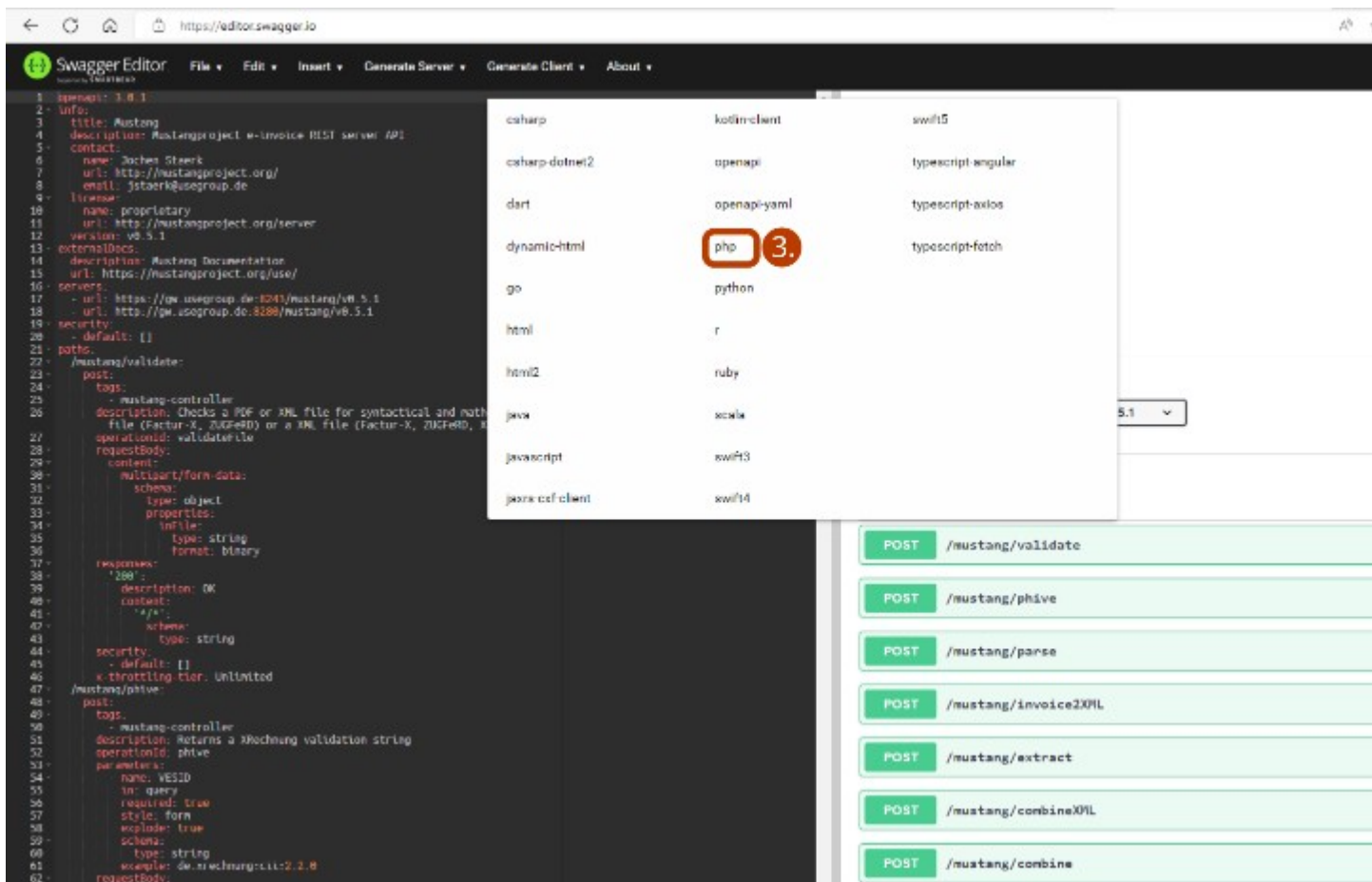 no need to create code in a private matter. However, it is possible: Swagger editor is open source under the APL license (https://github.com/swagger-api/swagger-editor) and e.g. a Docker Image can bei obtained from https://registry.hub.docker.com/r/sebp/swagger-editor , i.e. using

```
sudo docker run --rm -p 8080:8080 sebp/swagger-editor
```
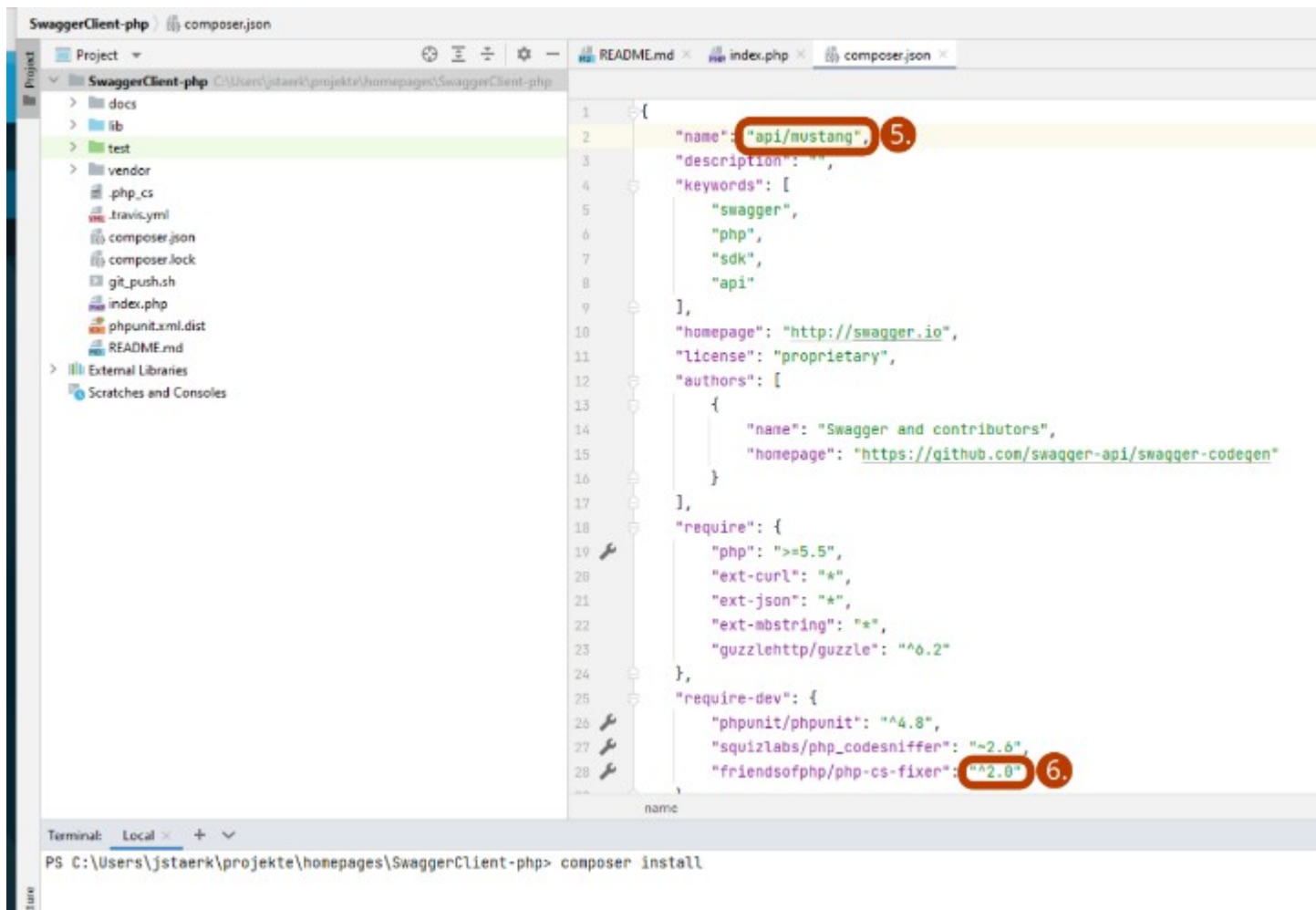
to run locally via port 8080.

Screenshot 2:

#4. Extract the downloaded file, edit composer.json.

#5. change the name of the project in the composer.json file to lowercaps/lowercaps (5.)
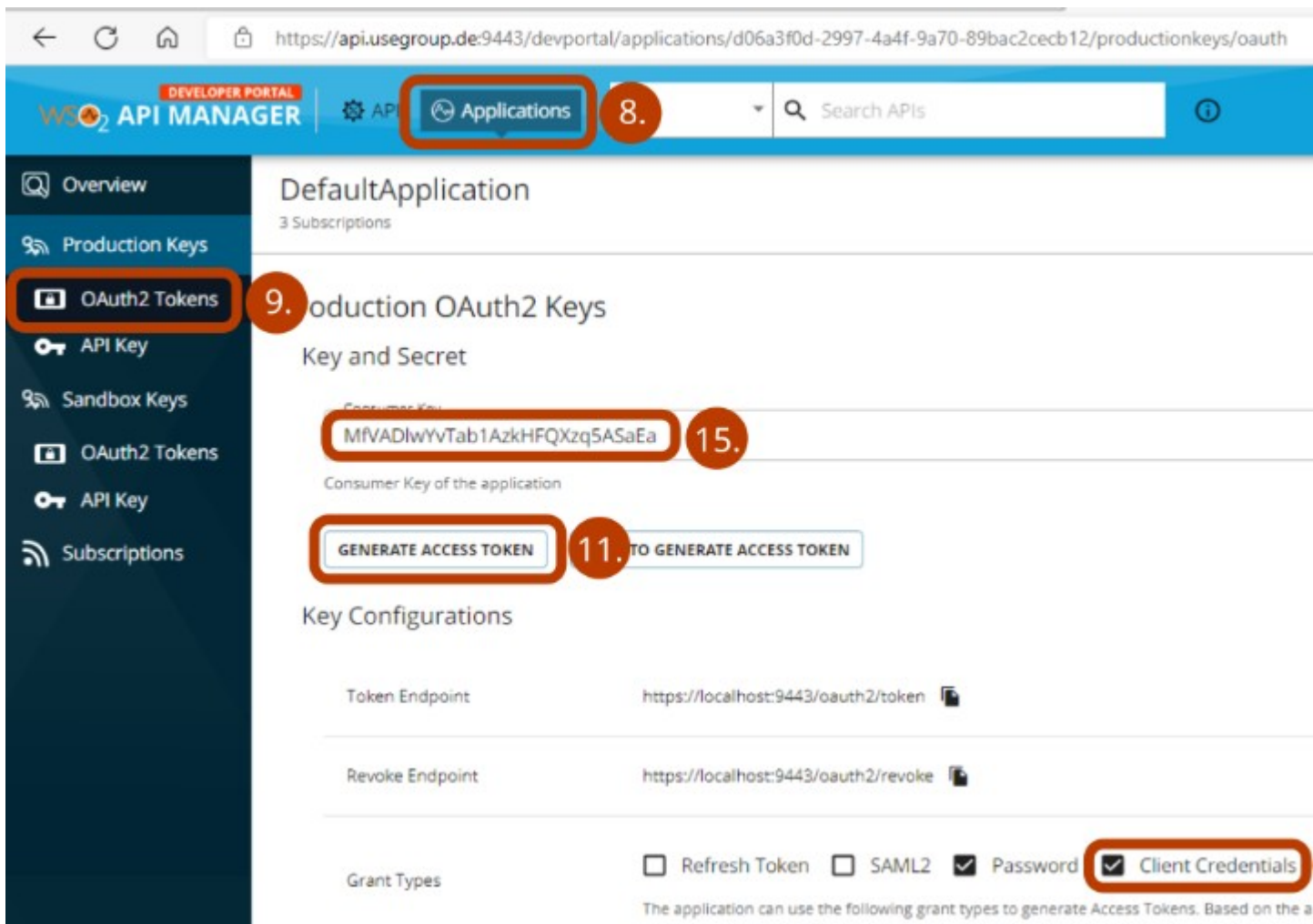
Screenshot 3:

#6. If you want to use PHP8+ upgrade the version number of php-cs-fixer to ^2.0. Then run "composer install" in that directory.
#7. Copy the example from the "Getting started" section of the readme.md to a new file, called index.php

Allowing Client Credentials

Screenshot 4:

#8. Click on Applications (8.), Default Application,
#9. Production Keys/OAuth2 Token (9.).

#10. As preparation for Part B: Check Client Credentials and click the Update button on the bottom of the page.

Get access token

#11. For this part we will use a token which will expire shortly. Click Generate Access Token (11.), Generate and copy the resulting token. Paste it in

Screenshot 5:

```php
<?php
require_once(__DIR__ . '/vendor/autoload.php');

// Configure OAuth2 access token for authorization: default
$config = Swagger\Client\Configuration::getDefaultConfiguration()->setAccessToken( access 'eyJ4NXQiOiJNbVZpTnpFMU1HVm
```
⑫
```php
$apiInstance = new Swagger\Client\Api MustangControllerApi( ⑬
// If you want use custom http client, pass your client which implements `GuzzleHttp\ClientInterface`.
// This is optional, `GuzzleHttp\Client` will be used as default.
    new GuzzleHttp\Client(),
    $config
);

try {
    $result = $apiInstance->pi(); ⑭
    print_r($result);
} catch (Exception $e) {
    echo 'Exception when
}
```

#12. index.php (12.), in the same file

#13. change ErrorController to Mustangcontroller (13.) and
#14. handle() to ping() (14.). Please note that usual PHP editors will give you code completion.
Now you can open resulting index.php via your server and PHP processsort in your browser, it
should now look like Screenshot 6:

pong

Part B: Sustainable OAuth2 Authentication and Validation of electronic invoices

Back to Screenshot 4:

In index.php paste the following code

```
$client = new GuzzleHttp\Client();
$res = $client->request('POST', 'https://gw.usegroup.de:9443/oauth2/token', [
    'auth' => ['<15.>', '<16.>'],
    'form_params' => [
        'grant_type' => 'client_credentials',
    ]
]);

$json = json_decode($res->getBody(), true);
```

Screenshot 7:

#15. copy Consumer Key (15., from screen 4) to the beginning of index.php (16.),
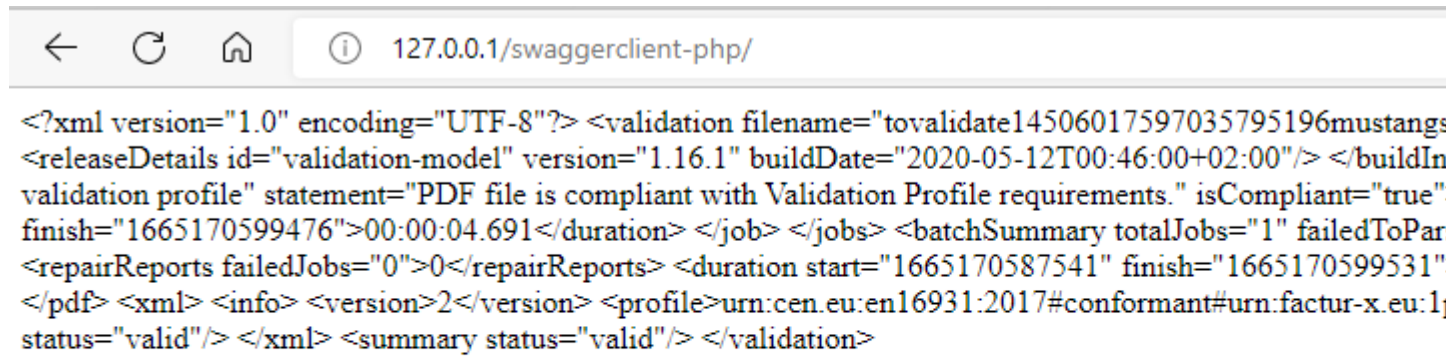
#17. reveal and copy Consumer Secret (18.).

#19 replace the static access token which will become invalid by $json["access_token"]

#20 Create or download a invoice to be validated, e.g. https://www.mustangproject.org/files/MustangGnuaccountingBeispielRE-20201121_508.pdf and save it as factur-x.pdf
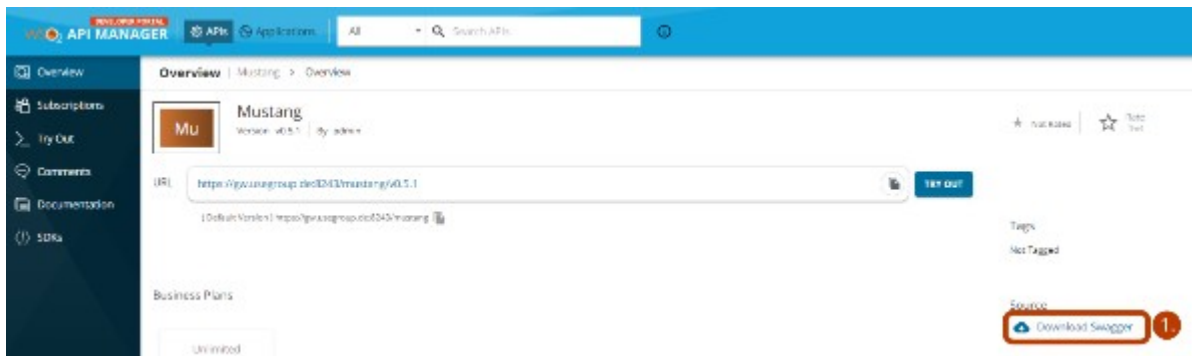
#21 Change the method to validateFile and

#22 html escape the validation result, so that the result in the browser looks like screenshot 8:

<?xml version="1.0" encoding="UTF-8"?> <validation filename="tovalidate14506017597035795196mustangs
<releaseDetails id="validation-model" version="1.16.1" buildDate="2020-05-12T00:46:00+02:00"/> </buildIn
validation profile" statement="PDF file is compliant with Validation Profile requirements." isCompliant="true"
finish="1665170599476">00:00:04.691</duration> </job> </jobs> <batchSummary totalJobs="1" failedToPar
<repairReports failedJobs="0">0</repairReports> <duration start="1665170587541" finish="1665170599531"
</pdf> <xml> <info> <version>2</version> <profile>urn:cen.eu:en16931:2017#conformant#urn:factur-x.eu:1
status="valid"/> </xml> <summary status="valid"/> </validation>

That's it. Instead of displaying the XML you can now parse it :-)

Feel free to also try the async functions.
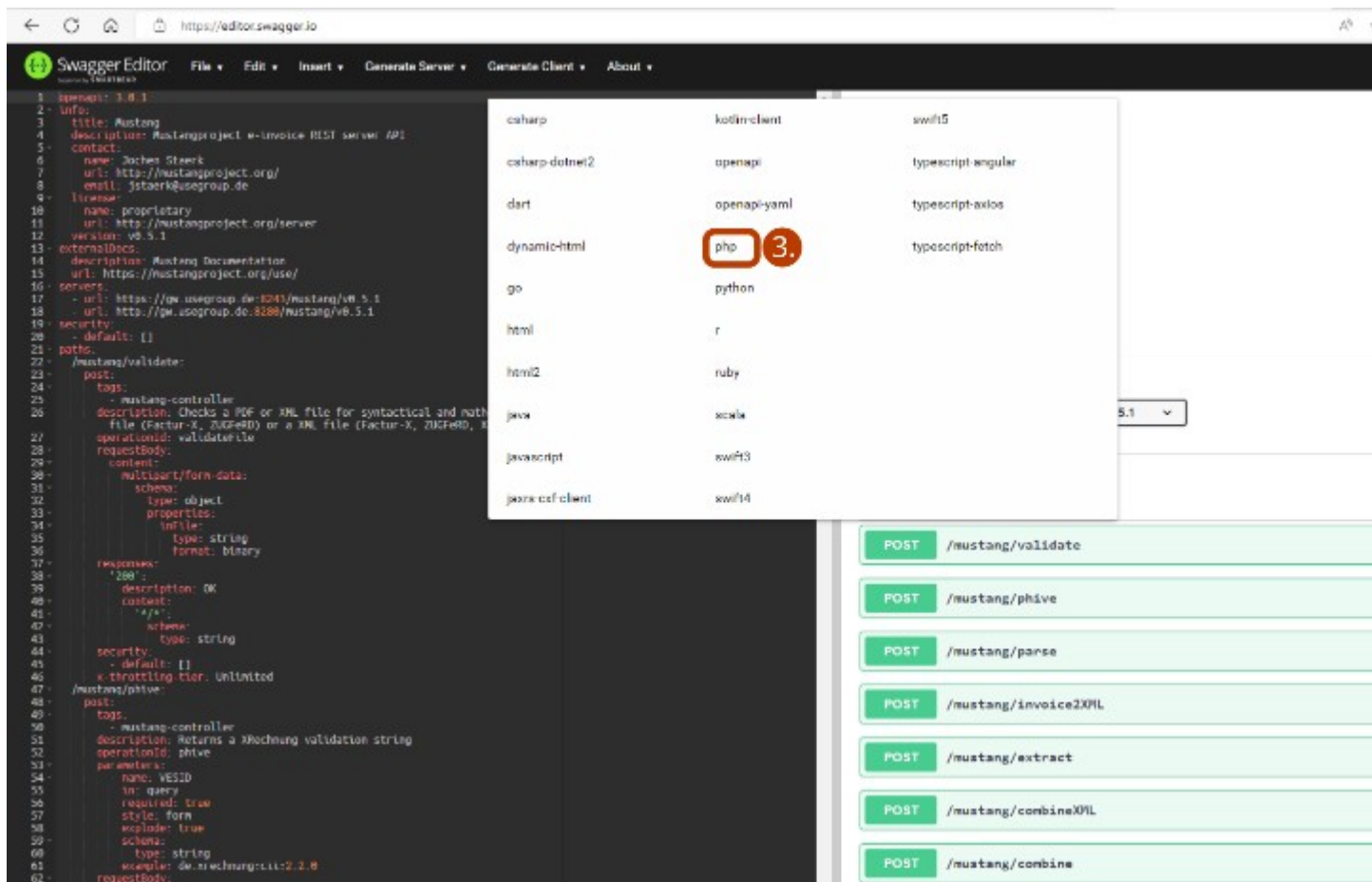
Screenshot 1:

#1. Log in on https://api.usegroup.de/devportal/ , select the latest Mustangserver API and download the OpenAPI (=Swagger) definition of the API (->1.)
#2. Open the file in a text editor, select all and copy
#3. Go to editor.swagger.io, paste the definition and confirm conversion to yaml. Select Generate Client|PHP (3.)
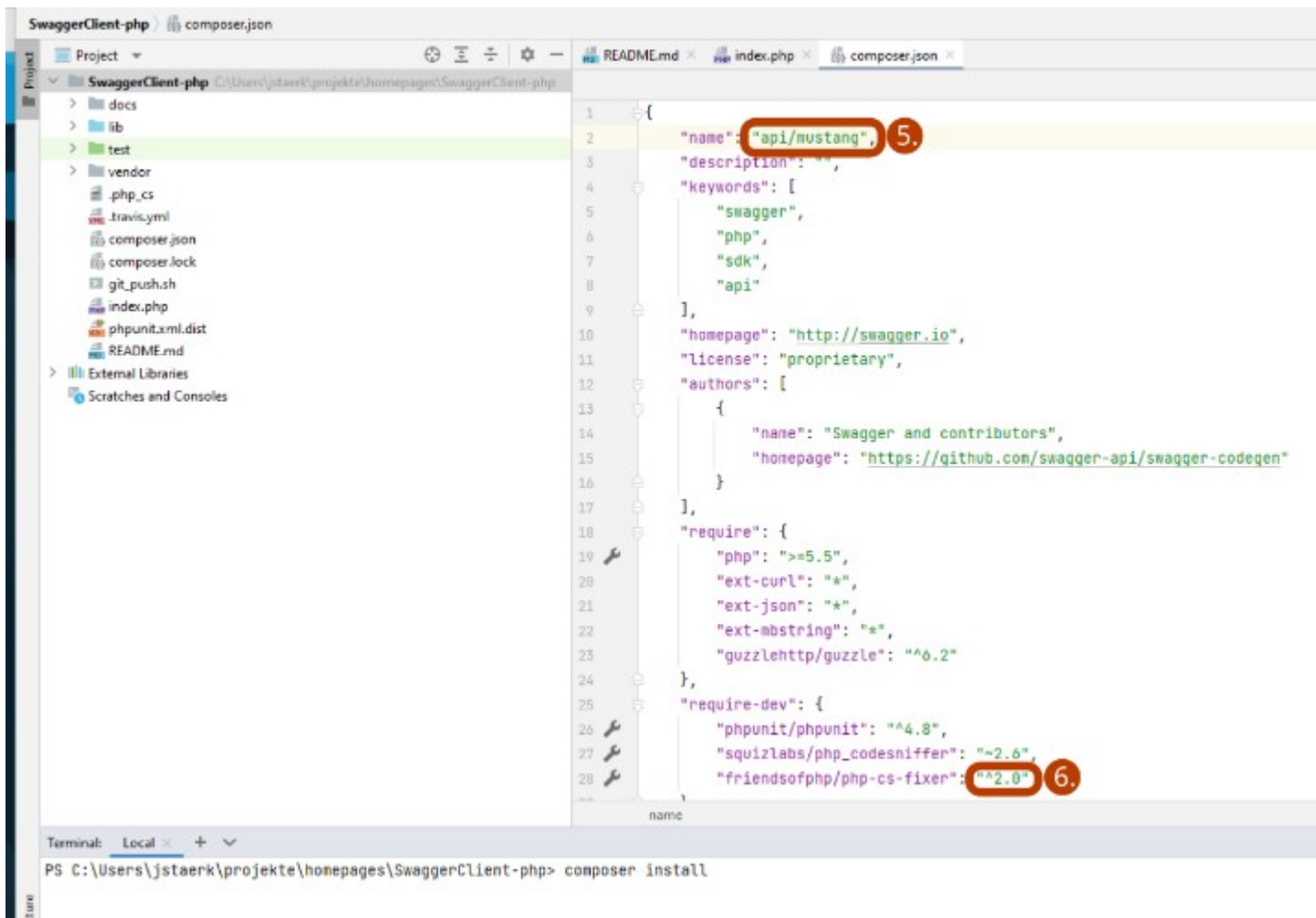
Screenshot 2:

#4. Extract the downloaded file, edit composer.json.

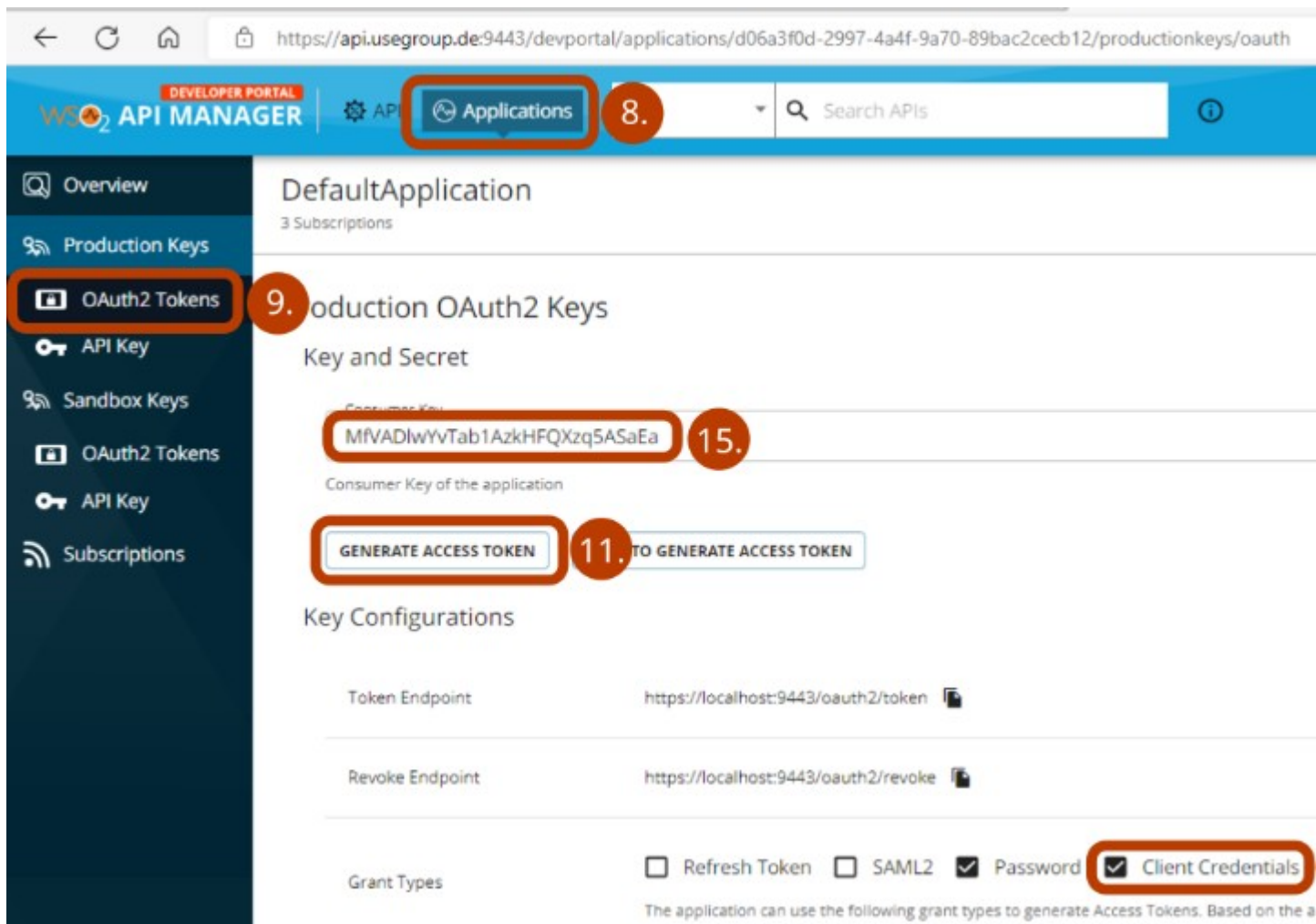#5. change the name of the project in the composer.json file to lowercaps/lowercaps (5.)

Screenshot 3:

#6. If you want to use PHP8+ upgrade the version number of php-cs-fixer to ^2.0. Then run "composer install" in that directory.

#7. Copy the example from the "Getting started" section of the readme.md to a new file, called index.php

Screenshot 4:

#8. Click on Applications (8.), Default Application,
#9. Production Keys/OAuth2 Token (9.).

#10. As preparation for Part B: Check Client Credentials and click the Update button on the bottom of the page.
#11. For this part we will use a token which will expire shortly. Click Generate Access Token (11.), Generate and copy the resulting token. Paste it in

Screenshot 5:

```php
<?php
require_once(__DIR__ . '/vendor/autoload.php');

// Configure OAuth2 access token for authorization: default
$config = Swagger\Client\Configuration::getDefaultConfiguration()->setAccessToken( access 'eyJ4NXQiOiJJNbVZpTnpFMU1HVm

$apiInstance = new Swagger\Client\Api MustangControllerApi(
// If you want use custom http client, pass your client which implements `GuzzleHttp\ClientInterface`.
// This is optional, `GuzzleHttp\Client` will be used as default.
    new GuzzleHttp\Client(),
    $config
);

try {
    $result = $apiInstance->pi();
    print_r($result);
} catch (Exception $e) {
    echo 'Exception when
}
```

#12. index.php (12.), in the same file

#13. change ErrorController to Mustangcontroller (13.) and
#14. handle() to ping() (14.). Please note that usual PHP editors will give you code completion.
Now you can open resulting index.php via your server and PHP processort in your browser, it should now look like Screenshot 6:

pong

OAuth2 Authentication and Validation of electronic invoices

Concerning Screenshot 4:
In index.php paste the following code

```
$client = new GuzzleHttp\Client();
$res = $client->request('POST', 'https://gw.usegroup.de:9443/oauth2/token', [
    'auth' => ['<15.>', '<16.>'],
    'form_params' => [
        'grant_type' => 'client_credentials',
    ]
]);

$json = json_decode($res->getBody(), true);
```

Screenshot 7:

#15. copy Consumer Key (15., from screen 4) to the beginning of index.php (16.),

#17. reveal and copy Consumer Secret (18.).

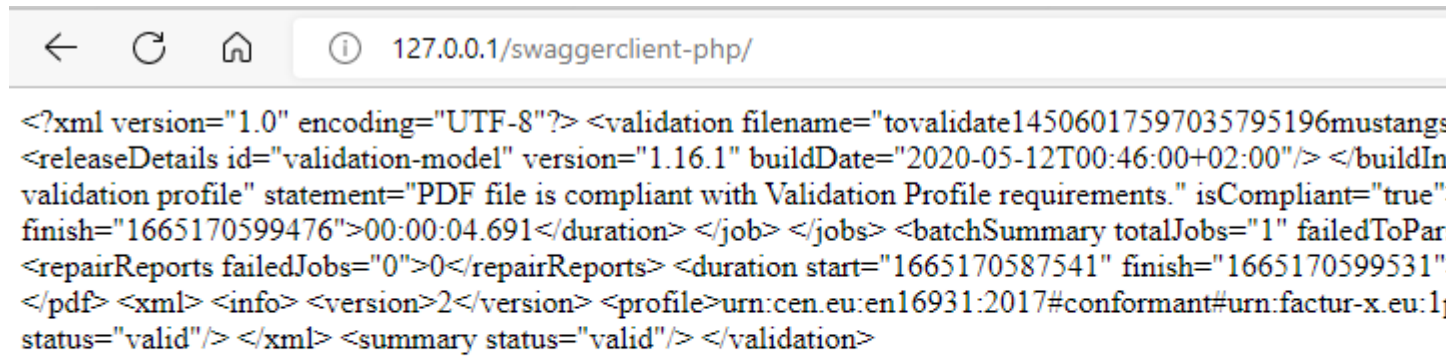#19 replace the static access token which will become invalid by $json["access_token"]

#20 Create or download a invoice to be validated, e.g.
https://www.mustangproject.org/files/MustangGnuaccountingBeispielRE-20201121_508.pdf and
save it as factur-x.pdf

#21 Change the method to validateFile and

#22 html escape the validation result, so that the result in the browser looks like screenshot 8:

That's it. Instead of displaying the XML you can now parse it.

Feel free to also try the async functions.

Postman

For this example, you will need enabled client credentials as described on page 7 in Allowing Client Credentials.

Use Import|File|Upload Files to upload you Openapi.yaml file into a Mustangserver collection.

Add a request to https://gw.usegroup.de:9443/oauth2/token?grant_type=client_credentials and call it Token Request. Postman will auto-detect the Parameter in the URL. Change the type to POST.

In Headers, add a new field Content-Type with application/json as its value.

The tab Authorization should be Basic auth with your client id and secret as username and password. Once you click Send, an according access token should be submitted:

Add
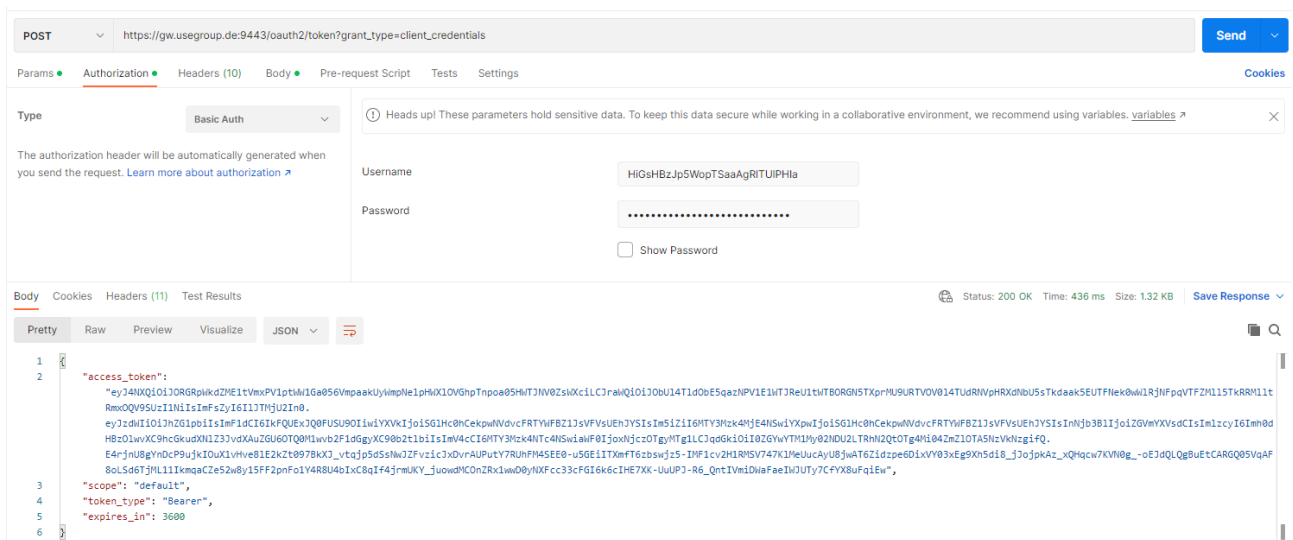
```javascript
var jsonData = JSON.parse(responseBody);
pm.collectionVariables.set("token", jsonData.access_token);
console.log(jsonData.access_token);
```

In the „Tests" tab and View|Show Postman Console (Alt+CTRL+C) . Once you click Send again you should be able to see the access token also in the Console:
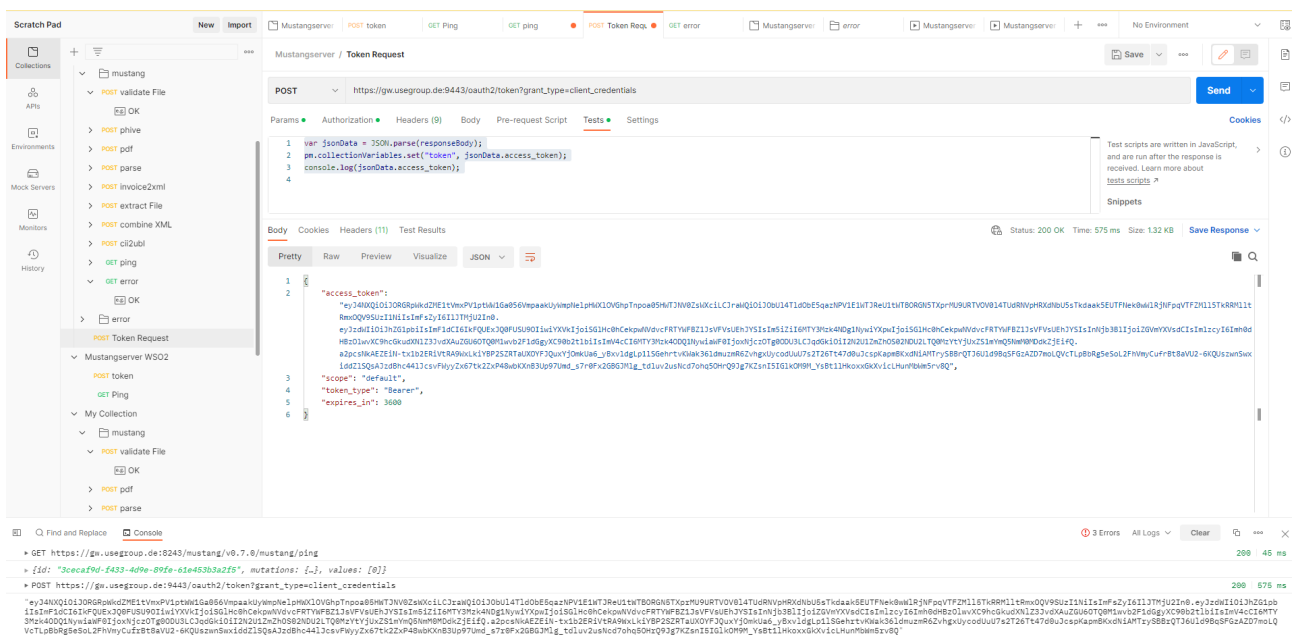
In the collection, click on the variables tab and add „token" as a collection variable.

Watch 0 Fork Run Save Share •••

Authorization ●   Pre-request Script   Tests   Variables ●

hese variables are specific to this collection and its requests. Learn more about collection variables ↗

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ		•••	Persist All	Reset All
☑	baseUrl	https://gw.usegroup.de:8243/mustang/v0.7.0	https://gw.usegroup.de:8243/mustang/v0.7.0				
☑	token						
	Add a new variable						

You can now add the variable {{token}} as authorization to any request. The variable will be available and the requests work after you click on „Send" of the Token Request for the first time.

Mustangserver / mustang / ping   🖫 Save ⌄ ••• ✎ ▭

GET ⌄   {{baseUrl}}/mustang/ping   **Send** ⌄

Params   Authorization ●   Headers (10)   Body   Pre-request Script   Tests ●   Settings   Cookies

Type   Bearer Token ⌄

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

ⓘ Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. variables ↗   ✕

Token   {{token}}

Body   Cookies   Headers (18)   Test Results   Status: 200 OK   Time: 45 ms   Size: 704 B   Save Response ⌄

Pretty   Raw   Preview   Visualize   Text ⌄

```
1  pong
```

Please note that ping was answered by pong.

Where appropriate, e.g. in the validation endpoint, Postman will allow you to select files, this being a valid factur-x:

Mustangserver / mustang / **validate File**   🖫 Save ⌄ ••• ✎ ▭

POST ⌄   {{baseUrl}}/mustang/validate   **Send** ⌄

Params   Authorization ●   Headers (11)   Body ●   Pre-request Script   Tests   Settings   Cookies

○ none   ● form-data   ○ x-www-form-urlencoded   ○ raw   ○ binary   ○ GraphQL

	KEY	VALUE	DESCRIPTION		•••	Bulk Edit
☑	inFile	factur-x (6).pdf ✕				
	Key	Value	Description			

Body   Cookies   Headers (18)   Test Results   Status: 200 OK   Time: 13.65 s   Size: 2.52 KB   Save Response ⌄

Pretty   Raw   Preview   Visualize   Text ⌄

```
18        <duration start="1673985270293" finish="1673985273311">00:00:03.018</duration>
19      </job>
20    </jobs>
21    <batchSummary totalJobs="1" failedToParse="0" encrypted="0">
22      <validationReports compliant="1" nonCompliant="0" failedJobs="0">1</validationReports>
23      <featureReports failedJobs="0">0</featureReports>
24      <repairReports failedJobs="0">0</repairReports>
25      <duration start="1673985260818" finish="1673985273315">00:00:12.497</duration>
26    </batchSummary>
27  </report>
28  <info>
29    <signature>Mustang</signature>
30    <duration unit="ms">12740</duration>
31  </info>
32  <summary status="valid"/>
33  </pdf>
34  <xml>
```

🔍 Find and Replace   ▣ Console   ⚠ 3 Errors   All Logs ⌄   Clear   ▭ •••

'id: "3cecaf9d-f433-4d9e-89fe-61e453b3a2f5", mutations: {…}, values: [0]}
GST https://gw.usegroup.de:9443/oauth2/token?grant_type=client_credentials   200 5?

rJ4NXQiOiJORGRpWkdZME1tVmxPV1ptWW1Ga056VmpaakUyWmpNelpHWXlOVGhpTnpoa0B5HWTJNV8ZsWXciLCJraWQiOiJObUl4TldO0bE5qazNPV1E1WTJReU1tWTBORGN5TXpzMU9URTVOV0l4TUdRNVpHRXdNbU8sTkdaak5EUTBZHEJIkdWN9VdvcFRTYWFBZ1JsVFVsUEhJYSIsIn5iIZiI6MTY3Mzk4NDg1Nywi1VXpwIjoiSGlHc0hCekpwNVdvcFRTYWFBZ1JsVFVsUEhJYSIsInJb3BlIjoiZGVmYXVsdCIsInlmlzcyI6Imh0dHBz0lwvXC9hcGkudXNlZ3JvdXAuZGU6OTQ8M1wvb2F1dGgyXC90b2tlbiIsInMV4cCI6N
rk4ODQ1Nywi1aWF8Ijox NjczOTg0ODU3LCJqdGki0iI2N2U1ZmZh0S82NDU2LTQ0MzYtYjUxZS1mYmQ6NmM8MDdkZjEi1Q.a2pcsNkAEZEiN-tx1b2ERiVtRA9WxLkiYBP2SZRTaUXOYFJQuxYjOmkUa6_yBxvldgLp1lSGehrtvKWak36ldmuzmR6ZvhgxUycodUuU7s2T26Tt47d0uJcspKapmBKxdNiAMTzySBBzQTJ6Uld9BqSFGzAZD7m
fLpBbRg5eSoL2FhVmyCufrBtBaVU2-6KQUszwnSwxiddZl5QsAJzdBhc44lJcsvFWyyZx67tk2ZxP48wbKXnB3Up97Umd_s7r0Fx2GBGJMlg_tdluv2usNcd7ohq5OHzQ93g7KZsnI6IGlkOM9M_YsBt11HkoxxGkXvicLHunMbWm5zvBQ˙
GST https://gw.usegroup.de:8243/mustang/v0.7.0/mustang/validate   200 13

▶ Runner   🗑 Trash

Jmeter

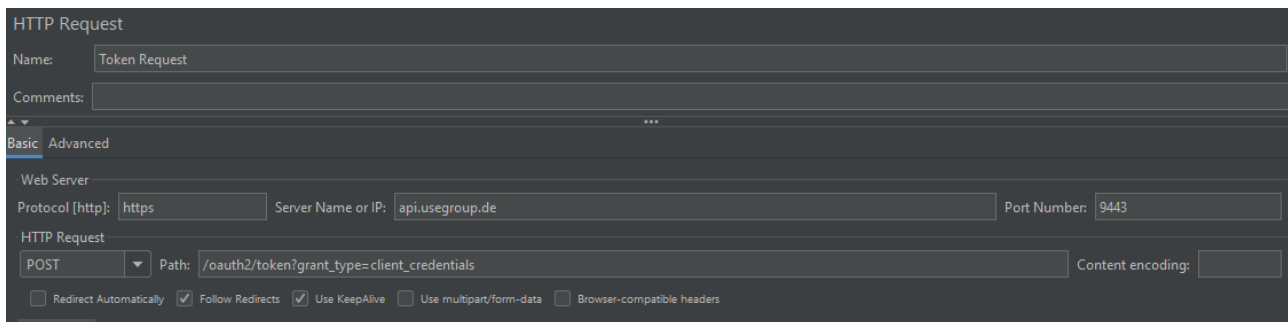https://openapi-generator.tech/ supports a Jmeter export but that does not handle authentication so this is a simple way to do some performance testing manually.

For this example, you will need enabled client credentials as described on page 7 in Allowing Client Credentials.

In your Test plan, add a Thread Group with a Once Only controller. Below that, add a HTTP request sampler, we'll call it Token Request. This is how it is defined: Change protocol to https, method to POST, add server name and port number, and add the path:

Base64encode your <consumer key>:<consumer secret> as described on the applications page of the API management:
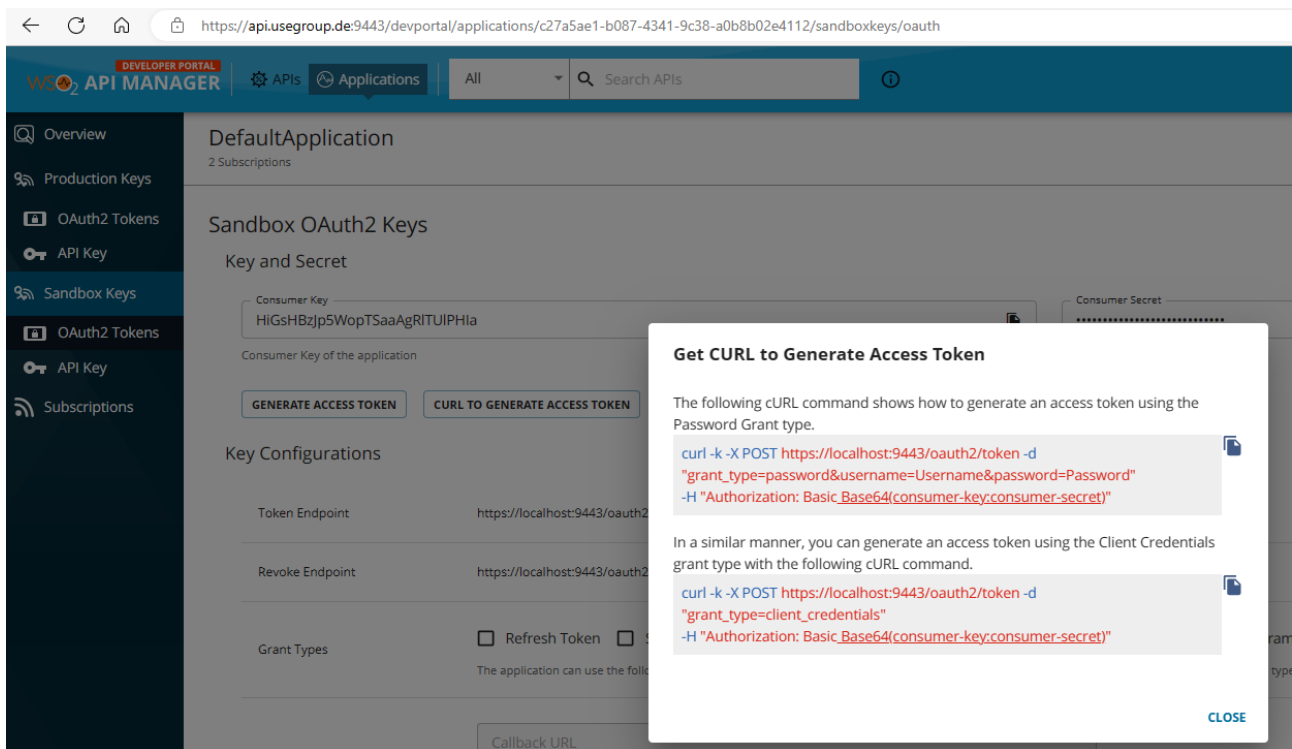
In Jmeter, below the Token request, add a HeaderManager with Basic Authorization as described:
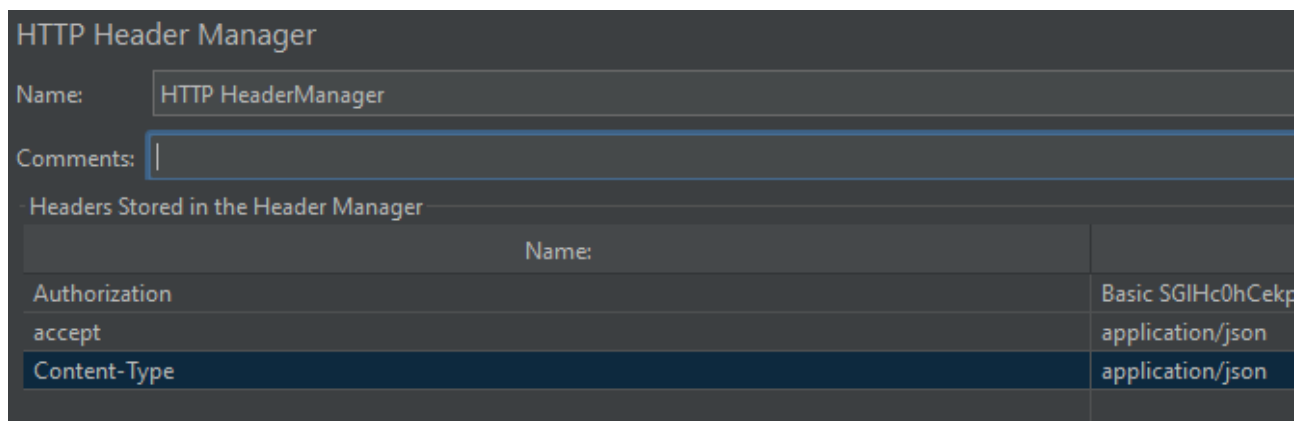
And from the response, also below Token Request, extract the JSON value access token into a Jmeter Variable access token using a JSON Extractor:

Add a sampler View Result Tree to confirm the results and a debug sampler if you like (in the results tree you will then be able to e.g. see the current variables when you click on the results of the debug sampler).

Run|Start should give you green entries in the results tree.

Now we will set the ordinary authentication as header: add another Header Manager outside of the Token request and add the variable as token, i.e. Authorization being Bearer ${access_token}

You can then add a simple ping request in the thread group

and e.g. a Response time graph.You can then set the Thread group loop count to infinite, start the sampling and check the results tree. After a while the response time graph will look like this, indicating the initial login took ~440ms and the usual response time to our „ping" is ~20ms.
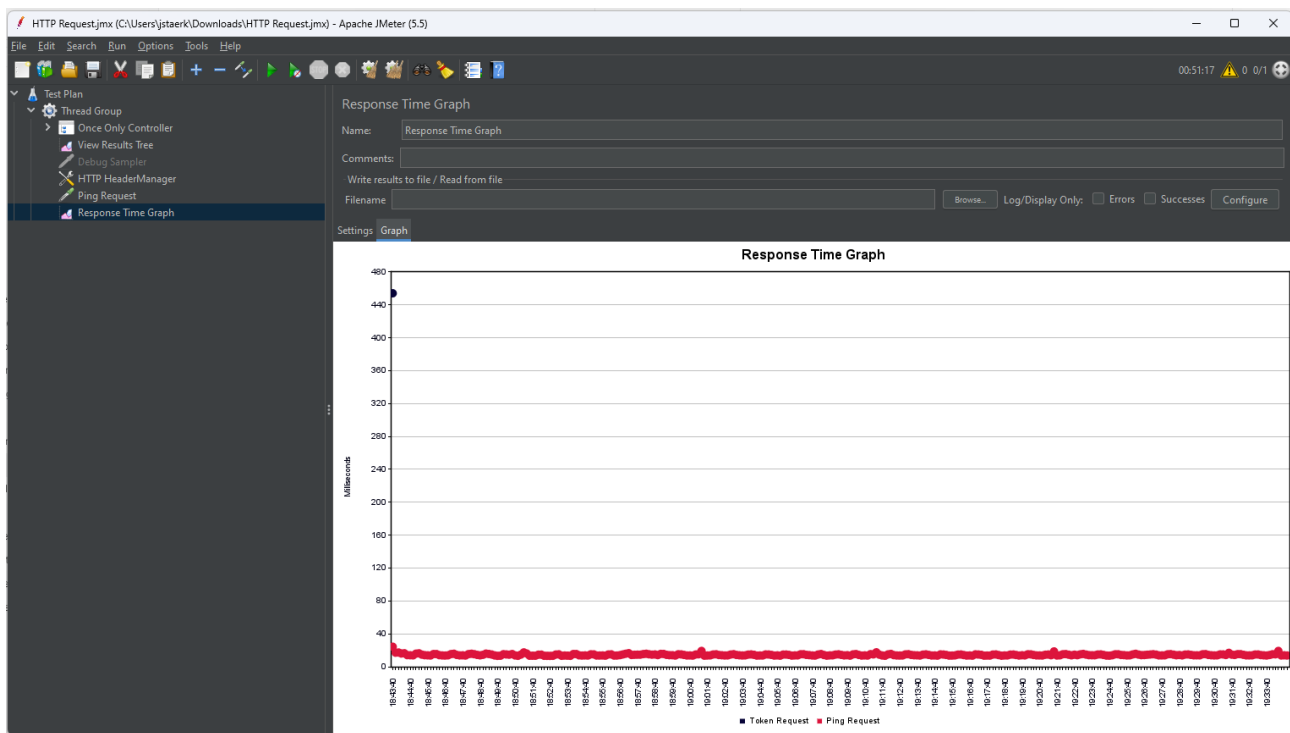
Terms of service

Test terms

To test and evaluate the service a valid email address has to be provided. Unless otherwise agreed (info@usegroup.de) test access is restricted to one account per legal entity, i.e. usually company. This email address will also be used to send availability, information about the roadmap, development and status with an expected maximum volume of one per week. You can terminate your test phase by unsubscribing from the announcements newsletter list. After the signup, access can then happen free of charge, with a limit of 1,000 operations/month, unless access is revoked by usegroup. You are not allowed to share personal data (e.g. real invoice recipient's names, addresses, email addresses, bank credentials or real invoice contents). Access may be revoked because the general test phase has ended, the test phase is over for a certain customer, or due to other terms which do not need to be disclosed. Under this test terms we also do not guarantee the availability of the service.

https://api.usegroup.de:9443/authenticationendpoint/privacy_policy.do

Production terms

To access Mustangserver productively including a data processing agreement a Mustang Pro license is required. Further info can be obtained at https://www.mustangproject.org/pro/