Realization

Festo Internship

Wout Hebberecht
r0844999

3CCS01

Academiejaar 2022-2023

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

Preface

This document give a more detailed view of my internship at Festo. Here I will explain what I did and my final result.

I did not work on this task alone. There was another student that helped on this task. Together with my mentor we worked on this task with 3 people.

I want to tell you that this task is not finished when my contact ended. They will keep working on this until September. In September they will show a POC to the bosses. After that, they will decide if they want to continue the project or stop with it.

Contents Table

Content

1 The Task

Festo is a multinational company. To keep everybody connected, they need a good IT platform. They decided on buying the rights to use ServiceNow. It makes it possible for employees to share information, ask for help, buy company related equipment, etc.

My task was to make it easier for employees to help themselves with common IT problems. I had to make this possible with the built-in Chatbot from ServiceNow. This will take some work off the level 1 support so they have more time for more difficult tasks.

1.1 Agreements

1.1.1 Work time

This internship happened from the 8 of March until the first of June. I had to work and be available on Teams every weekday from 10 AM till 4 PM.

1.1.2 Office hours

We agreed to be in office every Wednesday. The other days we could work from home if we wanted.

So that is what I did. On Wednesday I went to the office and the other days I stayed home. I only went to the office on the other days if I had to show some documents to my mentor.

1.1.3 Equipment

I got a laptop and all the necessary accessories. (Headphone, charger, computer mouse, backpack).

1.1.4 Meetings

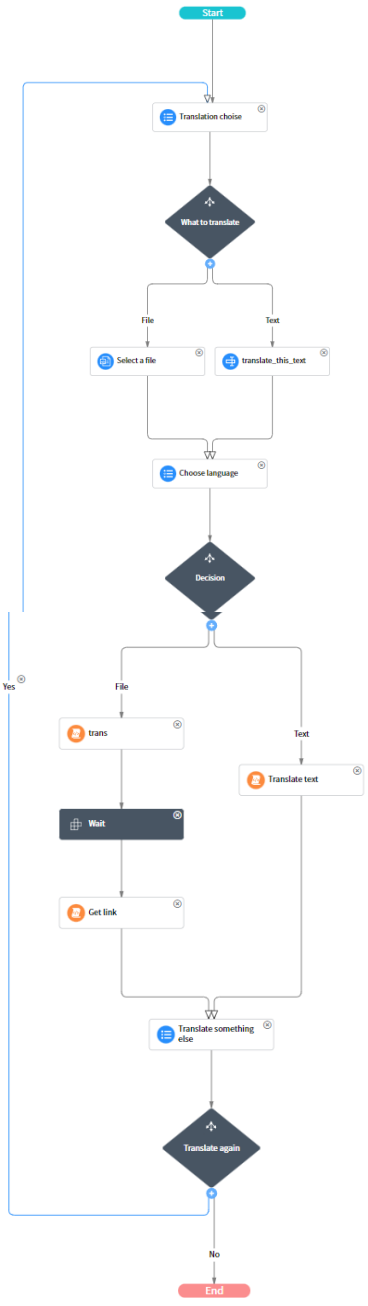
Every day there was a meeting at 10:30 AM. Sometimes the time would change if my mentor was busy.
There were also sprint meeting every week on Wednesday. My colleague and I would share our knowledge we gathered the past week during this meeting. We would also decide what parts should get priority.

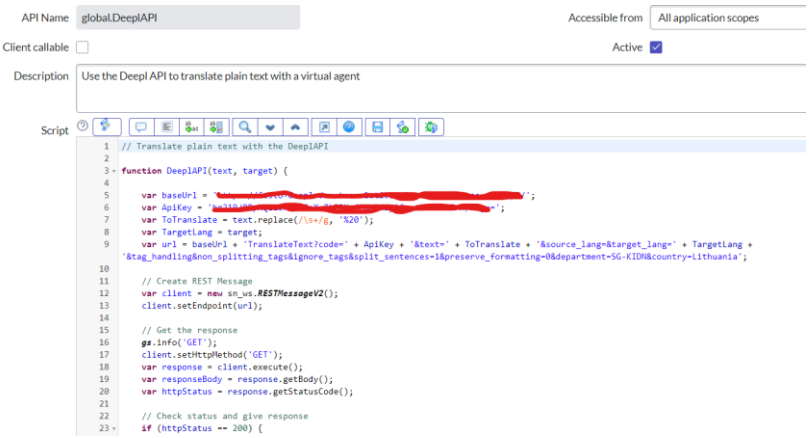
2 Used Technology

2.1 ServiceNow

ServiceNow is a Paas & Saas solution. It already has a lot of pre-build stuff you can use, but there is a lot you can modify to your specifications. It is possible to create these specifications with Javascript and save them on the ServiceNow server. The JavaScript files on ServiceNow have access to functions that are created by ServiceNow. This makes it easier to access certain things, like the databases that it uses, for example.

ServiceNow try's to give a lot of no code/low code solutions. But this is very limited.





2.2 JavaScript Code – Script Include

Script include gives us the possibility to create 1 script and use it in multiple places on the ServiceNow instance. Here are the script I wrote and stored in ServiceNow Script Include

2.2.1 DeeplAPI

This script sends an API request to the Deepl server to translate plain text.

We only have to provide the text and to what language we want to translate.

Code:

```
function DeeplAPI(text, target) {
    // var mysum = int1 + int2;
    // gs.addInfoMessage('The sum is: ' + mysum);

    var baseUrl =’https://www.#######.com ';
    var ApiKey = ‘#######';
    var ToTranslate = text.replace(/\s+/g, '%20');
    var TargetLang = target;
    var url = baseUrl + 'TranslateText?code=' + ApiKey + '&text=' +
ToTranslate + '&source_lang=&target_lang=' + TargetLang +
'&tag_handling&non_splitting_tags&ignore_tags&split_sentences=1&preser
ve_formatting=0&department=SG-KIDN&country=Lithuania';

    // Create REST Message
    var client = new sn_ws.RESTMessageV2();
    client.setEndpoint(url);

    // Get the response
    gs.info('GET');
    client.setHttpMethod('GET');
    var response = client.execute();
    var responseBody = response.getBody();
    var httpStatus = response.getStatusCode();

    // Check status and give response
    if (httpStatus == 200) {
        var result = JSON.parse(responseBody);
        var translatedText = result.translations[0].text.toString();

        gs.info('typeof');
        gs.info(typeof translatedText + ' = ' + translatedText);

        // Remove spaces
        // translatedText = text.replace(/\s+/g, '%20');
        //callback(translatedText);
        return translatedText;
    } else {
        gs.error('DeepL API error: ' + responseBody);
    }

    gs.info(translatedText);
    return translatedText;
}
```

2.2.2 GetContentOfFile

This script converts the content to BASE64 so we can send it to the Deepl server. It will use the most recent added file from the attachment table

Code:

```
// Select the most recent file from the Attachment table and converts
the content to BASE64
// It will send an POST request to the DeeplAPI containing the
original language of the document, to which language it needs to be
translated and the BASE64
// It will return a link to download the translated file

function GetContentOfFile(resultLang) {
    var b64attachment = "";
    var allAttachments = "";
    var attachmentData = "";

    // Select the most recent file from the current user
      var currentUserID = gs.getUserID();
    gs.info('Current user sys_id: ' + currentUserID);

    var sa = new GlideRecord('sys_attachment');
      sa.addQuery('sys_created_by', gs.getUserName());
    sa.orderByDesc('sys_created_on');
    sa.query();

    if (sa.next()) {
        // Get info from the Attachment
        var fileName = sa.getValue('file_name');
        var fileSysID = sa.getValue('sys_id');
        var fileExtension = fileName.split('.').pop();

        gs.log('Document Name: ' + fileName);
        gs.log('File extension: ' + fileExtension);
        gs.log('Document Sys_id: ' + fileSysID);

        // Convert content to BASE64
        var attachmentIS = new
GlideSysAttachmentInputStream(sa.sys_id);
        var byteArrayOS = new
Packages.java.io.ByteArrayOutputStream();
        attachmentIS.writeTo(byteArrayOS);
        b64attachment = GlideBase64.encode(byteArrayOS.toByteArray());
        attachmentData = b64attachment;

        gs.log('Get the BASE64 of the uploaded file');
        gs.log(attachmentData);
    } else {
        gs.log('Error: No file found');
    }
```

```
    // The body is created in 'REST Messages Deepl File Translation ->
Post File'

    // Send POST to DeeplAPI and returns download link
    // Use the REST Message and fill in the var
    try {
        var r = new sn_ws.RESTMessageV2('Deepl File Translation',
'Post File');
        r.setStringParameterNoEscape('base64', attachmentData);
        r.setStringParameterNoEscape('slang', '');
        r.setStringParameterNoEscape('filetype', fileExtension);
        r.setStringParameterNoEscape('tlang', resultLang);
        r.setStringParameterNoEscape('filename', 'document.' +
fileExtension);


        var url = 'https://#######';
        var apiKey = '#######';
        var response = r.execute();
        var responseBody = response.getBody();
        var httpStatus = response.getStatusCode();

        // Get only the usefull output from the response
        var obj = JSON.parse(responseBody);
        var id = obj.document_id;
        var key = obj.document_key;

        gs.log("object ID fron Deepl file; " + id);
        gs.log("FileKey from Deepl file: " + key);
        gs.log(responseBody);

        var link = url + '?code=' + apiKey + '&documentKey=' + key +
'&documentId=' + id;
        gs.log('link: ' + link);
            return link;
    } catch (ex) {
        var message = ex.message;
        gs.log(message);
    }
}
type: 'GetContentOfFile';
```

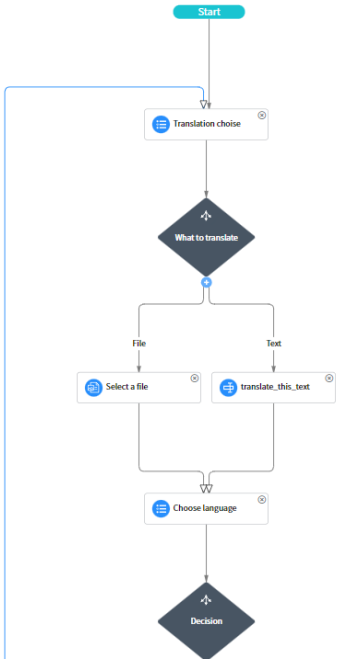
2.2.3 UploadFileToAttachment

This script gets the file from the Deepl server and uploads it to the Attachment table in ServiceNow.

The url variable would contain the url of the file. This url would be created by the getContentOfFile script.

```
function UploadFileToAttachment(url) {
    var translatedSa = new GlideRecord('sys_attachment');
    var singleOutMsg = new sn_cs.SinglePartOutMsg();

    translatedSa.addQuery('sys_created_by', gs.getUserName());
    translatedSa.orderByDesc('sys_created_on');
    translatedSa.query();
    if (translatedSa.next()) {
        // Get info from the Attachment
        var translatedFileName = translatedSa.getValue('file_name');
        var translatedFileSysID = translatedSa.getValue('sys_id');
        gs.log('here ' + translatedFileName);
    }

    var tablename = 'sys_attachment';
    var recordSysId = translatedFileSysID;
    var filename = 'tranlated-' + translatedFileName;

    // Download the file
    var fileUrl = url;
    var request = new sn_ws.RESTMessageV2();
    request.setHttpMethod('get');
    request.setEndpoint(fileUrl);
    // Configure the request to save the response as an attachment
    request.saveResponseBodyAsAttachment(tablename, recordSysId,
filename);

    // When we execute the request, the attachment will automatically
be
    // saved to the record we specified
    var response = request.execute();
    var httpResponseStatus = response.getStatusCode();
    var httpResponseContentType = response.getHeader('Content-Type');
    gs.debug("http response status_code: " + httpResponseStatus);
    gs.debug("http response content-type: " +
httpResponseContentType);
}
type: 'UploadFileToAttachment';
```

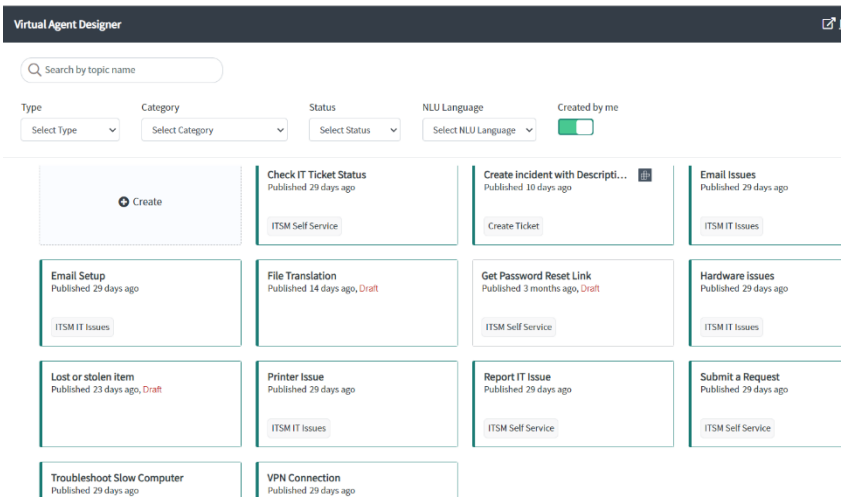
3 Conversation interface



The conversation interface is the place where we can modify the topics for the chatbot. This interface lets me do almost everything by dragging and dropping.

In the picture below are all the topics I worked on. Most of them are Out Of The Box. The topics that asked the most work were:

- Create incident with Description
- File Translation

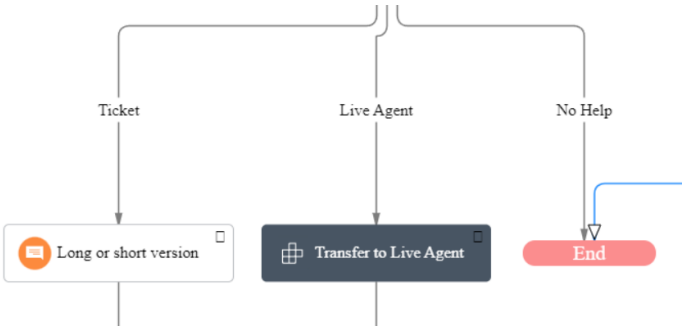


4 Topics I created

4.1 Create incident with Description

This is a 'Topic Block'. It means that we can use this topic in different topics.

The purpose of this topic block is ending the topic. It will give the choice to create a ticket, to request live support or to end without asking for help. This will be used at the end of almost every other topic.



4.1.1 End topic without doing anything

The chatbot will end the conversation without doing anything.

4.1.2 Transfer to Live Agent

This will let the user chat with a live agent (Human). To create a live agent, the user needs to be added to certain groups. The groups are:

- Live Chat Agent
- Service Desk Agent

Here you can see a demo of this topic: https://youtu.be/uP9sxwXGxho

4.1.3 Create Ticket

This will let the user create a ticket.
It will first ask if the user wants to give a detailed description of the problem. If he does so, the ticket will get priority over tickets that didn't do it. The detailed questions are:

- Give a description? (Description)
- On what device it the problem? (Configuration item)
- When did it start?
- Did anything change since the beginning? (Change info)
- Do you want to upload a file? (Choose file)

The topic makes sure the user can't make any mistake.

- The "Description" and the "Change info" block are plain text
- The "select configuration Item" let you only pick a device connected to your account. If your account doesn't have a device it won't add anything.
- The "When did it start" block lets you pick from a calendar and a clock. So the format is always the same

4.2 File Translation

This uses the Deepl script from Script Include.

Because this is a new topic I also needed to create some NLU's or Natural Language Understanding. With NLU the chatbot can detect what topic the user wants without using a structured sentence but just human language.

4.2.1 Translate Plane Text

The chatbot request the text and to which language you want to translate. It detects the original text by itself.

It uses the DeeplAPI script

```
(function execute() {

    var singleOutMsg = new sn_cs.SinglePartOutMsg();
    var target_lang = vaInputs.choose_language;

    // Set type to string to use in URL
    toTranslate = vaInputs.translate_this_text.toString();
    var TranslatedText = DeeplAPI(toTranslate, target_lang);

    // Give output
    singleOutMsg.setTextPart(TranslatedText.toString());
    gs.log('link: ' + TranslatedText.toString());
    return singleOutMsg;

})()
```

4.2.2 Translate Files

The chatbot request to upload a text file and to which language you want to translate.

It uses the GetContentOfFile script to send the file to the Deepl servers and it uses the UploadFileToAttachment script to upload the translated file to the ServiceNow Attachment table

```
(function execute() {
    // var singleOutMsg = new sn_cs.SinglePartOutMsg();
    var target_lang = vaInputs.choose_language;
    var translateDoc = GetContentOfFile(target_lang);

    //singleOutMsg.setTextPart(translateDoc);

    // Save file to attachment table
    var saveFile = UploadFileToAttachment(translateDoc);
})()
```

I use a small peace of code to get the link of the translated file from the ServiceNow Attachment table:

```
(function execute() {
    // Get the link for the most recent added attachment created by
the current user
    var translatedSa = new GlideRecord('sys_attachment');
    var singleOutMsg = new sn_cs.SinglePartOutMsg();

    // Get the most recent added attachment created by the current
user
    translatedSa.addQuery('sys_created_by', gs.getUserName());
    translatedSa.orderByDesc('sys_created_on');
    translatedSa.query();

    if (translatedSa.next()) {
        // Get info from the Attachment
        var translatedFileName = translatedSa.getValue('file_name');
        var translatedFileSysID = translatedSa.getValue('sys_id');
        gs.log('translated file name ' + translatedFileName );

        attachmentLink =
'https://####/nav_to.do?uri=sys_attachment.do?sys_id=' +
translatedFileSysID;

        gs.log('Link ' + attachmentLink );
        singleOutMsg.setTextPart(attachmentLink);
        return singleOutMsg ;
    }
})()
```

5 Research

5.1 ServiceNow

ServiceNow has it's own learning platform and an OK documentation. The documentation is OK because it is hard to look for the right version of ServiceNow.

5.1.1 NLU

A very interesting topic is NLU or Natural Language Understanding. This makes it possible for the chatbot to gives the topics a user requested without using a structured sentence, but just in human language.

Most of the NLU are Out Of The Box ,so I didn't have to modify them a lot. I still had to learn the basics of NLU for the little bits I had to modify. It uses the following things to identify the right topic

- **Intent**
  - Something a user wants to do or what you want your application to handle, such as granting access.
- **Utterance**
  - The string the user will provide to the chatbot. The chatbot will compare the given string to the pre-defined Utterances
- **Entity**
  - The object of, or context for, an action. For example: a laptop, a user role, or a priority level.

5.2 Deepl API

The Deepl API has a very useful documentation. I used this together with Postman to understand the API, the request and the response.

5.3 My mentor

If I had a problem I could not fix, I would ask my mentor for help. He would give me a link he thought would be useful. Most of the times it was a useful link.

5.4 ChatGPT

I used ChatGPT to get a start in the project. It never gave the solution right away. I always had to look further and modify it. Now, I understand that an 'AI browser' is not the solution for everything. At least not for now.

6 Conclusion

Although this project doesn't 100% fit in my education, I learned a lot of useful things of it. This internship showed me how it is to work in an IT department for a worldwide company.

I did improve my Javascript skills and learned a lot about AI and the process behind IT support.