Creating a web service for generating branded Word documents using HTML templates and JSON data sources

## Expected skills

* ASP.NET Core technologies
* familiarity with Serilog logging
* experience with the OpenXML SDK
* familiarity with Docker containers
* being fluent with git and gitflow is a must

Being familiar with the Atlassian Confluence API is an advantage but not a prerequisite.

# Setting up the new service

## Description

Create a new .NET Core based endpoint. Prepare the solution to use

* Docker host
* SEQ logger
* MassTransit subscription
* OpenXML SDK

## Acceptance criteria

The new service can be deployed and executed.

# Implement Confluence template download feature

## Description

The sections of the final Word document shall use Confluence pages as templates.

* Connect to Confluence (authentication with API keys)
* Download page JSON (use correct URL)
* Page ID must be used as input parameter

The Confluence API Documentation can be found here:  
<https://docs.atlassian.com/atlassian-confluence/REST/6.6.0/>

## Acceptance criteria

The HTML is downloaded.

# Parse the Confluence template HTML

## Description

The downloaded HTML string must be parsed. The following elements must be identified:

* Headers (<h1>, <h2>, <h3>, <h4>, <h5>, <h6>)
* Lists (<ul>, <ol> → <li>)
* Tables (<table>, <theader>, <tfooter>, <tbody>, <tr>, <th>, <td>)
* Paragraphs (not just <p>! in tables, lists and any other inline HTML tags, there is not necessary a <p> tag)
* <code> tags as expressions

Using an HTML parser NuGet package would help with this task a lot.

## Acceptance criteria

HTML is parsed and <code> expressions are found.

# Create a Word document using the parsed template

## Description

As a prototype, create a Word document from the parsed HTML.

**Not in scope:** Expression evaluation (conditions, variable replacement)

The Microsoft OpenXML Sdk Documentation can be found here:  
<https://docs.microsoft.com/en-us/office/open-xml/getting-started>

## Acceptance criteria

A simple Word document is created. Locally, not for final use.

# Integrate Word document generator and JSON data sources

## Description

Using the Word template created in the previous step and the variables extracted from the JSON input, replace variables and evaluate expressions in <code> tags.

You can use either stack-based RPN calculator class or expression tree for this purpose.

**Not in scope:** Handling conditions. (remove table rows, empty tables, list items, empty lists)

## Acceptance criteria

Variables are replaced.

# Handle inlined conditions (table rows, list items)

## Description

Where there are variable names or expressions (between <code> tags) at the beginning of a page, paragraph, table header, row header, etc., these expressions are *conditions* that determine whether or not the particular parts should appear in the document. Conditions must be evaluated.

Based on the *conditions*

* remove table rows
* remove list items
* remove empty tables
* remove empty lists (since the wrapping tags will remain for empty lists, these should not be included in the final document)

## Acceptance criteria

Conditions are handled correctly, sections are removed.

# Add title (cover) page function

## Description

Implement a title page creator function.

* Title
* Subtitle

## Acceptance criteria

Title page is created in the final Word document.

# Add page headers/footers

## Description

Page footers and headers are added.

Company logo can be added to the header.

## Acceptance criteria

Footers and headers appear in the document pages.

# Add table of contents

## Description

Add a table of contents (field!) to the Word document.

## Acceptance criteria

ToC field is added, ToC shows up in the Word document.