

Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki

Marcin Krocak

Konwerter formatów HTML/MD/SQL/CSV

Projekt zespołowy
na studiach stacjonarnych
o kierunku Informatyka

Kielce, 2021

Spis treści

1.	Wstęp teoretyczny	3
1.1	Różnice pomiędzy formatami HTML a MD	3
1.2	Formaty SQL i CSV	4
1.2	Serializacja i parsowanie	4
2.	Założenia projektu	5
2.1	Zakres projektu	5
2.2	Architektura	6
3.	Scenariusze użycia	6
3.1	Diagram przypadków użycia	7
4.	Implementacja	8
4.1	Technologie po stronie serwera	8
4.2	Technologie po stronie klienta	8
4.3	Schematy blokowe algorytmów konwersji	8
4.4	Omówienie kluczowych fragmentów implementacji	11
4.5	Obsługa plików w programie	13
5.	Uruchomienie programu	14
5.1	Konwersja pomiędzy formatami HTML i MD	15
5.2	Konwersja z formatu HTML na formaty SQL i CSV	18

1. Wstęp teoretyczny

Konwertery formatów plików są narzędziami oferującymi szeroki zakres konwersji. Obejmują m.in. wideo, audio, obrazy i dokumenty. Dostępne są w wielu formach. Aktualnie najpopularniejsze są wersje przeglądarkowe. Przykładami są narzędzia takie jak np. Convertio, smallpdf, lub 2conv. Ich wspólną cechą jest obsługa szerokiej gammy formatów. Konwersja odbywa się poprzez wgranie pliku po stronie klienta. Plik kierowany jest do serwera. Następuje tam konwersja i przesłanie z powrotem do klienta. Przesłane dane są bezpieczne i usuwane są zaraz po przeprowadzeniu konwersji. Zaletą jest obsługa wszystkich typów urządzeń, oraz szybkość mieszcząca się w przedziale kilku sekund, lub – w przypadku większych plików – kilku minut.

1.1 Różnice pomiędzy formatami HTML a MD

Markdown jest językiem znaczników przeznaczonym do formatowania tekstu. Format użyteczny jest m.in. celem lepszej jakości prezentacji projektu na platformie typu GitHub. Główne cechy:

- Formatowanie akapitów - Nowy akapit tworzony jest po przejściu do nowej linii w edytorze, bądź zostawieniu wolnej linii po zastosowaniu znacznika formatującego tekst (na przykład punkt na liście).
- Nagłówki - tworzone są przy użyciu znaku #. Stanowi to odpowiednik dla znacznika h w kodzie HTML.
- Przerwanie linii - odpowiednikiem znacznika br w HTML jest umieszczenie dwóch spacji na końcu linii.
- Pogrubienie tekstu - Wykonywane za pomocą znaków ** przed ciągiem znaków i na jego końcu. Przykład: pogrubiony tekst
- Cytat - Umieszczony za pomocą znaku *.
- Listy - Są tworzone, w zależności od potrzeb, jako lista numerowana (wtedy wystarczy wykonać numerowaną listę jak w edytorze tekstu), lub punktowana (należy wstawić znak * przed ciągiem znaków, wykorzystać można też znaki -, +).
- Bloki kodu - Umieszczane poprzez umieszczenie czterech spacji lub jednego tabulatora. W listach wykorzystać trzeba dwa tabulatory.
- Zdjęcia - tworzone za pomocą konstrukcji ![alternatywny zapis] (url)
- Odnośniki - tworzone za pomocą konstrukcji [tekst przy odnośniku] (url).

1.2 Formaty SQL i CSV

Plik wartości rozdzielanych przecinkami (CSV) to zwykły plik tekstowy zawierający listę danych. Pliki te są często używane do wymiany danych między różnymi aplikacjami. Na przykład bazy danych i menedżerowie kontaktów często obsługują pliki CSV. Pliki te mogą być czasami nazywane wartościami separowanymi znakami lub plikami rozdzielanymi przecinkami. Najczęściej używają one znaku przecinka do rozdzielania (lub ograniczania) danych, ale czasami używają innych znaków, takich jak średniki. Chodzi o to, że można eksportować złożone dane z jednej aplikacji do pliku CSV, a następnie zaimportować dane w tym pliku CSV do innej aplikacji.

SQL to strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych. Język SQL jest językiem deklaratywnym. Przykładowymi instrukcjami są m.in. `CREATE TABLE` – tworzenie tabeli w bazie danych, `INSERT` – umieszczanie nowego wiersza w tabeli.

1.2 Serializacja i parsowanie

Serializacja - W serializacji zamienia się strukturę obiektową na dane testowe. Wówczas pewna hierarchiczna struktura danych lub obiektów przekształcana jest do postaci szeregowej, czyli formatu, który może być przechowywany, z zachowaniem aktualnego stanu obiektów. W przypadku webu najpopularniejszym formatem dla serializacji jest JSON (dawniej prym wiódł XML). Taka zserializowana postać obiektu może być przechowywana, np. w buforze pliku lub pamięci, przesyłana do innego procesu lub innego komputera poprzez Sieć, i w późniejszym czasie odtwarzana w tym samym lub innym środowisku komputerowym. Kiedy otrzymany szereg bitów jest odczytywany zgodnie z formatem serializacji, to może być wykorzystywany do utworzenia semantycznie identycznych klonów dla oryginalnych obiektów. Dla wielu złożonych obiektów, np. takich które w szerokim zakresie używają referencji, proces ten nie należy do najłatwiejszych. Serializacja obiektów zorientowanych obiektowo nie zawiera żadnych związanych z nimi metod, z którymi wcześniej były ze sobą nierozzerwalnie powiązane. Procesem stricte odwrotnym do serializacji jest deserializacja. Polega ona na odczycie wcześniej zapisanego strumienia danych i odtworzeniu na tej podstawie obiektu wraz z jego stanem bezpośrednio sprzed serializacji.

Parsowanie - Parsowanie polega na zamianie danych wejściowych (test) na odpowiadającą im strukturę obiektową (wyjście). Dokonuje się wówczas analizy testu celem ustalenia pewnej struktury i zgodności ze składnią języka. Zastosowanie parserów jest silnie uzależnione od samego wejścia (danych wejściowych). W przypadku języków danych (np.

znacznikowego HTML lub XML) parser najczęściej występuje jako program ułatwiający czytanie plików. W przypadku języków programowania parser występuje jako komponent dla kompilatora lub interpretera. Analizuje on kod źródłowy w postaci jakiegoś języka programowania w celu utworzenia wewnętrznej reprezentacji.

2. Założenia projektu

2.1 Zakres projektu

Projekt obejmuje 4 formaty plików. Są to: HTML, MD, SQL, CSV. Pomiędzy dwoma pierwszymi formatami konwersja przebiega obustronnie. W przypadku SQL i CSV możliwa jest konwersja z formatu HTML. Z dokumentu wyodrębniane są elementy składające dane (tabele). Zamieniane są one na instrukcje CREATE TABLE i INSERT dla formatu SQL lub na wiersze i kolumny oddzielone średnikami dla formatu CSV. Z powodu złożoności formatu HTML wprowadzono ograniczenia w odniesieniu do ilości możliwych do skonwertowania tagów.

Lista obsługiwanych elementów przy konwersji z HTML na MD:

- H1, H3 – nagłówki.
- P – nowy paragraf
- BR – przejście do nowej linii
- STRONG – tekst pogrubiony
- EM – tekst pochylony
- BLOCKQUOTE – cytat blokowy
- OL – lista numerowana
- UL – lista punktowana
- LI – element listy numerowanej/punktowanej
- IMG – zdjęcie
- A – odnośnik
- CODE – blok kodu

Elementy nieobsługiwane przy konwersji:

- TABLE – tabela
- S – przekreślenie tekstu

Pozostałe elementy składni MD takie jak: Lista zadań, lista definicji, notatka nie zostały przeniesione z HTML, ponieważ nie mają tam swojego odpowiednika w tagach.

Lista obsługiwanych elementów przy konwersji z MD na HTML:

- Nagłówki
- Nowy paragraf
- Blok kodu
- Cytaty blokowe
- Tekst pogrubiony
- Tekst pochylony
- Lista numerowana/punktowana
- Zdjęcie/odnośnik

Brakujące elementy przy konwersji:

- Tabela
- Przekreślenie tekstu

Lista uwzględnionych elementów przy konwersji z HTML na SQL i CSV

- <table>
- <tr>
- <td>

Elementy pominięte przy konwersji

- Złączenia w tabelach (scalanie komórek)

2.2 Architektura

Aplikacja stworzona będzie w architekturze klient-serwer. W interfejsie użytkownika wgrywany będzie plik i wybierany będzie format na jaki ma być skonwertowany. Na serwerze realizowane będzie przechowywanie i obsługa plików oraz sama konwersja.

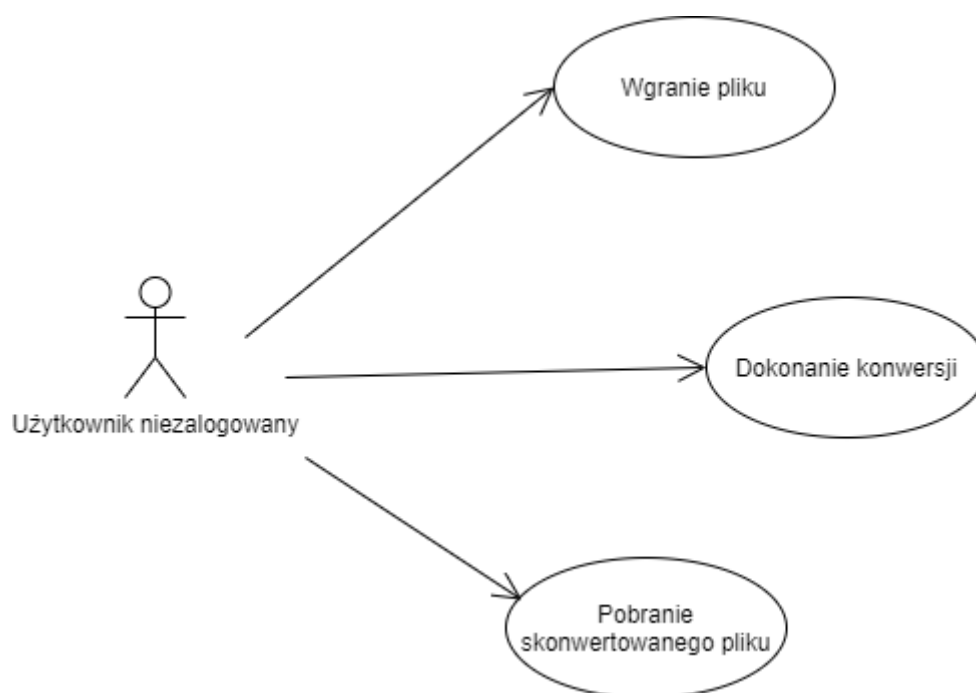
3. Scenariusze użycia

Nazwa przypadku użycia	Wgranie pliku
Aktor	Użytkownik niezalogowany
Warunki początkowe	BRAK
Warunki końcowe	Plik wgrany na serwerze
Scenariusz sukcesu	<ol style="list-style-type: none">1. Naciśnięcie przycisku “Wybierz plik”2. *Naciśnięcie przycisku “Upload”3. Plik zostaje zapisany na serwerze

Rozszerzenia	2.* Plik jest za duży, lub jest w złym formacie 2.1 Wyświetlenie okna dialogowego z komunikatem błędu 2.2 Powrót do kroku 1
--------------	---

Nazwa przypadku użycia	Dokonanie konwersji
Aktor	Użytkownik niezalogowany
Warunki początkowe	BRAK
Warunki końcowe	Otrzymanie skonwertowanego pliku
Scenariusz sukcesu	1. Wybór formatu na który ma być skonwertowany plik 2. *Naciśnięcie przycisku „Konwertuj” 3. Plik zostaje skonwertowany po stronie serwera i odesłany do klienta.
Rozszerzenia	2. * Wybrano format który nie jest obsługiwany 2.1 Wyświetlenie komunikatu 2.2 Powrót do kroku 1.

3.1 Diagram przypadków użycia



4. Implementacja

4.1 Technologie po stronie serwera

Po stronie serwera wykorzystany został język Java. Szkielet aplikacji zaprojektowany został przy pomocy Spring Boot. Celem wykonania konwersji przydatne okazały się biblioteki odpowiedzialne za parsowanie dokumentów HTML i MD:

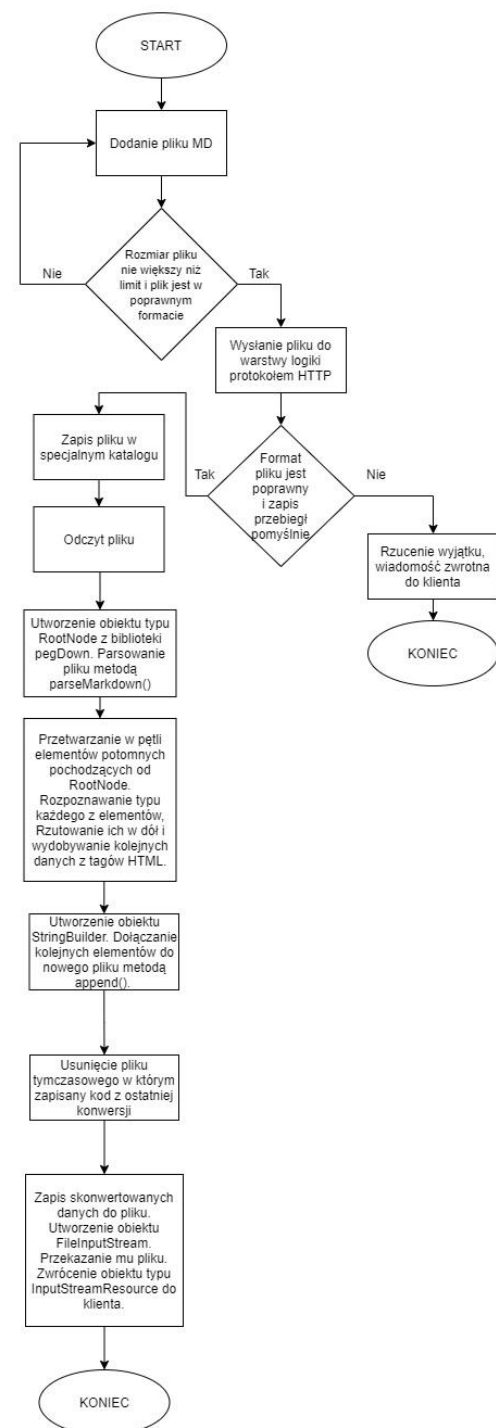
- JSOUP - Jsoup przede wszystkim korzysta z obiektów typu Document. Reprezentuje on drzewo dokumentu i umożliwia dostanie się do poszczególnych węzłów kodu źródłowego HTML. Oznacza to dostęp do drzewa DOM (Document Object Model) jak w przeglądarce internetowej.
- Pegdown – jest biblioteką służącą do przetwarzania plików Markdown. Umożliwia parsowanie. Dzięki temu łatwiejszy jest dostęp do poszczególnych elementów dokumentu. Tworzona jest wówczas kolekcja elementów (Node). Elementy mogą być przetworzone na odpowiadające im tagi HTML lub składnie innego języka.

4.2 Technologie po stronie klienta

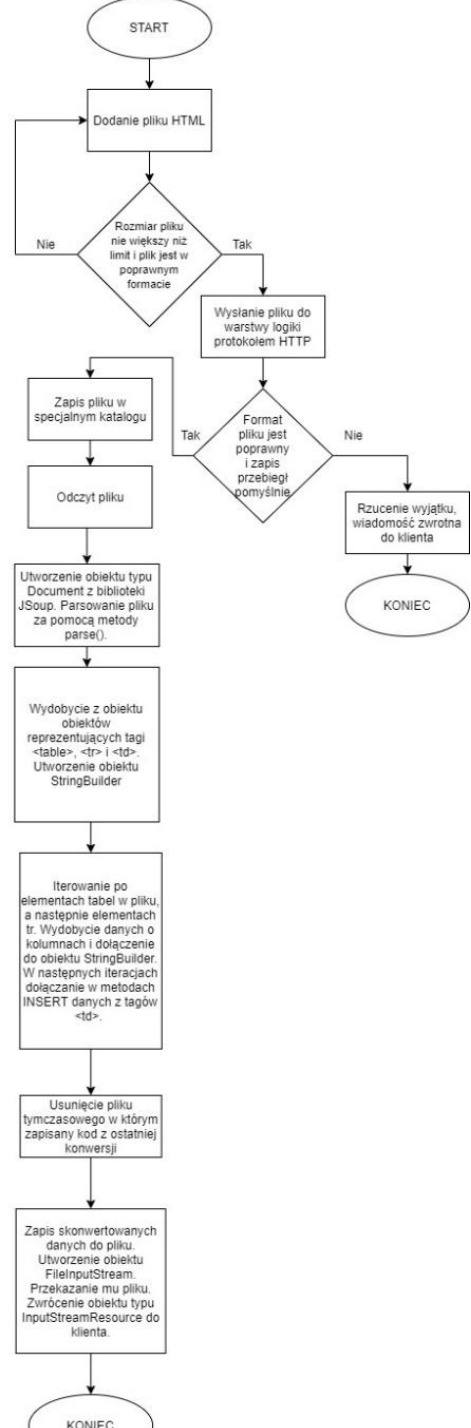
Po stronie klienta wykorzystana została technologia JavaScript i framework Angular.

4.3 Schematy blokowe algorytmów konwersji

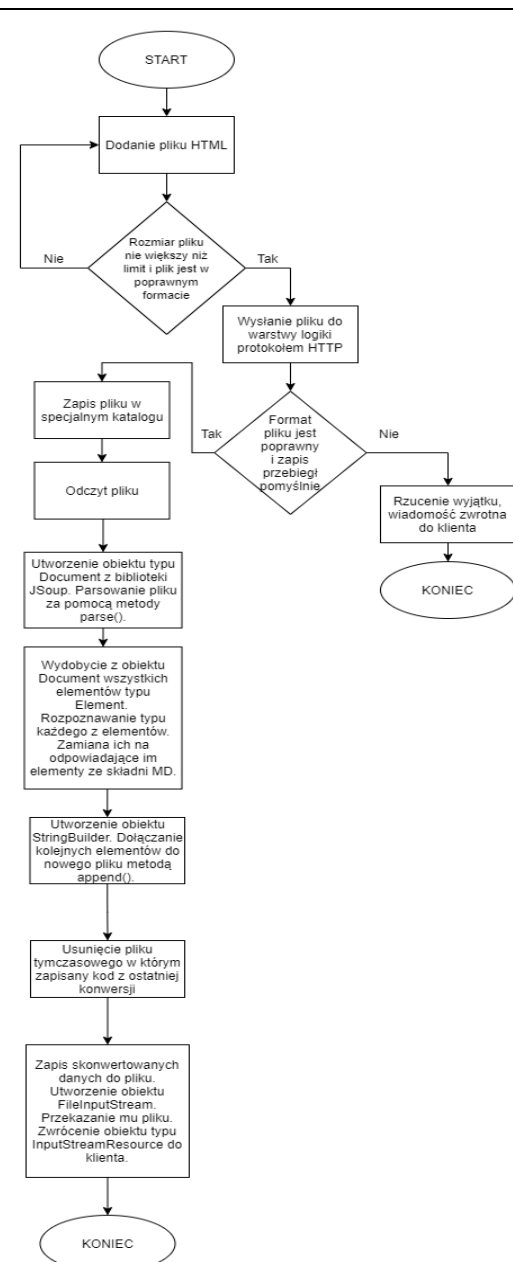
Schemat blokowy dla konwersji pomiędzy formatem MD do HTML.



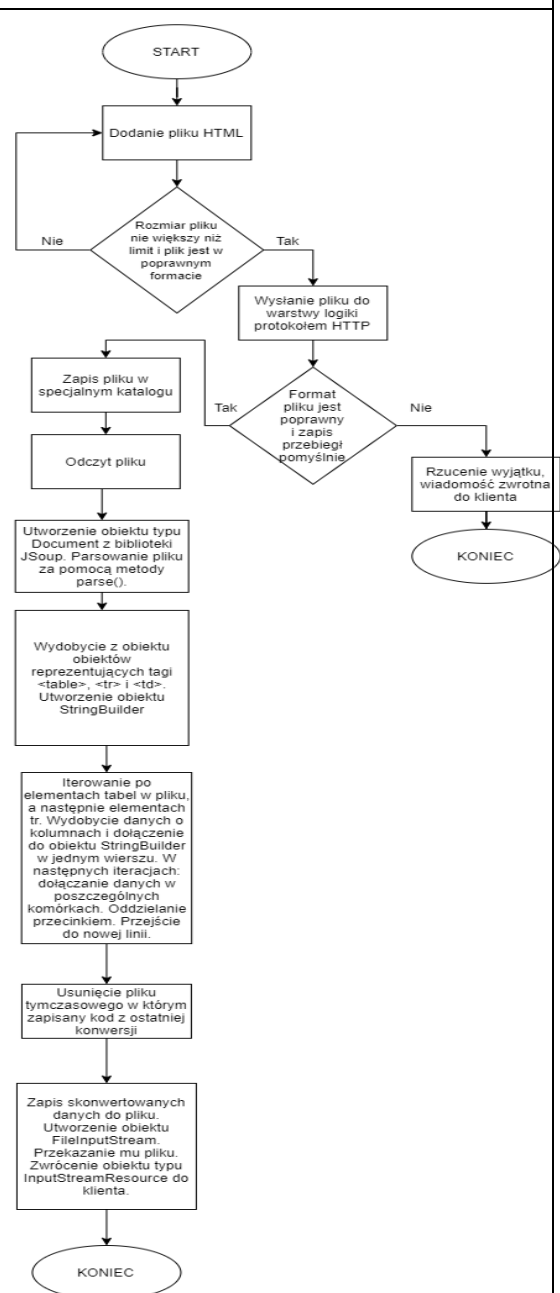
Schemat blokowy dla konwersji pomiędzy formatem HTML do SQL



Schemat blokowy dla konwersji pomiędzy formatem HTML do MD.



Schemat blokowy dla konwersji pomiędzy formatem HTML do CSV.



4.4 Omówienie kluczowych fragmentów implementacji

- **Konwersja z formatu HTML**

Format HTML może zostać przeniesiony na formaty MD, SQL, lub CSV. W każdym z przypadków najpierw usuwane są stare pliki tymczasowe. Następnie dokonuje się parsowania za pomocą biblioteki Jsoup.

```
@Override
public ResponseEntity<InputStreamResource> convertHtmlToMd() throws IOException {

    Document doc = null;
    //delete old files
    try {
        File file = new File("uploadir/test.html");
        doc = Jsoup.parse(file, "UTF-8");
        File fileUp = new File("src/main/resources/convertedfiles/result.md");

        if(fileUp.delete())
        {
            System.out.println("File deleted successfully");
        }
    }
```

Następnie dokonuje się konwersji. Tworzony jest obiekt typu StringBuilder. W przypadku konwersji na MD pobierane są wszystkie elementy.

```
StringBuilder content = new StringBuilder();
String tempTag = "";

Elements elements = doc.body().select("*");
for(Element element : elements) {
    //System.out.print(element.tagName());
    tempTag = element.tagName();
    switch (tempTag) {
        case "h1" :
            content.append("# ").append(element.ownText()).append("\n");
            break;
        case "h3" :
```

Następnie odbywa się zapis do pliku wynikowego. Celem tego tworzony jest obiekt typu FileWriter, InputStreamResource i FileInputStream. Po dokonaniu zapisu plik wysyłany jest do klienta i dokonywany jest zapis pliku do historii.

```
FileWriter fileWriter = null;
InputStreamResource resource = null;
FileInputStream fis = null;
File myObj = null;
try {
    myObj = new File("src/main/resources/convertedfiles/result.md");
    if (myObj.createNewFile()) {
        System.out.println("File created: " + myObj.getName());
    }
}
```

```
filesService.pushFileToHistory(myObj, "md");
return ResponseEntity.ok()
    .contentTypeLength(myObj.length())
    .contentType(MediaType.parseMediaType("text/html"))
    .body(resource);
```

W przypadku konwersji na formaty CSV i SQL brane są pod uwagę elementy związane z tabelami. Następnie są one sprawdzane i na ich podstawie budowane są np. polecenia SQL.

```

if(tables.size() > 0) {
    for(Element table : tables) {
        content.append("CREATE TABLE test_table").append(tableNr).append("\n");
        for(Element tr : trs) {
            if(temp==0) {
                for(int i=0; i< tr.childrenSize(); i++) {
                    columns.add(tr.child(i).text());
                    content.append(columns.get(i));
                    content.append(" VARCHAR(100)");
                    if(!(i == tr.childrenSize()-1)) content.append(",\n");
                    else content.append("\n");
                }
                content.append(");\n\n");
                temp++;
            }
            else {
                //INSERTS
                content.append("INSERT INTO test_table").append(tableNr).append(" VALUES(");
                for(int i=0; i< tr.childrenSize(); i++) {
                    content.append(tr.child(i).text());
                    if(!(i == tr.childrenSize()-1)) content.append(", ");
                }
                content.append(");\n");
            }
        }
        temp=0;
        tableNr++;
    }
}

```

- **Konwersja z formatu MD do HTML**

W konwersji z formatu MD używa się biblioteki Pegdown. Pobierany jest plik tymczasowy, inicjowany jest procesor i odczytywane są poszczególne elementy składni MD. Zapisane są w liście typów Node. W pętli są sprawdzane i do obiektu StringBuilder dopisywane są kolejne linijki skonwertowanego kodu.

```

@Override
public ResponseEntity<InputStreamResource> convertMdToHtml() throws IOException {
    File mdFile = new File("uploaddir/test.md");
    PegDownProcessor processor = new PegDownProcessor(Extensions.ALL);
    char[] markdown = FileUtils.readAllChars(mdFile);
    Preconditions.checkNotNull(markdown, "The specified file isn't found - "+
mdFile.getName());
    RootNode rootNode = processor.parseMarkdown(markdown);

    List<Node> nodes = rootNode.getChildren();
    StringBuilder content = new StringBuilder();

    String temp1;
    String temp2;
    for (Node node : nodes) {
        //System.out.print(node);
        if (node instanceof HeaderNode) {
            HeaderNode headerNode = (HeaderNode) node;
            temp1 = "<h" + headerNode.getLevel() + ">";
            temp2 = "</h" + headerNode.getLevel() + ">";

```

Na końcu obiekt StringBuilder zapisywany jest w pliku i przekazywany analogicznie jak w konwersji z formatu HTML.

4.5 Obsługa plików w programie

- Pliki tymczasowe

Obsługa plików rozpoczyna się inicjacją. Po uruchomieniu serwera inicjowany jest katalog w którym przechowywane będą tymczasowe pliki i pliki historii. Następnie pliki zapisywane są do folderu „uploadir”. W poniższym fragmencie kodu ukazana jest procedura zapisu plików. Przesłane pliki mają typ MultipartFile. Rozpoznawane jest ich rozszerzenie (html, lub md), a następnie dokonywany jest zapis pod tymczasową nazwą.

```
@Override
public void saveFile(MultipartFile multipartFile) {
    if(multipartFile.getContentType().equals("text/html")) {
        try {
            Files.copy(multipartFile.getInputStream(), this.root.resolve("test.html"),
StandardCopyOption.REPLACE_EXISTING);
        } catch (Exception ex) {
            throw new RuntimeException("Bład przy zapisie pliku: " + ex.getMessage());
        }
    }
    else {
        try {
            Files.copy(multipartFile.getInputStream(), this.root.resolve("test.md"),
StandardCopyOption.REPLACE_EXISTING);
        } catch (Exception ex) {
            throw new RuntimeException("Bład przy zapisie pliku: " + ex.getMessage());
        }
    }
}
```

Po zapisaniu pliki mogą być odczytywane celem konwersji lub listowaniu ich u klienta. Mogą być też usunięte przez niego. Po zakończeniu pracy serwera są one automatycznie usuwane.

- Pliki historii

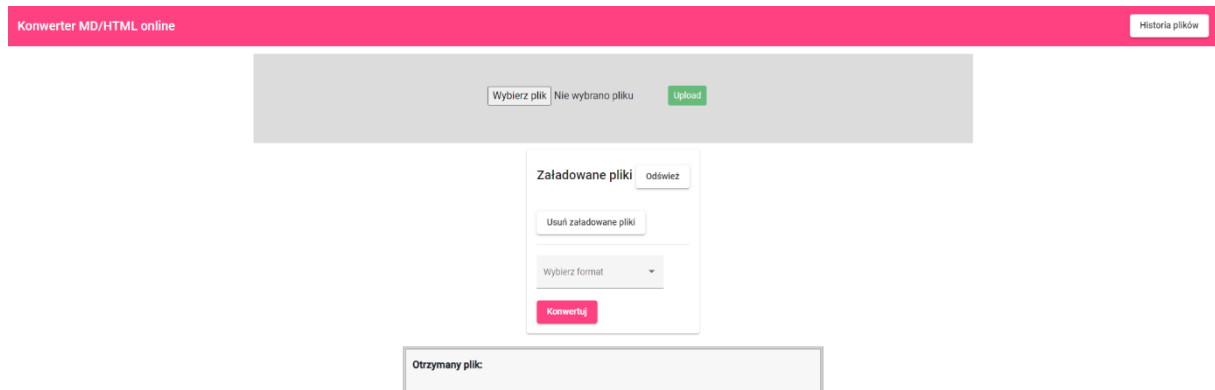
W katalogu historii umieszczane są pliki poddane już konwersji. Można je w każdej chwili pobrać po stronie klienta. Poniżej zamieszczony jest fragment kodu prezentujący pobieranie właściwości wszystkich plików. Tworzony jest w tym celu obiekt DTO w którym znajdują się informacje o nazwie pliku, dacie modyfikacji i rozmiarze. W metodzie getFilesFromHistory tworzona jest lista tych właściwości, a następnie w pętli tworzone są instancje obiektu przy użyciu buildera. Lista przesyłana jest do klienta.

```
public ResponseEntity<List<FilesHistoryDto>> getFilesFromHistory() {
    List<FilesHistoryDto> fh = new ArrayList<FilesHistoryDto>();
    String date;
    File dir = new File(historyRoot.toUri());
    File[] list = dir.listFiles();
    System.out.println("path " + dir.getName());
    for (int i = 0; i < list.length; i++) {
        if (list[i].isFile()) {
            System.out.println("File " + list[i].getName());
            date = convertTime(list[i].lastModified());
            fh.add(FilesHistoryDto.builder().size(String.valueOf(list[i].length())).name(list[i].getName())
                .dateOfConversion(date).build());
        } else if (list[i].isDirectory()) {
            System.out.println("Directory " + list[i].getName());
        }
    }
    return ResponseEntity.ok(fh);
}
```

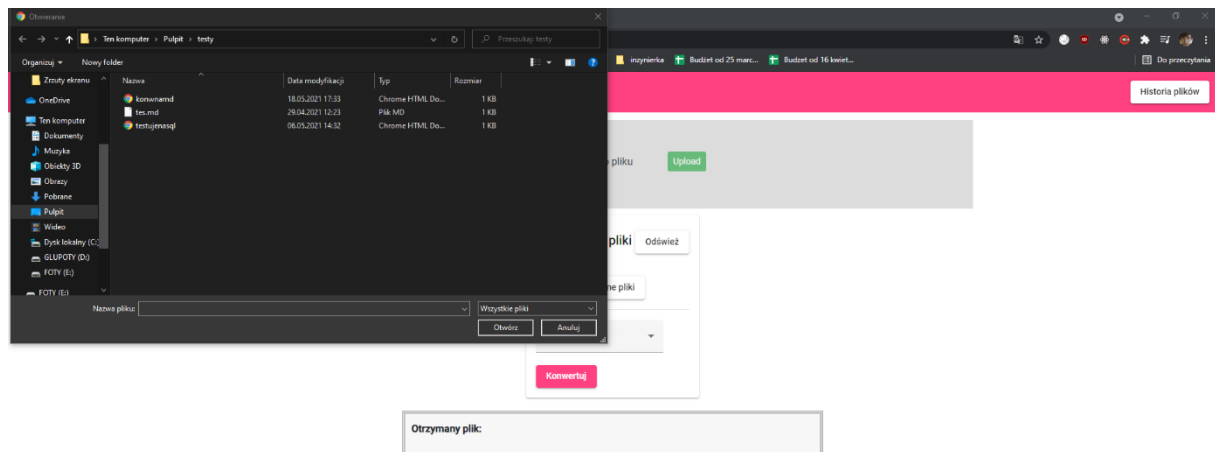
Nazwy plików przy tworzeniu są generowane automatycznie. Tworzą schemat: „converted” + numer. Po wyłączeniu serwera pliki są automatycznie usuwane. Możliwe jest ich pobieranie poprzez podanie przez klienta nazwy pliku. Po każdej konwersji plik jest automatycznie przenoszony do historii.

5. Uruchomienie programu

Po uruchomieniu projektu pojawia się interfejs użytkownika



Istnieje możliwość wgrania pliku.



5.1 Konwersja pomiędzy formatami HTML i MD

Konwersja HTML na MD. Wgrywany jest plik.

```
1  An introduction sentence. Another introduction sentence.
2  An introduction sentence.
3  <h1>First header title h1</h1>
4  <h3>h3 header</h3>
5  Some content. Some content.
6  <ul><li>List item 1: some description</li><li>List item 2: some description</li></ul>
7  Some content. Some content.
8  <code>SomeClass clazz = new SomeClass();
9  clazz.test();
10 </code>
11 Some content.
12 <strong>test text</strong>
13 <em>test text</em>
14 <blockquote>sample blockquote</blockquote>
15 <a href="https://www.example.com">link ex</a>
16 </img>
17
18
```

Wybierz plik konwnamd.html

Upload

Przesłano plik!

Załadowane pliki

Odśwież

test.html

Usuń załadowane pliki

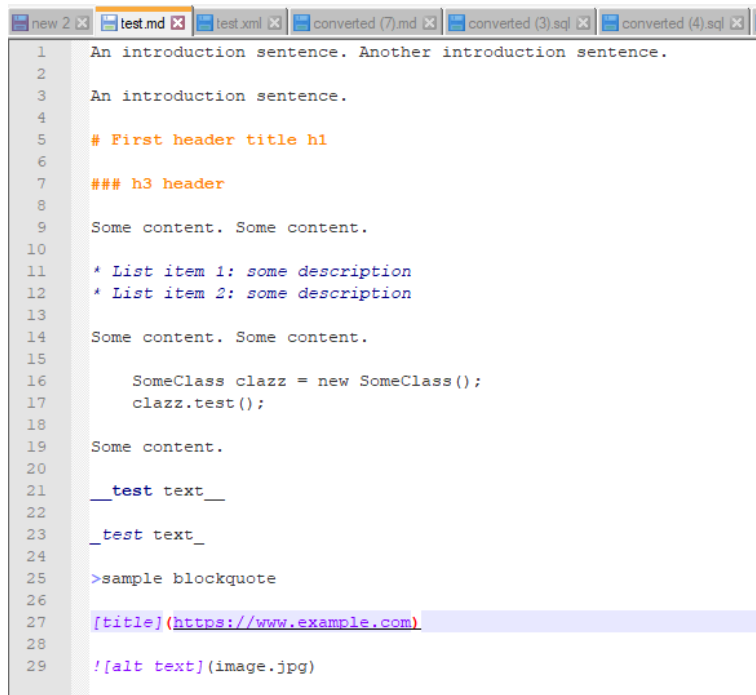
Wybierz format

Konwertuj

Po konwersji i pobraniu pliku widoczny jest wynik

```
1  An introduction sentence. Another introduction sentence. An introduction sentence. Some content. Some content. Some content. Some content.
2  # First header title h1
3  ### h3 header
4  - List item 1: some description
5  - List item 2: some description
6  SomeClass clazz = new SomeClass(); clazz.test();
7  **test text**
8  *test text*
9  > sample blockquote
10 [link ex](https://www.example.com)
11 ![image](image.jpg)
12
```

Konwersja MD na HTML. Analogicznie wgrany jest plik, o rozszerzeniu MD. Następnie zostaje skonwertowany i pobrany



```
1 An introduction sentence. Another introduction sentence.
2
3 An introduction sentence.
4
5 # First header title h1
6
7 ### h3 header
8
9 Some content. Some content.
10
11 * List item 1: some description
12 * List item 2: some description
13
14 Some content. Some content.
15
16     SomeClass clazz = new SomeClass();
17     clazz.test();
18
19 Some content.
20
21 __test text__
22
23 _test text_
24
25 >sample blockquote
26
27 [title](https://www.example.com)
28
29 ![alt text](image.jpg)
```

Wybierz plik

test.md

Upload

Przesłano plik!

Załadowane pliki

Odśwież

test.md

Usuń załadowane pliki

Wybierz format

HTML

Konwertuj

Otrzymany plik:

```
An introduction sentence. Another introduction sentence. An introduction sentence. <h1>First
header title h1</h1> <h3>h3 header</h3> Some content. Some content. <ul><li>List item 1: some
description</li><li>List item 2: some description</li></ul>Some content. Some content.
<code>SomeClass clazz = new SomeClass(); clazz.test(); </code> Some content. <strong>test
text</strong> <em>test text</em> <blockquote>sample blockquote</blockquote> link<a
href="https://www.example.com">link</a> </img>
```


Talia1 test.html converted (8).md converted (6).sql converted (5).csv converted (9).md converted (42).html

1

An introduction sentence. Another introduction sentence.

2

3

An introduction sentence.

4

5

<h1>First header title h1</h1>

6

7

<h3>h3 header</h3>

8

9

Some content. Some content.

10

11

List item 1: some descriptionList item 2: some descriptionSome content. Some content.

12

13

<code>SomeClass clazz = new SomeClass();

14

clazz.test();

15

</code>

16

17

Some content.

18

19

test text

20

21

test text

22

23

<blockquote>sample blockquote</blockquote>

24

25

linklink

26

27

28

29

5.2 Konwersja z formatu HTML na formaty SQL i CSV

Tworzony jest plik testowy o rozszerzeniu HTML. Zawarta jest w nim tabela z danymi

```
1 <html>
2 <head>
3   <title>test</title>
4 </head>
5 <body>
6   <h1>test</h1>
7   <p>another p test</p>
8   <p>and another</p>
9   <table>
10    <tr>
11      <td>pin</td>
12      <td>time</td>
13    </tr>
14    <tr>
15      <td>0001</td>
16      <td>1524810082</td>
17    </tr>
18    <tr>
19      <td>0002</td>
20      <td>1524810082</td>
21    </tr>
22    <tr>
23      <td>0003</td>
24      <td>1524810082</td>
25    </tr>
26    <tr>
27      <td>0004</td>
28      <td>1524810082</td>
29    </tr>
30  </table>
31 </body>
32 </html>
```

Konwersja na format CSV. Plik wgrywany jest przez klienta na serwer. Plik jest konwertowany i odsyłany. Jest automatycznie pobierany po naciśnięciu przycisku „Konwertuj”.

Wybierz plik

testujenasql.html

Upload

Przesłano plik!

Załadowane pliki

Odśwież

Usuń załadowane pliki

Wybierz format

CSV

Konwertuj

Otrzymany plik:
pin,time 0001,1524810082 0002,1524810082 0003,1524810082 0004,1524810082

Plik po konwersji:

```
1 pin,time
2 0001,1524810082
3 0002,1524810082
4 0003,1524810082
5 0004,1524810082
6
```

Konwersja na format SQL. Analogicznie wgrywany jest na serwer ten sam plik.

Wybierz plik

testujenasql.html

Upload

Przesłano plik!

Załadowane pliki

Odśwież

test.html

Usuń załadowane pliki

Wybierz format

SQL

Konwertuj

Otrzymany plik:

```
CREATE TABLE test_table0( pin VARCHAR(100), time VARCHAR(100) ); INSERT INTO test_table0
VALUES(0001, 1524810082); INSERT INTO test_table0 VALUES(0002, 1524810082); INSERT INTO
test_table0 VALUES(0003, 1524810082); INSERT INTO test_table0 VALUES(0004, 1524810082);
```

Po konwersji plik jest pobierany. Podgląd pliku w edytorze tekstowym

```
1 CREATE TABLE test_table0(
2   pin VARCHAR(100),
3   time VARCHAR(100)
4 );
5
6 INSERT INTO test_table0 VALUES(0001, 1524810082);
7 INSERT INTO test_table0 VALUES(0002, 1524810082);
8 INSERT INTO test_table0 VALUES(0003, 1524810082);
9 INSERT INTO test_table0 VALUES(0004, 1524810082);
10
```

Wnioski

Złożoność języka HTML utrudnia pełne przełożenie go na inne języki. Elementy związane z tabelami i atrybutami do tagów nie zostały przeniesione do składni Markdown. Format ten jest jednak skromny i przeznaczony jest do tworzenia prostych dokumentów. W przypadku konwersji na MD problematyczne okazało się przeniesienie elementów bez tagów, ponieważ parser przy tworzeniu obiektów bierze pod uwagę jedynie elementy otoczone tagami.

Konwersja z formatu HTML na pliki CSV i SQL może okazać się przydatna w przypadku dużych tabel, jednak traci sens przy złożonych tabelach w których występuje m.in. scalanie komórek.