Archives service

Description

Services extracting files from provided archive path, and sending message for each file back to the Scraper service

Investigation

Found libraries:

- java: https://commons.apache.org/proper/commons-compress/index.html
- python: https://pypi.org/project/Archive/
- c#: https://products.fileformat.com/compression/net/sharpcompress/
- node: https://www.npmjs.com/package/archive

Chosen library:

- https://commons.apache.org/proper/commons-compress/index.html

Implementation

Technologies and libraries

- Scala with SBT
- Apache Commons Compress

How it works

1. Receives request with json data from the queue (with path to archive in the "file" field)
2. Extracts files from the archive to a directory on host machine "/host/extracted/archive-*currentTimeInMillis*" (e.g. "/host/extracted/archive-1651494403242")
3. Appends "archive" object to the json data (with two fields: *filePathInVolume* and *filePathInArchive*)
4. Sends message with json data to the Scraper service (*words.scraper* queue) for each of the extracted files

Supported archive formats:

- .zip
- .tar
- .tar.gz

Example input

Example for top-level archive (not nested in another archive)

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.zip",
  "originalFile": "/host/test/test.zip",
}
```

Example for .tar archive nested in .zip archive from previous example

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/extracted/archive-1651571108500/tarfolder.tar", //tar archive that was nested in .zip
archive from previous example, and was extracted from that .zip archive
  "archive": {
    "filePathInVolume":"/host/extracted/archive-1651571108500/tarfolder.tar", //path to extracted .tar file
(extracted from .zip file from previous example)
    "filePathInArchive":"/host/test/test.zip/tarfolder.tar" //original path
  }
}
```

Example output

Services sends as many messages as there are extracted files from the archive (one message per extracted file)

Example for top-level .zip archive, file tarfolder.tar was extracted

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.zip",
  "archive": {
    "filePathInVolume": "/host/extracted/archive-1651571108500/tarfolder.tar", //location of the extracted
file
    "filePathInArchive": "/host/test/test.zip/tarfolder.tar" //original location of the file, needed for
displaying on the frontend
   }
}
```

Example with .txt file extracted from nested .tar archive

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/extracted/archive-1651571108500/tarfolder.tar", //tar archive that was nested in .zip
archive from previous example, and was extracted from that .zip archive
  "archive": {
    "filePathInVolume":"/host/extracted/archive-1651571662145/tarfolder/tarfile1.txt", //path to extracted .
txt file (extracted from .tar file nested in .zip file)
    "filePathInArchive":"/host/test/test.zip/tarfolder.tar/tarfolder/tarfile1.txt" //original path of the .
txt file
  }
}
```

How to launch

Service starts automatically after launching the entire docker-compose

Frontend

Description

Desktop app providing graphical interface for interaction with the system.

Investigation

-

Implementation

Technologies and libraries

- Electron (with React)
- JavaScript
- STOMP.js - library providing a STOMP over WebSocket client for web browser and node.js applications

How it works

1. User specifies data in the form
2. Query with json data is sent to an exchange that distributes it further
3. Files matching the query are received and displayed in a list

Example input from user (data send to the queue)

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
}
```

Example result data (single result presented in the interface) with zip and video inside

```
 {
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/volume/test.docx",
  "originalFile": "/host/test/test.zip",
  "text": "some text in docx file",
  "found": true,
  "fileState": {
    "fileFound": true
   },
   "video": {
    "filePathInVolume": "/host/extracted/output/frame-0.png"
   },
  "archive": {
    "filePathInVolume":"/host/extracted/archive-1651571662145/tarfolder/tarfile1.mp4", //path to extracted .
txt file (extracted from .tar file nested in .zip file)
    "filePathInArchive":"/host/test/test.zip/tarfolder.tar/tarfolder/tarfile1.mp4" //original path of the .
txt file
  }
}
```

How to launch

Service starts automatically when launching the entire docker-compose.

Microsoft service

Description

Service searching for text in Microsoft format files (.docx, .pptx, .xlsx) and returning raw text extracted from these files.

Investigation

Libraries found:

- https://github.com/OfficeDev/Open-Xml-Sdk
- https://www.e-iceblue.com/
- https://products.aspose.com/slides/family/
- https://www.telerik.com/aspnet-core-ui
- https://textract.readthedocs.io/en/stable/

Chosen library:

- https://github.com/OfficeDev/Open-Xml-Sdk

Implementation

Technologies and libraries

- C#
- .NET 6.0
- Open-XML-SDK

How it works

1. Receives request with json data from the queue (with file in microsoft format)
2. Extracts raw text from the file
3. Appends "text" field to the json (with raw text extracted from the file)
4. Sends json data to the result queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true
   }
 }
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
   },
  "text": "some text in docx file"
}
```


How to launch

Service starts automatically when launching the entire docker-compose.

OCR service

Description

Service searching for text in image files (.jpg, .png, ...) and returning raw text extracted from these files.

Investigation

Libraries found:

- https://github.com/tesseract-ocr/tesseract#installing-tesseract
- https://ironsoftware.com/csharp/ocr/examples/simple-csharp-ocr-tesseract/
- http://manpages.ubuntu.com/manpages/jammy/en/man1/gocr.1.html
- http://jocr.sourceforge.net/
- https://docs.opencv.org/4.x/d1/dfb/intro.html / https://opencv.org/releases/
- https://cloud.google.com/vision/docs/handwriting#vision-document-text-detection-additional-langs
- https://developers.mindee.com/docs

Chosen library:

- https://github.com/tesseract-ocr/tesseract#installing-tesseract

Implementation

Technologies and libraries

- Kotlin
- Tesseract

How it works

1. Receives request with json data from the queue (with file in image format)
2. Extracts raw text from the image file
3. Appends "text" field to the json (with raw text extracted from the file)
4. Sends json data to the result queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/someimage.png",
  "fileState": {
    "fileFound": true
   }
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/someimage.png",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
    },
  "text": "some text present on the image"
}
```


How to launch

Service starts automatically after launching the entire docker-compose.

Scraper service

Description

Scraper microservice that goes through the directory tree and finds all the files matching formats specified in the query, and sends them to specialized microservices.

Investigation

-

Implementation

Technologies and libraries

- Elixir

How it works

1. Service receives query with json data from queue
2. Service goes through the directory tree and finds all the files matching formats specified in the query
3. After finding such file, it appends its path as "file" field in json data
4. Sends modified json data to an exchange that distributes files across specialized microservices handling files with specific file type (**so for each query received, it sends a separate query for each file found**)

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true
   }
 }
```


How to launch

Service starts automatically after launching the entire docker-compose.

State service

Description

Microservice to remove all files created by our system (files extracted from archives and frames extracted from video files).

Investigation

Implementation

Technologies and libraries

- Java
- Apache commons-io

How it works

1. Receives copy of every message sent to frontend by text analyzer
2. File is removed if it has been created by our system.

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx(zip)",
  "text": "some text in docx file",
  "found": true,
  "fileState": {
    "fileFound": true
   },
  "video": {
    "filePathInVolume": "/host/test/time_of_frame.png"
   },
  "archive": {
    "filePathInVolume": "/volume/test.png",
    "filePathInArchive": "/host/x.rar/y.zip/z.zip/test.png"
   }
}
```

Example output

This service does not send any messages.

How to launch

Service starts automatically when launching the entire docker-compose.

Text analyzer service

Description

Service finding specified words in raw text provided by services specialized in various file formats.

Investigation

Libraries found:

- https://github.com/buka102/Lucene-NET-Core
- https://lucene.apache.org/pylucene/
- https://lucene.apache.org/core/index.html
- https://solr.apache.org/
- https://nutch.apache.org/ --> also for html, xml, zip
- https://www.elastic.co/

We didn't use any of the libraries, because they weren't needed.

Implementation

Technologies and libraries

- C#
- .NET 6.0

How it works

1. Receives request with json data from the queue
2. Searches provided words in raw text specified in the data
3. Appends "found" field to the json (with true/false value depending on search result)
4. Sends data to the result queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
    },
  "text": "some text"
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
    },
  "text": "some text",
  "found": true
}
```


How to launch

Service starts automatically after launching the entire docker-compose.

Words typos service

Description

Service generates typos from requested words.

Investigation

Python:

https://www.codegrepper.com/code-examples/python/create+a+typo+with+python

https://github.com/alexyorke/butter-fingers/tree/master/butterfingers

Ruby:

https://github.com/siman-man/typogen?fbclid=IwAR0yvdr8oR40YVP3dbPPF6mu-aPMQPJDCyFCj45ReQYj98gQWQggcR4MQ_4

Implementation

Technologies and libraries

1. Python

2. https://www.codegrepper.com/code-examples/python/create+a+typo+with+python - script that generates typos.

How it works

1. Receives request with json data from the queue
2. Accesses file specified in data from the queue
3. Generates typos for each word specified in data from the queue
4. Appends "words" field to json with typos as payload
5. Sends data to the scraper queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["typos", "scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text", "sime", "sone", "twxt", "texp"],
  "filters": {
    "searchModes": ["scraper", "typos"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
}
```

How to launch

Service starts automatically after launching the entire docker-compose.

Text extractor service

Description

Service extracting raw text from files with .txt format.

Investigation

Implementation

Technologies and libraries

Python

How it works

1. Receives request with json data from the queue
2. Accesses file specified in data from the queue
3. Extracts raw text from file
4. Appends "text" field to json with the extracted text as payload
5. Sends data to the text queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true
   }
 }
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/test/test.docx",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
    },
  "text": "some text"
}
```


How to launch

Service starts automatically after launching the entire docker-compose.

Words forms service

Description

Service generating list of forms of the words that were given in a phrase by the user.

Investigation

1. NLTK Library, **Lemma** class. Lemma.derivationally_related_forms(). https://stackoverflow.com/questions/45145020/with-nltk-how-can-i-generate-different-form-of-word-when-a-certain-word-is-giv
2. https://stackoverflow.com/questions/61030931/wordnet-getting-derivationally-related-forms-of-a-word
3. https://github.com/ksopyla/awesome-nlp-polish
4. http://morfeusz.sgjp.pl/

Other libraries: **TextBlob, Gensim, spaCY, polyglot.**

Chosen solution:

http://morfeusz.sgjp.pl/

Implementation

Technologies and libraries

- Python
- Morfeusz2

How it works

1. Receives request with json data from the queue
2. Creates list of different forms of the requested word
3. Appends list containing the requested word and its forms to “words” field in json
4. Sends data to the text queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["forms", "scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text", "texts", "textual"],
  "filters": {
    "searchModes": ["scraper", "forms"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  }
}
```

How to launch

Service starts automatically after launching the entire docker-compose.

Words synonyms service

Description

Service generating list of synonyms of the words that were given in a phrase by the user.

Investigation

Libraries found:

1. NLTK Library, PyDictionary https://www.geeksforgeeks.org/get-synonymsantonyms-nltk-wordnet-python/
2. Synonyms Apache OpenMLP https://github.com/apache/opennlp-sandbox/blob/master/opennlp-similarity/src/main/java/opennlp/tools/apps/relevanceVocabs/SynonymMap.java
3. https://github.com/bensheldon/panlexicon-rails?fbclid=IwAR2uQUEwTFQc6DS7WZtZ6k0qx1nlCYyZbfZ6snpBR38VDJ9znDV0Ud4R4Fg

Polish dictionary: https://spacy.io/models/pl

Implementation

Technologies and libraries

- Python
- Thesaurus Polish Dictionary (https://dobryslownik.pl/pobierz/)

How it works

1. Receives request with json data from the queue
2. Creates list of synonyms to the requested word
3. Appends list containing the requested word and its synonyms to “words” field in json
4. Sends data to the text queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text", "texts", "textual"],
  "filters": {
    "searchModes": ["synonyms", "scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  }
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text", "texts", "textual"],
  "filters": {
    "searchModes": ["scraper", "synonyms"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  }
}
```

How to launch

Service starts automatically after launching the entire docker-compose.

Video Service

Description

Services extracting files from provided video path, and sending message for each frame back to the Scraper service

Investigation

Two approaches:

use our ocr service on frames extracted from video with:

- https://github.com/iammert/Frames frames extraction (java and kotlin)
- https://github.com/FFmpeg/FFmpeg processing different multimedia (mainly C)
- https://github.com/Kitware/kwiver image and video analysis (mainly C++)
- https://pypi.org/project/opencv-python/ openCV for python
- https://docs.opencv.org/4.x/d9/d52/tutorial_java_dev_intro.html openCV for java

apply library function on the whole video:

- https://github.com/A9T9/Copyfish javascript

Implementation

Technologies and libraries

- Rust
- OpenCV

How it works

1. Receives request with json data from the queue (with path to video in the "file" field)
2. Extracts frames from the video to a directory on host machine "/host/extracted/output"
3. Sends message with json data to the Scraper service (*words.scraper* queue) for each of the extracted frame

Example input

```
{
  "phrase": "some text",
  "path": "~/testFolder/",
  "words": ["some", "text"],
  "filters": {
    "searchMode": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/testFolder/test_video.mkv"
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/testFolder/",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [
      ".pptx",
      ".docx",
      ".txt",
      ".jpeg",
      ".jpg",
      ".png",
      ".mp4",
      ".zip"
    ]
  },
  "file": "/host/testFolder/test_video.mkv",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
    },
  "video": {
     "filePathInVolume": "/host/testFolder/extracted/output/frame-0.jpg"
  }
}
```


How to launch

Service starts automatically after launching the entire docker-compose

PDF service

Description

Service searching for text in PDF files (.pdf) and returning raw text extracted from these files.

Implementation

Technologies and libraries

- Scala
- sbt
- Apache PDFBox

How it works

1. Receives request with json data from the queue (with PDF file)
2. Extracts raw text from the file
3. Appends "text" field to the json (with raw text extracted from the file)
4. Sends json data to the result queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [".pdf"]
  },
  "file": "/host/test/test.pdf",
  "fileState": {
    "fileFound": true
   }
 }
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [".pdf"]
  },
  "file": "/host/test/test.pdf",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
   },
  "text": "some text in pdf file"
}
```

How to launch

Service starts automatically when launching the entire docker-compose.

OpenDocument service

Description

Service searching for text in OpenDocument format files (.odt, .odf) and returning raw text extracted from these files.

Implementation

Technologies and libraries

- Java
- Maven
- Apache Tika

How it works

1. Receives request with json data from the queue (with file in OpenDocument format)
2. Extracts raw text from the file
3. Appends "text" field to the json (with raw text extracted from the file)
4. Sends json data to the result queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": ["odt"]
  },
  "file": "/host/test/test.odt",
  "fileState": {
    "fileFound": true
   }
 }
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": ["odt"]
  },
  "file": "/host/test/test.odt",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
   },
  "text": "some text in odt file"
}
```

How to launch

Service starts automatically when launching the entire docker-compose.

Converter service

Description

Service which converts .mp3 and .mp4 files to .wav files.

Implementation

Technologies and libraries

- Python
- ffmpeg

How it works

1. Receives request with json data from the queue (with file in .mp3 or .mp4 format)
2. Converts file to .wav format and saves it to /host/extracted directory with "-converted" suffix.
3. Sends json data with location of converted file

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [".wav", ".mp3"]
  },
  "file": "/host/test/test.mp3",
  "fileState": {
    "fileFound": true
   }
 }
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [".wav", ".mp3"]
  },
  "audio": {
    "filePathInVolume": "/host/extracted/test-converted.wav"
  }
  "file": "/host/test/test.mp3",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
  }
}
```

How to launch

Service starts automatically when launching the entire docker-compose.

Audio service

Description

Service searching for text in .wav and returning raw text extracted from these files.

Implementation

Technologies and libraries

- Python
- Google Speech Recognition

How it works

1. Receives request with json data from the queue (with file in .wav format)
2. Uses speech recognition to extract raw text from the file
3. Appends "text" field to the json (with raw text extracted from the file)
4. Sends json data to the result queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [".wav"]
  },
  "file": "/host/test/test.wav",
  "fileState": {
    "fileFound": true
   }
 }
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper"],
    "fileTypes": [".wav"]
  },
  "file": "/host/test/test.wav",
  "fileState": {
    "fileFound": true,
    "fileProcessed": true
   },
  "text": "some text in wav file"
}
```

How to launch

Service starts automatically when launching the entire docker-compose.

Translations service

Description

Service, which translates given phrase to defined subset of languages

Implementation

Technologies and libraries

- NodeJS
- Google Translate

How it works

1. Receives request with input phrase and languages
2. Translates input phrase to defined languages
3. Appends translated phrase to "words" array
4. Sends json data to the words queue

Example input

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text"],
  "filters": {
    "searchModes": ["scraper", "translations"],
    "fileTypes": [".txt"]
  },
  "languages": ["pl", "de"]
}
```

Example output

```
{
  "phrase": "some text",
  "path": "~/test",
  "words": ["some", "text", "jaki tekst", "etwas Text"],
  "filters": {
    "searchModes": ["scraper", "translations"],
    "fileTypes": [".txt"]
  },
  "languages": ["pl", "de"]
}
```

How to launch

Service starts automatically when launching the entire docker-compose.