Principles Of Software Engineering

Initial Technical Report

Team CHARLIE

Project: LACR-Search

Critical use case model

Parsing XML files

During the process of transcribing the documents were stored in XML format according to the TEI standard. The system should parse the XML structure of the files and extract the important data that should be provided to the search engine. Maintaining the data structure of the XML files is important to allow correct search results. The module XML to JSON transforms the extracted data in JSON format that is suitable for Elasticsearch and contains the important information stored as arguments within the XML tags.

In the process of document parsing the system links each of the XML files with the record of extracted text that is then imported into the search engine. This method allows quick responsiveness of the search engine and direct link for each element in the search result to original XML file of the transcribed documents.

Searching

The search functionality of the system is based on the open-source project Elasticsearch. This core element of the system provides quick search response even with large amount of data. This open-source project have large pack of tools for language processing that "work out of the box". Some of those features will be used to provide suggestions and misspelling correction for the search query.

The good documentation of the Elasticsearch project and the large developer community allows quick development process and simple maintenance.

The storage layer module Search Interface generates specific JSON queries that are then sent to the search engine. This module transforms the response from the search engine into a list of matched text results. This list is extended by the application layer module Search Handler which links each of the matched results to the corresponding XML record.

The already extended list of search results along with search suggestions from the Elasticsearch are passed to the view layer module Search Documents. This module is responsible for generating the HTML content of the web page that contains the search results of the user input.

The searching functionality provides the use of filters such as:

- ID of document
- Annotations
- Proper names

Upload/Edit XML files

This functionality of the system is allowed only for authorised users with administrative privileges. It aims to provide simple way of importing new documents. This process requires the user to submit scanned image of the original document along with XML file of the transcribed version. Both of the submitted documents are stored from the system and can be modified or removed later only from the authorised user.

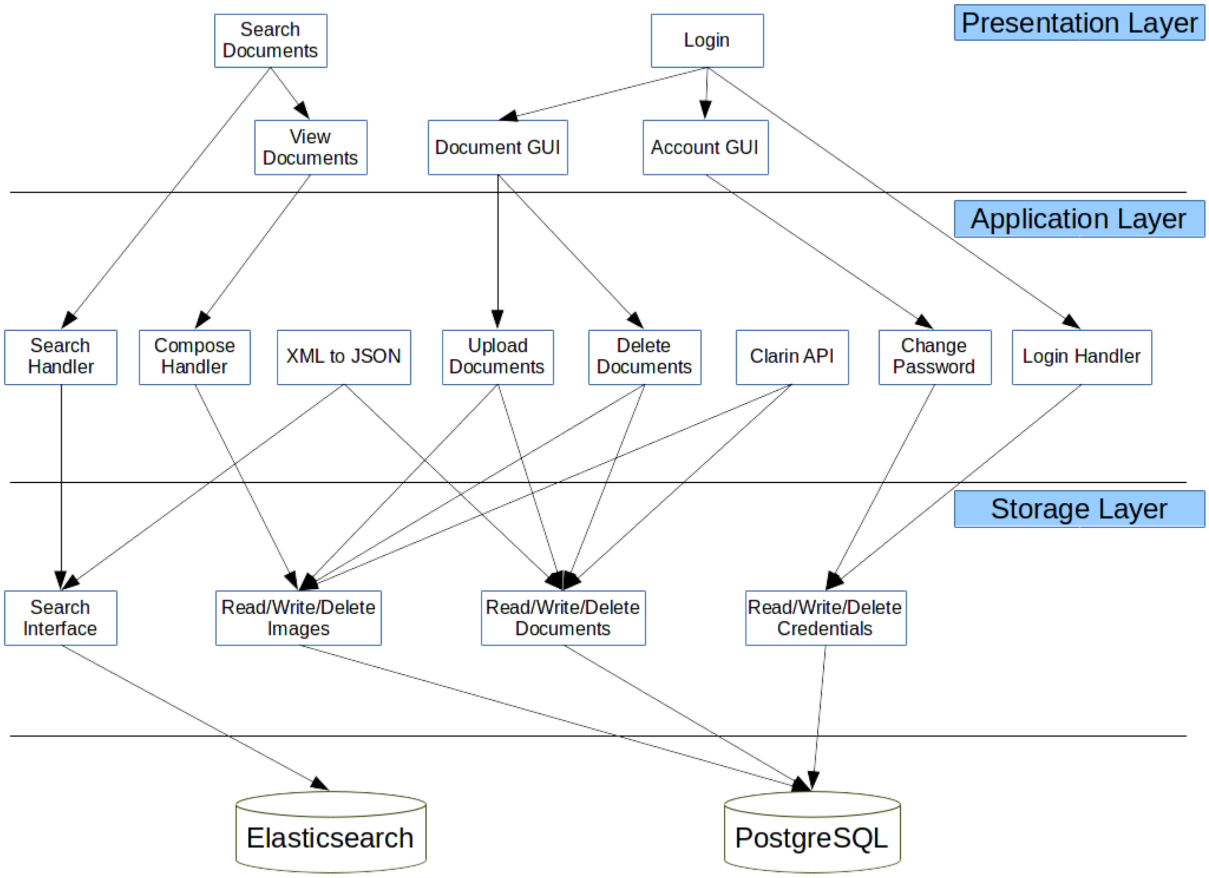
The application layer module Login handler is responsible for the user authentication. It allows only authorised users to import and modify documents. The fundamental component of this part of the system is based on the Ruby on Rails gem Devise which provides complete authentication system that “works out of the box”. Proper configuration of this Ruby gem will provide secure user authentication process for the system. The storage layer module Read/Write/Delete Credentials is interacting with the gem Devise and reads or modifies the stored user credentials.

The storage layer module Read/Write/Delete documents receives the image and XML files from the user and stores them on the server by creating the corresponding records in the PostgreSQL database server. This method allows efficient access and retrieve of the imported documents.

Candidate system architecture

The system provides three main functionalities:

- Display Documents.
- Searching through transcribed version of the documents.
- Control of the content.



The system is based on the three-tier architecture model. Each of these functionalities consist of separate modules split in each of the three layers of the system.

The presentation layer of the system consist of the modules responsible for the user interface. The application layer provides the business logic for each of the functionalities and the storage layer includes the modules that interact with the PostgreSQL database, Elasticsearch and apply read/write operations to the stored files.

The presentation layer have significant role in transforming the data gathered from the system into visual format that is displayed to the user and retrieving the user input that is processed by the system. Significant aspect of the modules in this layer is to provide smooth navigation between the different functionalities. To achieve this the modules in the presentation layer make use of CSS and JavaScript libraries provided by several open-source project such as BootStrap and Jquery.

The components in the application layer contain the business logic of the system. This layer implements the core part for each of the functionalities. The modules in this layer are responsible for transforming the data gathered from the storage layer into format that would be suitable for the presentation layer. The implementation for some of the system's functionalities is achieved by integrating open-source Ruby on Rails gems such as Searchkick and Devise. This method provides quick development process and bug-free system functionalities.

The modules in the application layer analyse the user input and executes the corresponding functions. This layer contains the methods that are responsible for the user authentication, uploading new documents and composing the search result into suitable format that can be processed from the presentation layer. Minimising coupling requires each of these functionalities to be managed from small number of classes and methods at the same time to maximise cohesion of the system we require to split each of the functionality into at least several independent classes or methods. Achieving both minimum coupling and maximum cohesion in the application layer of the system requires each functionality to be composed into two independent parts.

The user authentication of the system is based on the ruby gem Devise which implements all necessarily modules to provide secure authentication. However, the module Login handler stores additional data and records the system requires for each user where the module "Change Password" provides the functionality for authenticated user with administrative privileges to set new password for the every user of the system.

The functionality of displaying stored documents is driven by the module "Compose handler" which is responsible for combining the transcription of a document with the corresponding scanned image of the document and the additional information such as: date, document ID.

The module Search handler is responsible for analysing the user input, executing the corresponding functionality of the storage layer module Search Interface and provide the results in suitable format to the presentation layer module Search Documents.

The users of the system are allowed to upload new documents and modify/remove existing ones. This functionality is composed of two modules Upload Document and Delete Document. The first module provides the upload/modify function and the second module is responsible for removing documents from the system. These are critical components of the system and they are available only for authenticated users with corresponding privileges.

The storage layer contains the modules that are responsible for the interaction of the system with the PostgreSQL database and the search engine. These modules are transforming the data from the user input to SQL queries and JSON requests. They are then sent to the corresponding external servers. The components in this layer also transform the response received from both the PostgreSQL database and Elasticsearch server into proper format that can be processed by the application layer.

Initial risk assessment

In order to successfully finish this project it is crucial to identify possible obstacles and develop a plan to cope with them. There are many obstacles which could influence the proposed time schedule and the major ones are these:

- Personnel shortfalls (High)
- Lack of skills and knowledge in this domain (High)
- Human error (High)
- Not enough accurate search results (High)
- Developing the wrong user interface (Medium)
- Unrealistic schedule (Low)
- Inadequate budget (Low)
- Requirements volatility (Low)

One of the obstacles is personnel shortfalls. Since the application should be developed in a small team, it represents very high risk. Especially, if only one team member is specialist in particular technology. The event of personal failure would introduce major delay in schedule. Therefore, it is necessary to decrease the impact. It will be achieved by assigning tasks to a pair of developers and not individuals. Since they will work together, both will have similar knowledge about the problematic and one can still continue working on the task. This technique will lower the associated risk and delay.

Another difficulty with high risk associated is lack of skills and knowledge in this particular domain. Because it is highly specialized field, It is very likely that the developers would misunderstand the requirements specification. For this reason, it is crucial to closely cooperate with the client who is an expert and can easily direct the developers. The regular meetings are necessary in order to prevent this situation.

The next item on the list is human error. It is very common that developer make mistake without noticing. This could lead to major failure and malfunctioning of the system and potentially to great delay in schedule. So it is required to write tests for every modules in this application. This will decrease the chance the chances of human error.

In case that the search results are not enough accurate, the search engine needs to be adjusted for a particular set of searched documents and language. If the used language in the documents is not one of the supported by Elasticsearch Analyzer, the search results can be improved by increasing Levenshtein distance for the searched query. The next thing which can improve search results is introducing autocomplete function. It will offer autocompletion only from the set of words which are currently present in its index. Therefore, it will decrease the chance of wrong input. The last but not least thing is enabling suggestions for searched query. It is important to tune up these parameters and sometime maybe even feasible to use own stemmer.

Developing the wrong user interface is less likely to happen because it should be simple web page with search functionality. Despite this fact, it is necessary to regularly discuss the progress with the client and incorporate reasonable feedback as soon as possible.

Important part of the development is to make realistic schedule which should be followed. Based on the time which was required for the development of the functional prototype, it was estimated the time schedule for developing the complete application. The plan also include time reserve which should cover some minor delays. Even though, it is essential to monitor the progress.

The next thing on the list is inadequate budget. The chance that it would dramatically increase are low due to the fact that the application incorporate already developed technology which was not introduced recently. Therefore it was already well tested and the core technology is also used by many different applications on the Internet. It is also important that the core technology has great community support and in case of any troubles it is possible to require paid support. Based on the time and complexity of the functional prototype were estimated costs for this application.

The last part which could introduce delay in schedule is requirements volatility. The core things of requirements specification should not change because they represent the whole idea of the application. In fact only minor changes can be done and they should not cause any major delay. On the other hand, it is vital to regularly monitor if the requirements are fulfilled.

Initial project plan

The first and most important part of this application is a search engine called Elasticsearch. It is the core technology which needs to be installed and properly set up for the needs of this application. This search engine is not particularly developed for searching in Latin or Middle Scots languages and therefore it will require more time to tune it up.

The second important part of the application is relational database. Based on research was made the decision to use PostgreSQL. It is well known database with great performance and it will be used to store original XML files, credentials, images and privileges.

The third thing which is required to be done is a module which will provide translation of XML documents to JSON. The reason is that Elasticsearch is natively used for searching inside JSON files and not XML which will be used. In parallel RSpec tests need to be written for this module.

The fourth stage of the development includes creating all the modules in application layer. These modules provides the business logic for the application. Again in parallel with the development of these modules their tests in RSpec need to be written.

The fifth part includes fine tuning Elasticsearch, improving search results and potentially including Latin and Middle Scots thesaurus in to Elasticsearch. In this time the storage layer need to be done and tested.

In the sixth stage, the whole application layer need to be finished and tested too. This also include XML to JSON module. Now it can begin the work on presentation layer and its tests.

The seventh and also last stage is about finishing the presentation layer and writing documentation.

Schedule

The schedule for developing this application is 10 weeks.

1. Week: Installing dedicated Linux OS with Docker. Deploying separate linux containers with Elasticsearch, PostgreSQL and Ruby on Rails
2. Week: Setting up Elasticsearch, PostgreSQL, Ruby on Rails with Puma webserver
3. Week: Setting up Elasticsearch and writing storage layer modules with tests

4. Week: Testing Elasticsearch and finishing the work on storage layer. Start developing application layer and its tests.
5. Week: Tuning up Elasticsearch, finishing application layer and start working on presentation layer and design of the interface.
6. Week: Set up Nginx web-server “in front of” the Ruby on Rails application. Testing security and performance. Start writing documentation for admin users.
7. Week: Major meeting with client, receiving feedback on user interface and implementing changes.
8. Week: Writing documentation for end user.
9. Week: providing access to documents and testing Clarin API communication.
10. Week: Application will be delivered and presented to the customers.

Milestones

- Setting up server with Docker, Elasticsearch, PostgreSQL and Ruby on Rails
- Elasticsearch is tuned up for this application purposes
- Pass all tests for storage layer
- Pass all tests for application layer
- Pass all tests for presentation layer
- Documentation for admin users is done
- Satisfied customer

Budget

The application was expected to be designed, developed and documented within 5 months and the overall budget is approximately £46,229.

- Expense for labour is £16,000

- Expense for rent is £10,500
- Hardware depreciation is about £340
- Marketing and Advertising £ 384
- Accounting and Bookkeeping Services £300
- Legal Advice £5 000.
- Travelling Costs £6 000.
- Reserve 20%