LACR Search

Maintenance Manual & Documentation

Hardware Requirements

Minimal Hardware Requirements	Recommended Hardware Requirements:
• 2 x virtual CPUs • 16 GB RAM memory • +500GB hard drive	• 16 x virtual CPUs • 64 GB RAM memory • 2TB hard drive

The hardware requirements are calculated for all XML transcriptions and Images of the LACR project.

The application itself would work with much smaller resources such as:

2 CPUs, 4 GB memory, 4 GB disc space.

Installation instructions

Installation steps for Ubuntu 16.04

1. Install and setup Docker on Ubuntu 16.04:
   1.1. Please follow the instructions provided by Digital Ocean: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04
      - Note: Common mistake is that Step 2 is not completed. It is important to **log out** and **log back in** from your current session to apply the changes of this step.

```
sudo usermod -aG docker $(whoami)
```

   1.2. As an alternative, installation instructions are provided from the official Docker documentation: https://docs.docker.com/engine/installation/
2. Install docker-compose

   2.1. Using the Python package manager "*pip*"

```
pip install docker-compose
```

   2.2 Alternative ways to install docker-compose are provided in the Docker documentation: https://docs.docker.com/compose/install/

3. Navigate to lacr-search/src and execute "setup.sh". This is simple shell script which aims to automate the installation process.

```
cd lacr-search/src && ./setup.sh
```

3.1. Alternatively, the web application can be started with the following steps:

4. Navigate to lacr-search/src

```
cd src
```

4.1.1. Build Docker images

```
docker-compose build
```

4.1.2. Create database structure

```
docker-compose run web bash -c "rails db:create && rails db:migrate && rails db:seed"
```

4.1.3. Start Lacr Search

```
docker-compose up
```

- To stop the application press once Ctrl + C or twice to force the stop -

- To run the server in background use –

```
docker-compose up -d
```

Useful Tips:

Full list of available options https://docs.docker.com/compose/reference/overview/

Stop the server:	docker-compose stop
Start the server:	docker-compose start
Stop the server and destroy all containers:	docker-compose down
List all containers	docker-compose ls
Open shell inside the Ruby container	docker-compose exec web bash

Note: There are two common problems you might encounter during installation.

1. Error message "A server is already running."
   - This error occurs when Docker has killed the Ruby container. Happens, if you press Ctrl + C twice.
     - To solve this problem remove the file tmp/pids/server.pid

```
sudo rm lacr-search/tmp/pids/server.pid
```

2. Error message: FATAL: database "…" does not exist
   - Occurs, when the container for PostgreSQL has been destroyed. This happens, as a result of "docker-compose down"
   - To solve this problem execute:

```
docker-compose run web bash -c " rails db:create && rails db:migrate && rails db:seed"
```

Software dependencies and libraries used

Description of third-party software used:

Name	Description
Ruby-on-Rails	Web Framework used as base for the LACR Search.
Docker	Provides container management.
docker-compose	Used to automate container management.
Elasticsearch	Search engine used to provide main search functionality.
BaseX	XML native database, used as XQuery processor.
ImageMagick	Used to convert images from TIFF to JPEG format.

List of ruby gems and JavaScript libraries:

Name	Description
Twitter Bootstrap	Used to create web interface for responsive web-design.
JQuery	Makes it much easier to use JavaScript.
Font-awesome	Set of icons, specially designed for websites and applications.
CarrierWave	Simple and extremely flexible way to provide file upload for Ruby-on-Rails web applications.
Rubyzip	Ruby gem for working with zip files.
Mini_magic	Ruby wrapper for ImageMagick
Prawn	Ruby PDF generation library.
will_paginate-bootstrap	Pagination library for Ruby on Rails.
Nokogiri	HTML and XML parser.
Searchkick	REST API client for Elasticsearch.
Devise	Flexible user authentication.
FullPage.js	Create full screen (landing) pages.
jquery.highlight.js	Used to highlight pieces of text on the web page.
jquery.noty.js	Used to display UI notification.
jquery.zoom.js	A plugin to enlarge images mouse over.
js.cookie.js	JavaScript API for handling cookies.
prittify.js	An embeddable script to provide syntax highlight for code snippets.
scrolloverflow.js	Smooth scrolling.
HisTEI CSS	HisTEI Stylesheets
Typhoeus	Significantly increase performance with persistent HTTP connections. (Used to optimise Searchkick)
Oj	JSON parser and Object marshaller. (Used to optimise Searchkick)
devise-bootstrap-views	Bootstrap views for Devise

Description of files used

Configuration files

- `docker-compose.yml`
  - Specifies docker-compose services, dependencies between containers and others.

- `Dockerfile`
  - Used to build Docker image for the Ruby-on-Rails application.

- `src/Gemfile`
  - Contains a list of Ruby gems which are required from the application.

- `src/config/application.rb`
  - Require Gems listed in the Gemfile. The settings in src/config/environments/* take precedence over those specified here.

- `src/config/boot.rb`
  - Set up gems listed in the Gemfile. Specifies the path to the Gemfile.

- `src/config/cucumber.yml`
  - Store and reuse commonly used Cucumber command line arguments.

- `src/config/database.yml`
  - Database configurations (PostgreSQL).

- `src/config/environment.rb`
  - Load and initialises the web application.

- `src/config/puma.rb`
  - Configuration for the Puma web server.

- `src/config/routes.rb`
  - Configurations of the Rails router. Links URL paths and request types with methods defined in the controllers.

- `src/config/secrets.rb`
  - Contains the secret key used to verify integrity of signed cookies.

- `src/config/environments/*`
  - Contains configuration files used mainly for setting environment variables on start.

- `src/config/initializers/*`
  - Contains initialisation files for the Rails application.

Models

- `src/app/models/application_record.rb`
  - Main model, defines class inherited from all other models.

- `src/app/models/page_image.rb`
  - Responsible for storing images of the scanned documents using image_uploader.rb.

- `src/app/models/search.rb`
  - Responsible for the interaction with Elasticsearch (ES). Import and configure Searchkick gem. Provides link with DB table Searches which store information made searchable through ES.

- `src/app/models/transcription_xml.rb`
  - Responsible for parsing and storing information from the uploaded XML files using Transcription_xmls DB table and xml_uploader.rb

- `src/app/models/tr_paragraph.rb`
  - Responsible for storing paragraphs in XML and HTML format and generate PDF file from selected documents.

- `src/app/models/user.rb`
  - Responsible for user authentication, imports and configures Devise gem.

Uploaders

- `src/app/uploaders/image_uploader.rb`
  - Configurations for upload of image files.

- `src/app/uploaders/xml_uploader.rb`
  - Configurations for upload of XML files.

Controllers

- `src/app/controllers/application_controller.rb`
  - Main controller class extended by all other controllers.

- `src/app/controllers/documents_controller.rb`
  - Methods:
    - **Index**: Used for Document browse functionality.

    - **Selected**: Corresponds to /doc/selected URL. Shows all selected paragraphs. The list of selected entries is transferred using Cookie.

    - **List**: Responds with JSON response with a list of all pages and volumes stored in the DB. Used to generate the navigation menu on /doc/show and /doc.

        - **New**: Shows the view for document upload. This method is accessible only for administrative users.
        - **Show**: Takes get parameters “p” and “v”, corresponding to page and volume respectively. Used to load the /doc/show where the transcription and image are shown one next to the other.

        - **Page_simplified**: Takes get parameters “p” and “v”, corresponding to page and volume respectively. Responds with HTML response which renders the documents/page_simplified view. Shows all paragraphs on a page with information about the Date, ID, and Language. Used to for the Document browse functionality. In case of incorrect input it responds with HTTP code 500.

        - **Page**: Takes get parameters “p” and “v”, corresponding to page and volume respectively. Responds with HTML response which renders the documents/page view. Shows all paragraphs on a page with the corresponding image. Used by the document/show functionality. In case of incorrect input it responds with HTTP code 500.

        - **Upload**: This method utilises the strong parameters protection from Rails. Only POST HTTP requests are accepted with parameters xml[] and image[]. Note both of these are in array type. This is used to allow upload of multiple files at the same time. This method allows upload of XML files and/or Image files. It process them independently. The XML files are filtered here if they cannot be opened with Nokogiri or do not include HisTEI namespace. This method calls the function histei_split_to_paragraphs defined in the Transcription_xml model to parse and store the content of XML files in the DB and ES. This method also uploads the successfully processed XML files to BaseX and calls the reindex method of Searchkick to generate new index for ES. This method is accessible only for administrative users.

        - **Destroy**: This method is used to remove document pages. It is accessible only for administrative users. Accepts only POST HTTP request, with parameter “selected” of type list, which contains tuples. Each tuple has keys “page” and “volume”. This method iterates through each paragraph and remove it from PostgreSQL, Elasticsearch and BaseX. (It does not remove the XML file stored on the disk) Responds with JSON response which contains appropriate message indicating success or failure.

- `src/app/controllers/download_controller.rb`
    - Methods:
        - **Index**: Accepts POST HTTP request with parameter “selected” of type list which contains tuples. Each tuple has keys “page” and “volume”. This method iterates through each paragraph and creates a **set** of corresponding XML and image files. Note that we use data type set to not allow duplicates. This prevents same XML or Image file name to

be included multiple times. Each of the files in this set are copied into zip archive created with RubyZip gem. The archive is stored in the /tmp directory of the server. When the archive is created this method generates random string named “key” which contains 6 characters. This “key” string is used as a key of Ruby hash “tmp_file” which is stored for user session. The “tmp_file” hash is used to store the file path of the archive. This is used to prevent user control over the path to the file. The string “key” is sent as JSON response to indicate successfully created archive. Otherwise error message is sent.

- **Archive**: This method accepts the 6 character “key” string and retrieves the file path of the archive. It sends the zip file as response. After the file transfer is finished the file stored in /tmp is removed. If the key parameter is not found in “tmp_file” hash, response HTTP 404 is sent.

- **Selected_gen_pdf**: Retrieves a list “selected_entries” from a browser cookie. Each element of the list corresponds to ID of paragraph. This method calls the function “print_data” defined in the TrParagraph model. Responds with PDF file as response. If there are no IDs or not paragraphs found “Not Found” error message is shown.

- `src/app/controllers/home_controller.rb`
  - This controller is used for the home page. The home page is implemented primarily with JavaScript and CSS scripts.

- `src/app/controllers/search_controller.rb`
  - Methods:
    - **Search**: This method implements the simple and advanced search functionality. It makes use of Searchkick to interact with Elasticsearch. This method is utilising the strong parameters from Rails. Filters the user input with the function simple_search_params which accepts only the parameters specified in the table below.

Parameter name	Description
q	Query -- Plain text query which is send to Elasticsearch from Searchkick.
r	Results per page -- Number of search results per page (pagination is used).
m	Misspellings -- Number of “Spelling variants” corresponds to misspellings in Elasticsearch.
o	Order by -- A few different conditions are used here. Each of them is passed using the “where” condition of Searchkick
sm	Search Method -- Sets which search method to be used by Elasticsearch.
entry	Entry ID -- This the ID of paragraph (entry)
date_from	Date From -- Used to filter the results with condition “gte” (greater than or equal)
date_to	Date To -- Used to filter the results with condition “lte” (less than or equal)
v	Volume -- Used to filter the results by volume. This parameter is represented

	as a list of numbers. Search result will match this condition only if its volume number exist in the specified list.
pg	Page -- Used to filter the results by page. This parameter is represented as a list of numbers. Search result will match this condition only if its page number exist in this list.
pr	Paragraph -- Used to filter the results by paragraph. This parameter is represented as a list of numbers. Search result will match this condition only if its paragraph number exist in this list.
lang	Language -- Filters the results by language. (e.g. “lat” for Latin)
page	Page for pagination -- Navigation between pages of the search results.

- **Autocomplete**: Used for the autocomplete functionality of the simple search. The implementation performs search with method “word_start” and responds with JSON response. The response contains only unique words which have been “highlighted” as matching the search query from Elasticsearch. Non-alphabetic characters are removed.

- **Autocomplete_entry**: Autocomplete for the search for Entry ID (xml:id). It is implemented as search with method “word_start” with disabled misspellings.

- `src/app/controllers/xquery_controller.rb`
  - Index: Renders the page with XQuery examples.
  - Show: Send XQuery expression to BaseX utilising the /lib/BaseXClient.rb and shows the result.

Views

- `src/app/views/devise/confirmations/new.html.erb`
  - Confirm email address for new user.

- `src/app/views/devise/mailer/confirmation_instructions.html.erb`
  - Template message for email send for confirmation of user account.

- `src/app/views/devise/ mailer/reset_password_instructions.html.erb`
  - Template message for email send when password reset was requested.

- `src/app/views/devise/ mailer/unlock_instructions.html.erb`
  - Template message for email with unlock instructions for user account.

- `src/app/views/devise/passwords/edit.html.erb`
  - View used for password change.

- `src/app/views/devise/passwords/new.html.erb`
  - View used to reset forgotten password.

- `src/app/views/devise/registrations/new.html.erb`
  - View used for registration of new user.

- `src/app/views/devise/ registrations/edit.html.erb`
  - View used to edit user details.

- `src/app/views/devise/sessions/new.html.erb`
  - Sign up page.

- `src/app/views/devise/shared/_links.html.erb`
  - Partial page used to organise the links below the forms.

- `src/app/views/devise/unlocks/new.html.erb`
  - Page used for unlock instructions.

- `src/app/views/documents/index.html.erb`
  - This view is used Document browse. The page is generated using JavaScript. See `src/app/assets/javascripts/documents_browse.js`

- `src/app/views/documents/new.html.erb`
  - Upload of new XML and image files.

- `src/app/views/documents/_page.html.erb`
    - Partial page used by “show.html.erb” and “search.html.erb”. Generates HTML for to list XML transcriptions along with images on side.

- `src/app/views/ documents/_page_simplified.html.erb`
    - Partial page used by “documents.html.erb”. Generates HTML for to list XML transcriptions.

- `src/app/views/documents/selected.html.erb`
    - Show a list of selected paragraphs.

- `src/app/views/documents/show.html.erb`
    - Display transcriptions accompanied with the images.

- `src/app/views/documents/upload.html.erb`
    - This view is used when the file upload has finished. Shows a list of successfully, unsuccessfully uploaded files, and overwritten.

- `src/app/views/home/index.html.erb`
    - Home page including advanced_search.html.erb

- `src/app/views/search/search.html.erb`
    - Display the search results.

- `src/app/views/search/_search_tools.html.erb`
    - Partial page included in the page displaying search results to apply search filters such as spelling variants, number of results per page, etc.

- `src/app/views/search/advanced_search.html.erb`
    - Consist of the input fields used for advanced search. Used only on the home page.

- `src/app/views/xquery/index.html.erb`
    - XQuery examples and show search input field.

- `src/app/views/xquery/show.html.erb`
    - Show search input field and the result received from BaseX.

- `src/app/views/xquery/_search_form.html.erb`
    - Input field used from “xquery/show.html.erb” and “xquery/index.html.erb”.

Assets

JavaScript

- `src/app/assets/config/manifest.js`
    - Specifies which file to be processed from Rails as assets.

- `src/app/assets/javascripts/application.js`
    - Contains the JavaScript which will appear on every page. It is used mainly to globally include libraries such as JQuery and Bootstrap.

- `src/app/assets/javascripts/documents.js`
    - On page-ready it will get a list of all page and volume numbers and will generate HTML form of menu used to browse the documents. The function “load_document()” is used to request the document/_page.html.erb via AJAX and display the result on the web-page. It also initialises some of the other JavaScript functions when the AJAX content has been loaded.

- `src/app/assets/javascripts/documents_browse.js`
    - On page-ready the function ajax_loader() is called. This function is used to construct the Browse functionality of the LACR Search. It sends a request for a list of all page and volume numbers and generates the HTML lists and buttons used to display them. This function also appends event listener for the checkboxes to show their number as a “bootstrap badge”. On page-load also is added event listener for Download and Delete buttons. The delete functionality sends AJAX request to “ajax/doc/destroy” and displays the result as message notification utilising the note.js library.

- `src/app/assets/ javascripts/documents_selected.js`
    - On page-ready add event listener for the check boxes of all entries. When entry is deselected it is automatically removed them from the list.

- `src/app/assets/javascripts/home.js`
    - Used to initialise the JavaScript function on the home page.

- `src/app/assets/javascripts/init.js`
    - This JavaScript file is included globally at application.js. Its purpose is to combine all of the JavaScript functions and variables which should be accessible on every page.

- `src/app/assets/javascripts/search.js`
    - This JavaScript file initialises the fullpage.js library on the page which contains the search results. It also contains the implementation used for navigation between search result and the document page.

- `src/app/assets/javascripts/xml2html.js`

- This file contain the method used to switch between XML and HTML using small “XML” button in the bottom of each paragraph.

Stylesheets

- `src/app/assets/stylesheets/application.css`
  - Contains the CSS styles which are global (for all pages).

- `src/app/assets/ stylesheets/bootstrap_and_overrides.css`
  - Used to overwrite some the styles defined in Bootstrap.

- `src/app/assets/stylesheets/documents.scss`
  - Contains the styles applied only on documents views.

- `src/app/assets/stylesheets/home.scss`
  - Used for styles applied only on the home page.

- `src/app/assets/stylesheets/search.scss`
  - Used for styles applied only on the search views.