Implementing a paleo sea level Database with open source materials

Master Project

Marine Geosciences (M. Sc.)

Author: Jan Drechsel 2018

Supervisor: Dr. A. Rovere, SLCC @ MARUM

University of Bremen

https://github.com/jDrechsel/PaleoSeaLevelDatabaseInterface.git

Abstract

Software development Project to provide data management structure for field data with extensive metadata. This software was developed for the PALSEA template data structure, however it can serve as a prototype to handle other georeferenced point datasets and help to establish data standards. Using Open Source materials allows the software to be platform independent, modifiable and integrated into other frameworks.

Table of contents

Introduction

Understanding the history of sea level change is crucial for modelling future change to mitigate the drastic effects climate change will have on our society.

Calculations of relative sea level rely on several interconnected parameters, from local tectonics and sediment compaction over ice sheet history (both, melting ice and glacial isostatic adjustment) to interpretational uncertainties (Shennan, Long, & Horton, 2015). Thus, comparable datasets, even on a regional scale should be as comprehensible as possible in terms of data and metadata used. To reduce systematic and other errors, huge numbers of measurements of different but also the same indicators should be used to determine mean/more probable values. Extensive error metrics should also be provided.

(Düsterhus et al., 2016) proposed a structure to handle such large datasets: full disclosure description of **measurements** (location, age) and extensive quantification of uncertainties, followed by **interpretation**, including uncertainties and mentioning alternative interpretations. For this to work on a global scale, comparability must be guaranteed by providing comprehensible auxiliary data. (Khan, 2017) proposed a comprehensive data structure that is starting to be adopted throughout the PALSEA community, being very comprehensive, some navigation for the data-sheet will certainly be of help.

The main objective of this work is to close that gap by providing a graphical user interface (GUI) for adding new entries to the database and browse already published data. Furthermore, integration of visualization (map, plots) in the GUI allows easier selection of the desired entries.

Scalability, integration into other data frameworks, running on different machines (Computer, Tablet, etc.) and modification is provided by using open source material.

Methods

Approach

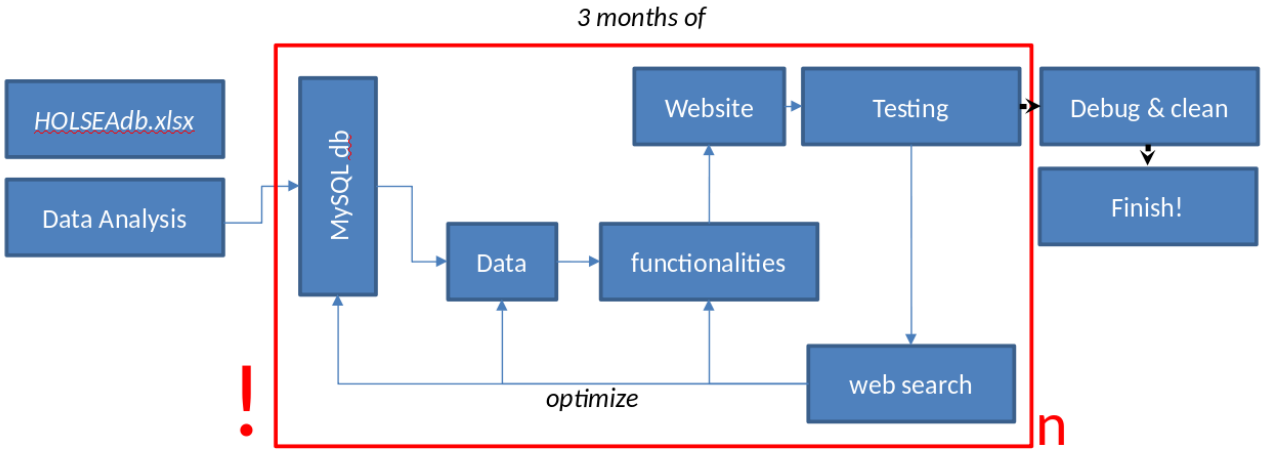

Figure 1: General strategy. Since web development skills are acquired along the way, some iterative approach was applied. With growing complexity of single elements, the architecture of the whole application had to be changed several times to incorporate some of the functionalities integrated later in the process

Since this project required some web development skills, some basics tutorials provided some insights prior to starting the project. The main goal was to implement an input form rendering the extensive data-sheet more manageable, and a simple data browser to export subsets of data.

Browsing the internet, some page designs drew the attention and were inspected for the frameworks used. Then the corresponding literature (mainly tutorials and stackoverflow.com) was consulted and the concepts were used where applicable.

Since some of the later changes required updating the general data structure, the workflow shown schematically in Figure 1, illustrates the 'iterative' nature of this approach.

Interface Design Concepts

The desired functionalities can be divided in two. First, the essential element is the communication with the database for uploading new and requesting already published data. Second, the data has to be visualized and made interactive. The browser to display entries was the first design concept, followed by integration of a map to browse data plotted in the geographical realm. The basic concept is shown in Figure 2, separated into static and dynamic pages: the form posts to the database where map and browser receive data from the database.

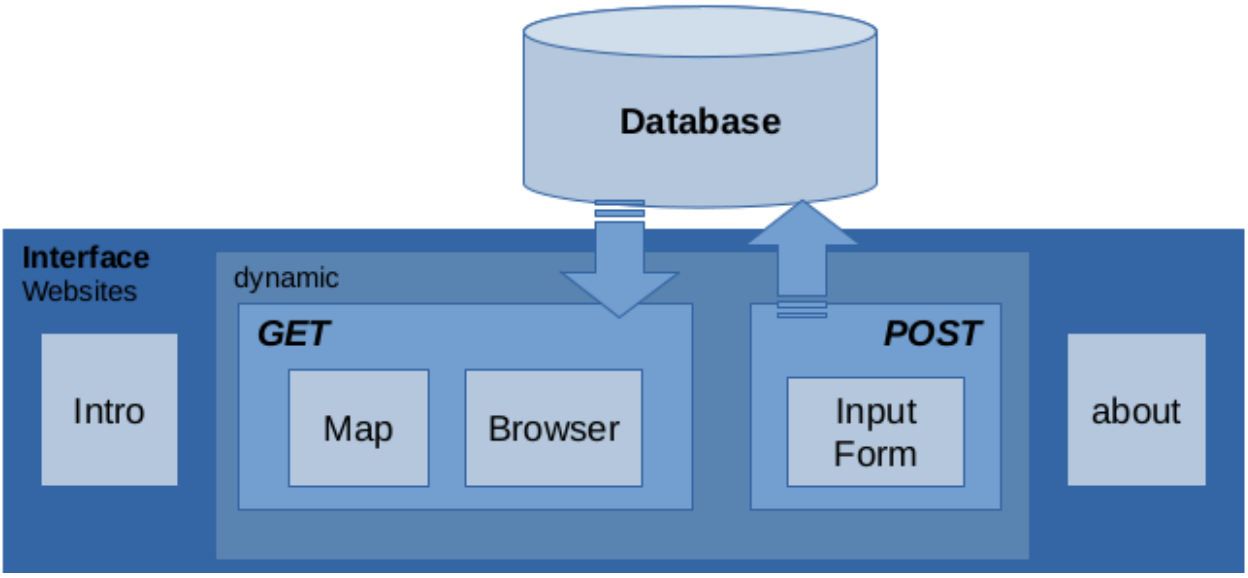

Figure 2: Basic layout of the website: five pages in total, three of them are dynamic, split in passive (GET) and active (POST). The two other pages are static pages containing additional information on the project.

Tools and Frameworks

The basic tools making up a webpage are **HTML** to define the content of a website, **CSS** to define layout and design of the html objects and **JavaScript** to define the behaviour of the pages. Additionally, **jQuery** was used to reduce the extent of JavaScript that has to be written by providing simplified syntax for common tasks.

In order to publish this site on different browsers and machines, the design must be responsive. **Bootstrap** is a HTML, CSS, and JavaScript framework providing responsive design

templates allowing the elements of the website to be adjusted to the respective screen or browser.

PHP is used to run scripts server-side, e.g. requesting data from external sources and prepare them for use within the page. PHP is the most used scripting language for this, it is implemented for example in WordPress and Facebook.

The database is stored in SQL format, a widely used database (ANSI & ISO standard). Here **MySQL** was used, running on an apache based server (xampp: www.apachefriends.org for local version)

The retrieved entries have to be converted to objects in a format suitable for JavaScript. This was done using **JSON** objects which are human readable and do not require XML parsing.

In order to build a dynamic web page, **AJAX** works behind the scenes to asynchronously exchange data between page and server. Furthermore, **AngularJS** was used to extend HTML vocabulary for the application, allowing data binding, custom controllers in plain JavaScript. Thus, the elements on the web site are generated by the data and its structure. E.g. on changing the grouping, HTML elements will change in the browser view. Additionally, AngularJS is of great importance for creating single page applications, crucial for implementing one single "shopping cart" for both, map and browser.

Visualization of data required frameworks compatible to AngularJS. (Touffe-Blin, 2017) provided an AngularJS library for **Chart.js,** a highly customizable, responsive JavaScript plotting framework. For plotting geographic data, (Rubert, 2015) developed an integration of **Leaflet maps** for AngularJS. Using these plotting libraries for AngularJS enables asynchronous visualization, enabling data in the plot to be based dynamically on user.

Chronology

Table 1: Project Chronology

25.06.18	Start
16.07.18	Basic functionalities (Form & MySQL Database) first test form with 20 fields connect form to dummy database style formatting Accordion
23.07.18	Full form with all entries adding placeholders (option shows full label but submits the number) reactive (showing only fields according to choices e.g. dating method)
06.08.18	Data explorer V1 (alongside input form)
27.08.18	Data Explorer V2 detail view list samples (left) table (right) select samples for export (checkboxes) Plot (left) array for plot *Indicator type → Array for series*
03.09.18	Export Dialog export plots fetch marked values of export array get their full entries from databases generate CSV file “save file” dialog
17.09.18	Map initial map prototype add markers from array selection rectangle

	mark for export
24.09.18	Combine export lists (“shopping cart”) changing page architecture to allow keeping the export array (AngularJS: ngRoute)
18.10.18	“Customer” feedback
15.10.18	cleaning, rearranging pages (start page), minor style changes clean-up code Debug Map add manual to “add Sample” page Reduce loading time by pre-allocating markers array
25.10.18	Upload (www.jdsandbox.000webhostapp.com)

Results

Overview

The functionality is explained schematically more detailed in Figure 3.

The datasets are retrieved by the server and processed into objects (arrays) with fitting formatting for the different tasks (map markers, data structure for grouping and sorting, and the export list). The website has different pages that work with one of the arrays respectively (1). User interaction (2) possible in the pages is selection of samples for export on the map and explorer page and revising them on the export dialog. Confirming the selection will request the full records corresponding to the elements in the export list (3). Additionally, an input form for adding samples can be found on the “add sample” page.

Architecture

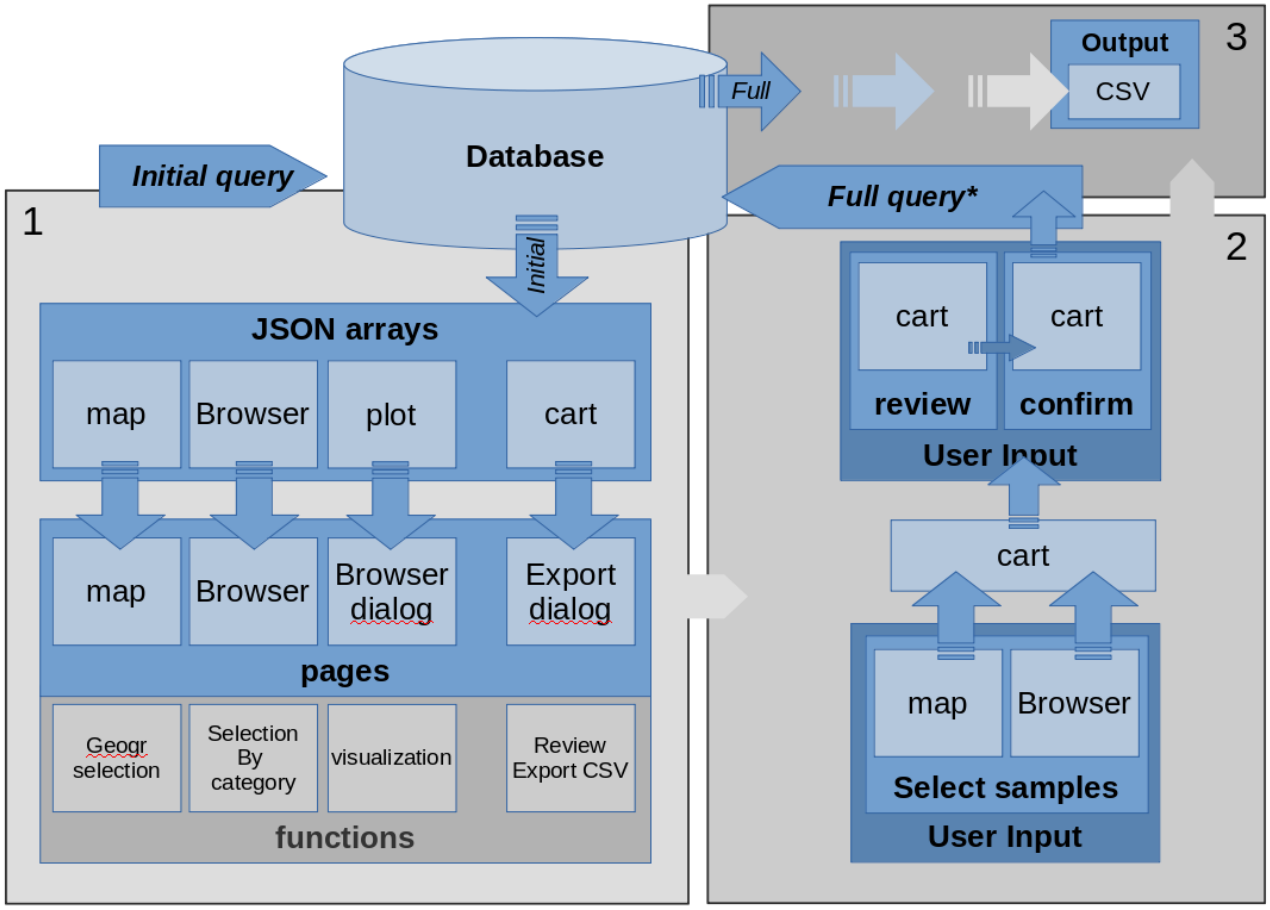

Figure 3: More detailed schematic of data handling: 1: an initial request retrieves the entries from the DB and creates different JSON arrays for each task (functions); 2: Selection of samples by the user either via map or browser, followed by review and confirm.

Detailed Data Handling

Optimizing performance is of essence for building interfaces for huge datasets. To achieve this, one initial request pulls all entries form the database into an object, which then can be further processed, filtered and sorted based on the selections made by the user on the interface (website). This results in the initial loading time being longer, but further operations can be carried out solely on the server, reducing their processing time to a minimum and allowing dynamic page updates.

Since the different pages need slightly different format of input, three JSON arrays are created, optimized for each task:

Map

Map markers for Leaflet need a different structure than in the two other arrays and do require geographic coordinates. This JSON array is only used in the map view to properly plot the sample locations as markers. Every sample has an additional property 'show' (true or false). This property is used to identify the samples to be displayed as markers on the map canvas.

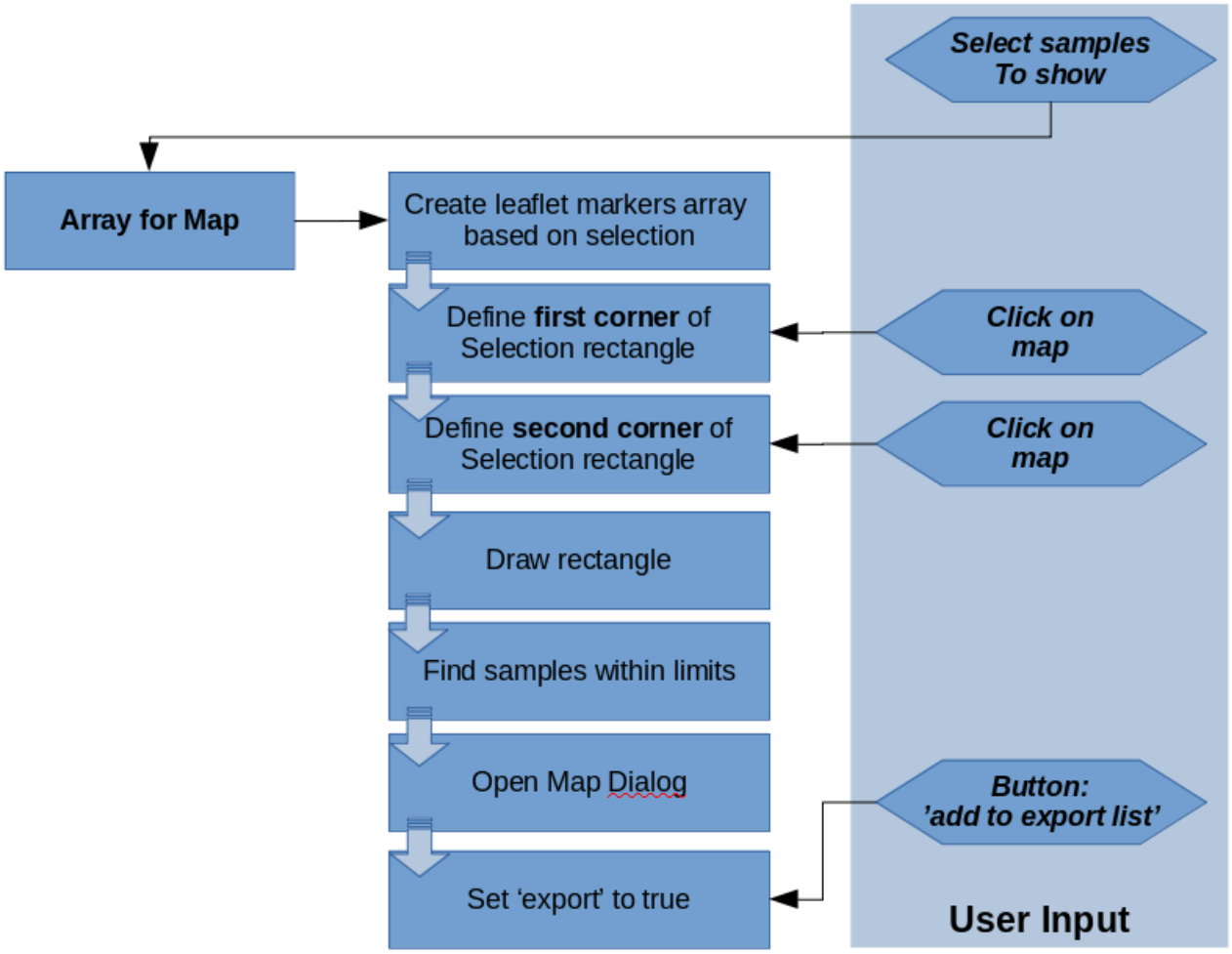

Figure 4: Schematic of data handling in map view: If a sample is selected the property 'show' will be set to true and all markers with show=true will be displayed on the map. Next, an area of interest rectangle (two clicks on the map) defines the area to search for samples with show=true. These samples are then displayed as a list on top of the map and can be added to the cart (show AND export = true)

The maps main purpose is to select samples by area of interest (drawing a rectangle). This is done by retrieving all samples which are tagged with 'show' = true and are located within the

limits of the selection rectangle. After drawing the rectangle, a small table opens to show some details on the selected samples, and a button that allows adding them to the export list by setting the corresponding property 'export' to be true (Figure 4).

Browser

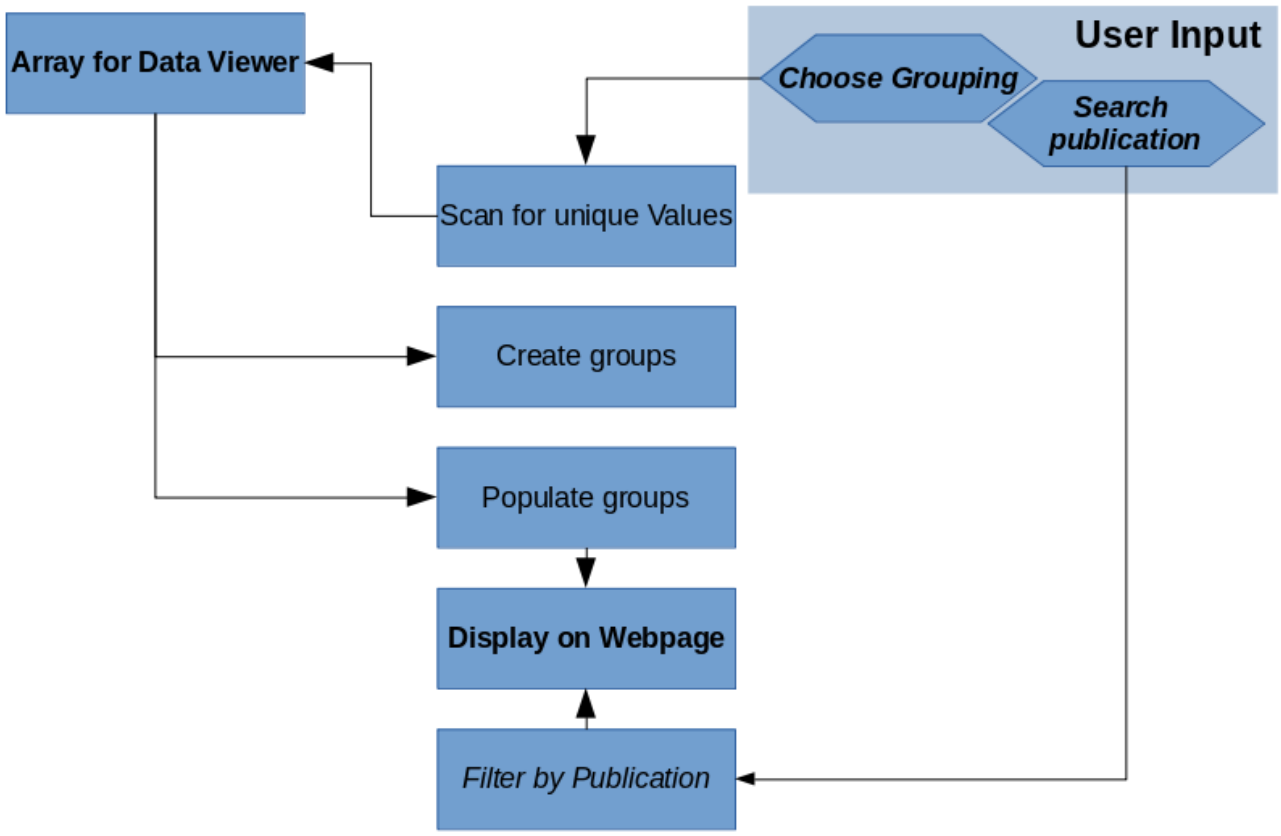

Figure 5: Schematic view of the data handling in browser view: grouping will search for unique values within the column, creates groups from them and populates with the corresponding samples. Additionally, the displayed groups can be filtered by publication (changes displayed data).

When a grouping is chosen a function iterates over the array searching for unique values in the respective column (Figure 5). From these unique values, groups are created and populated with the corresponding samples. AngularJS is used to generate HTML code to display one card per group showing the group name, number of samples in it, and providing a button to open the side panels, which contain information on each sample within the group.

Browser Dialog

The program iterates through each sample of the selected group, checking the value of Indicator Type. If the indicator equals zero (“Indicator”) the value of relative sea level will be used as y in the plot, otherwise sample elevation is used. An object of three arrays (0: Terrestrial, 1: Indicator, 2: Marine) containing the respective samples is generated to plot the three different series. On selecting to export, the property ‘export’ of sample with corresponding ‘UniqueID’ in the export list is set to true.

Export dialog

On opening the export dialog, the third array is used to determine samples to show in this view by extracting only samples where the property ‘export’ is set to true. The dialog allows reviewing selected samples similar to the browser dialog and exporting them as CSV when satisfied with the selection. On ‘export’, an array of unique sample IDs is created and used to request the full records of every ID in this array. The received data is converted to a CSV-file and can be saved with the following prompt.

User Access

All the data in the background provided, the result is an interface for the dataset. Samples can be browsed either with the map view or the browser, sample selection made in either is valid in the other. This chapter can be used as manual for using the interface.

Map

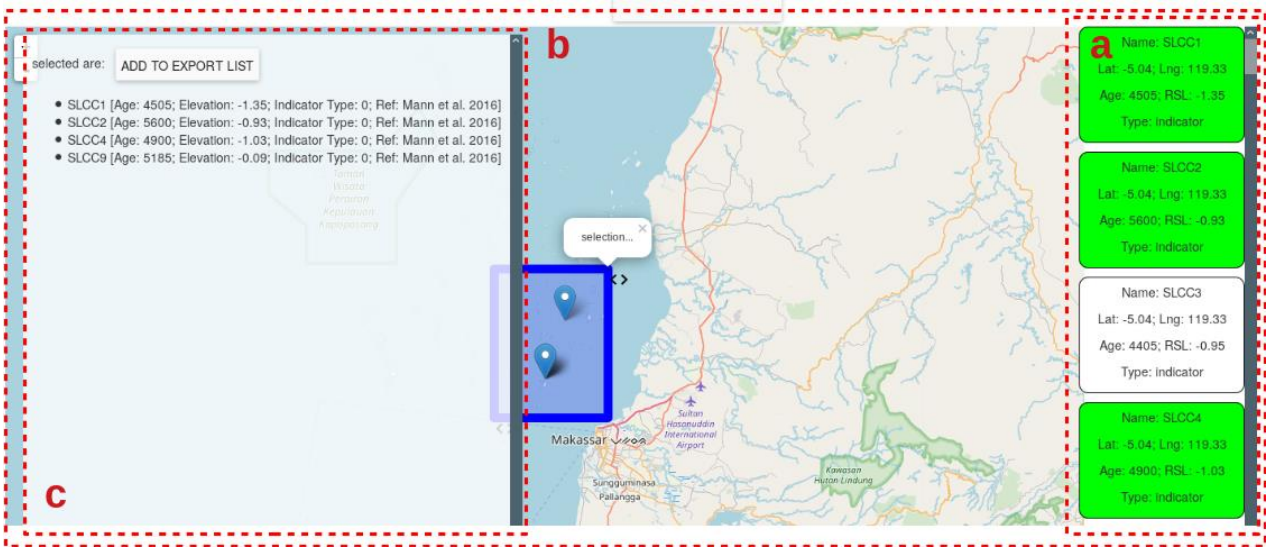

Figure 6: On the right-hand side, a list shows all available samples(a); by clicking on them they are added to the map (displayed samples have green background). With two clicks on the map (b), the bounding box for selection is defined. Afterwards, a small dialog opens on top of the map (c), allowing reviewing selected samples. The button 'add to export list' will place all the samples in the list into the cart (export list).

The map allows showing either all the samples at their respective location or based on the selection on the right-hand side of the map (Figure 6), and select them by defining a bounding box (blue) in order to add them to the export list. After selecting the corners of the box, a dialog appears on the left side of the map, allowing revising the selected markers. A click on the "add to export list" button changes the export list property "export: true" putting the selected samples into the 'shopping cart'

Browser

The browser is made up of three main parts (Figure 7): Sorting and Searching (Figure 7a), a dialog for plotting and marking for export (Figure 7d). The button to call the export dialog can be found on this page as well (Figure 7c).

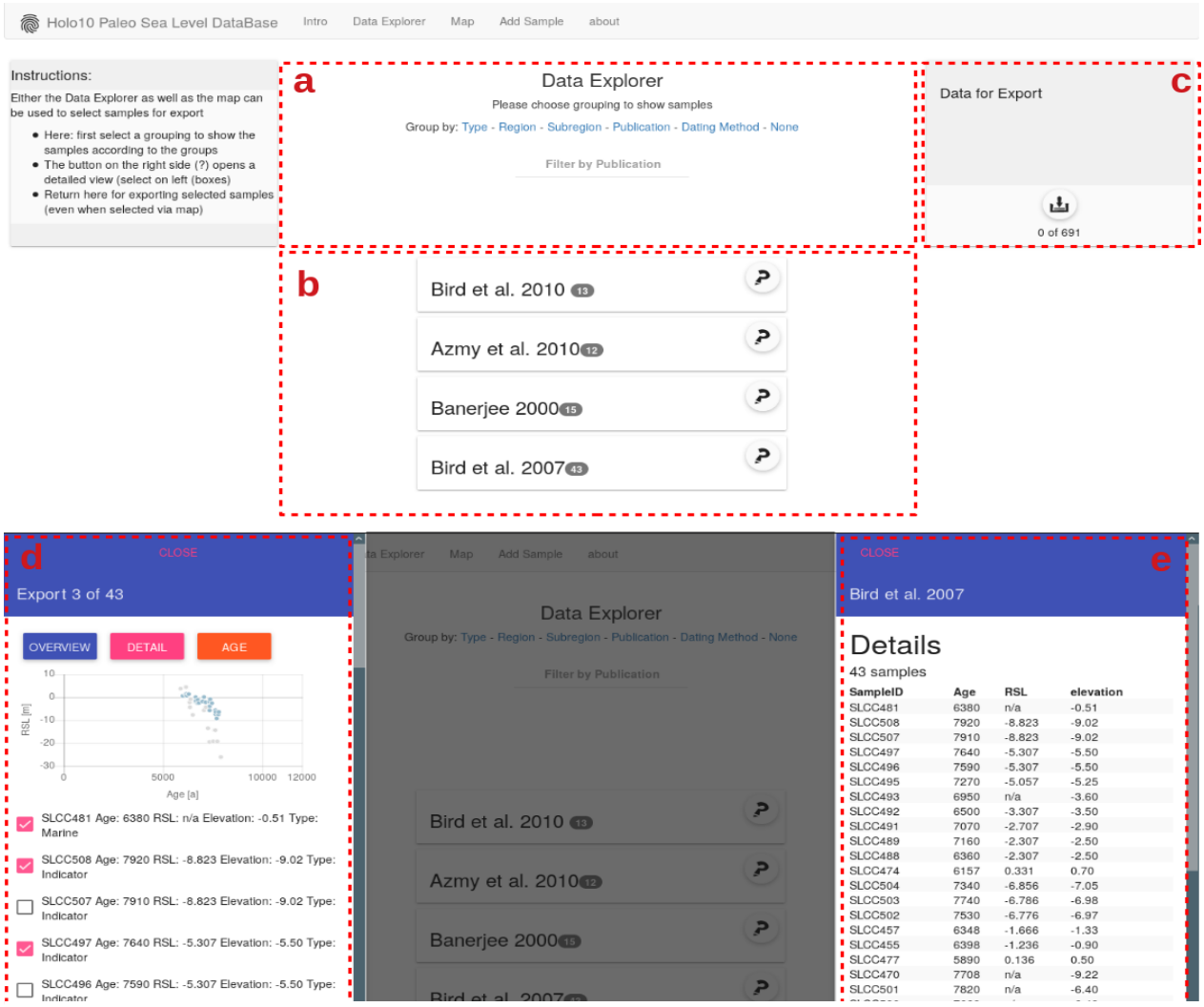

Figure 7: The browser page structures data based on different selectable parameters. In the middle the grouping and filtering can be selected (a). The groups based on the prior selection will be shown as cards below (b). On the right-hand side of each card is a button marked with '?', which will open the detailed view on the respective group. Two dialogs will open: Left (d), showing plots and selection options; Right (e), additional parameters of displayed samples). Furthermore, the button to open the export dialog can be found on this page as well (c).

In the Data viewer, the array can be sorted into six different categories:

Region	**Subregion**	**Publication**
Dating Method	**Indicator Type**	**None (for displaying all)**

The groups based on the respective choice will then be shown as boxes below the interface (Figure 7b). The button (marked with "?") on the right-hand side of each element will open the browser dialogs.

Plot and Selection

The button on the right of the groups card is bound to the samples within each group and opens two panels (left and right, Figure 7d, e) showing information on the samples.

To provide an overview of the samples within the selected group, the samples are plotted against age, with RSL as y value for indicators and Sample elevation for marine or terrestrial limiting.

The list of samples below the plot enables to select samples for export by checking the boxes on the left next to each sample. When a sample is marked for export, another property "export" is added to the sample object in the export list and set to true. The export function uses all samples in the export list with "export: true".

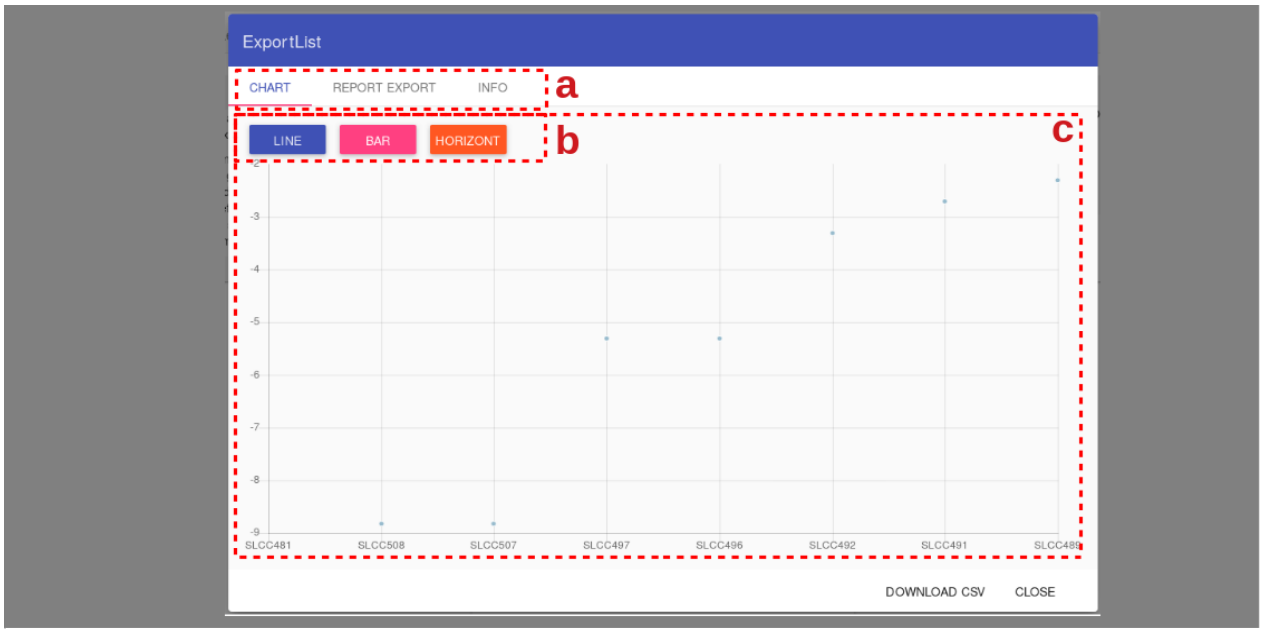

Figure 8: The export dialog allows to review selected data (c), either by investigating plotted data (first tab, a) or by checking a tabular view (second tab, a). Different plots can be selected using (b). Then the subset (selection) can be exported by pressing 'download CSV' or closed if adding or removing samples from the list is necessary.

Export

The cart can be found on the upper right-hand side of the browser page (close right panel). On Export another dialogue opens (Figure 8), allowing to review the selected samples in a more detailed view. The Message box has two main tabs (Figure 8a), the first (Chart) contains different plots (Figure 8b) for the selected samples. The first plot is similar to the one in the data explorer, showing elevations with age (same extents as explorer). The second plot can be used to have a look on relative values, since the extent will be cropped to the data extent. The last plot gives an overview over the ages, showing gaps (hiatus) in the record. The second tab (Figure 8a) will display the selected data as tabular form.

After checking the export list, press "generate CSV". When the button is clicked, a long form report is generated from the export selection by requesting the full records for each selected entry from the MySQL database.

If additional samples have to be added or some removed, simply close the dialog, either by clicking close in the lower right or clicking anywhere outside the dialog. The architecture of the website allows to jump between the different views until all samples desired are selected. As long as it is not reloaded manually, the export list will be stored. Even when the page for some reason seems slow, **do not reload the page**. The new initial request will overwrite the export list, rendering it empty.

Uploading new data

In order to add samples to the database, an input form is available (Figure 9). The main objective of this was to provide a clearly arranged input mask to facilitate data upload (Figure 9a). Additionally, CSV upload for multiple entries can be found on the upper right part (b).

The Form is designed to be reactive, meaning that only fields necessary will be displayed. This is the case for the dating submenu, where the selection of the first field "Dating Method" will decide which following fields will be displayed (e.g. U-Series will display just the field necessary

for Uranium dating and 14C will show the corresponding fields for radiocarbon). This will work with the checkbox rejected as well: on checking "rejected" a form field appears to give reasons for rejection.

Some values are calculated from the inputs from other fields (based on (Khan, 2017)), so no user input is necessary here, only in the fields on which the respective calculation is based. It will be populated as soon as all values necessary are given.

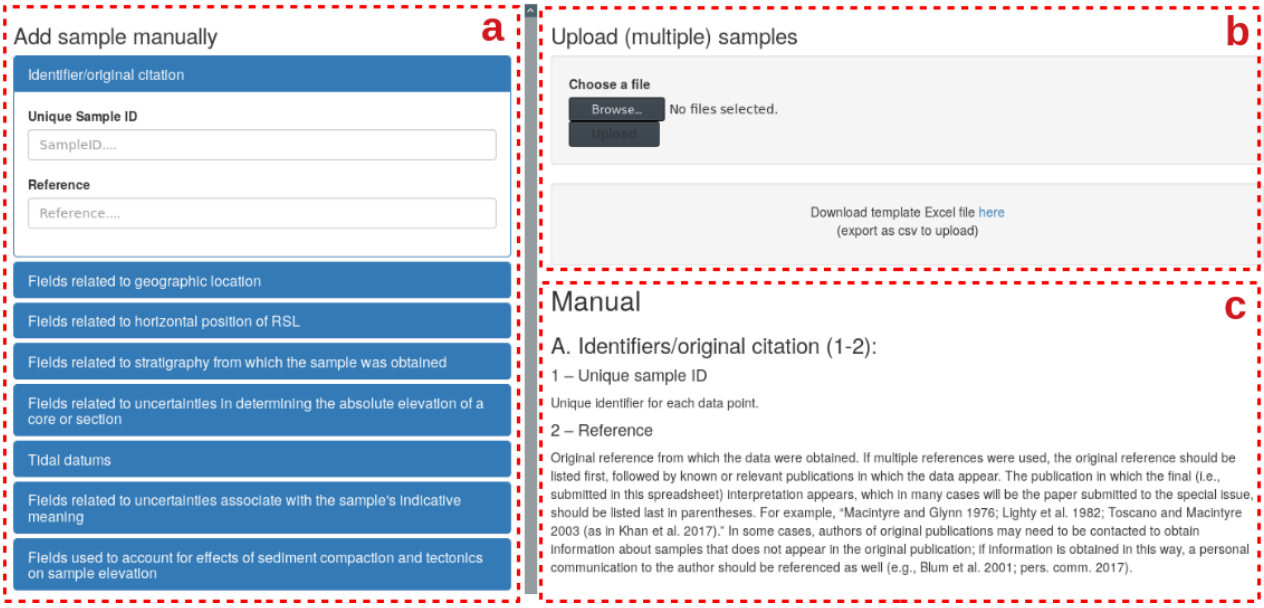

Figure 9: Upload form data: The upload form for new entries (a) is organized as accordion, so only one category can be opened at the same time to maintain simplicity/overview. Some of the fields are based on prior selections, e.g. dating method relevant fields will be displayed based on the choice of dating method. On the right-hand side, the options for uploading files (b) can be found as well as the manual (HOLSEA workbook) for populating the fields (c).

Online & Local Version

As reviewing the functionalities ("User testing") showed, one main problem with populating the data template is the complexity. To keep track more easily, an offline version of the input form is provided as well. This allows working on a private dataset until ready for publication. When ready, the dataset has to be exported from the local database as CSV and uploaded onto the online database using the "upload file" function. This approach also bypasses the need for

implementing account handling and securing log-ins, which would go beyond the time frame of this project.

Consequently, the interface has two versions: one online version containing published data without input fields to upload and a local version, which has all the functionalities of the online version and an additional input form for uploading sample data to a temporary local database. The main purpose is for every researcher working on paleo sea level data to have a private database until confident to publish them online.

Summary

This project was set out to facilitate data management for paleo sea level measurements, by providing a perspicuous interface, using only open source materials. This interface enables researchers to browse published data based on geographic location or other characteristics from the dataset. In order to further analyse the published data, samples can be exported as CSV file. Both map and browser allow to mark samples for export, enabling users to select subsets, e.g. area of interest or indicator type.

An input form was designed to provide a more intuitive method for populating the database. Due to security reasons uploading data to the online version, is restricted to the PALSEA data administrator. However, the local offline version allows working on a local database. Here, the input form can be used to populate a private database. In order to submit the private data, exporting to CSV and sending it to the current PALSEA data admin is best practice (for now).

Despite having to acquire the web development skills necessary for this project, this small specialized tool can help to establish a comprehensive paleo sea level database, by adding a coherent interface for researchers. Additionally, it might help to convince to use the data template of (Khan, 2017).

Outlook

In spite of its limitations, this tool could now go officially online, accessible for the paleo sea level community. An appropriate URL would make it more findable, by hosting the site on an official server, instead of free hosting.

Implementing user accounts could allow having one online version with private and public area, rendering the local version redundant. Further functionalities should be added, e.g. bibliography management, allowing generating e.g. BibTeX from exported samples. Another improvement could be expanding the filter functions, to enable to filter by age range or publication date.

This software has the potential to be integrated into other scientific data frameworks, since it makes use of only open source, widely applied solutions.

Since feedback already suggested that this tool will be appreciated by the community, further development should be realized. This can also allow having peer group testing reviews implemented into the new version.

Critical self-evaluation

With this project, I demonstrated that learning and applying state of the art web development could indeed reduce workload on extensive databases, by providing a more intuitive interface to the data.

I see this project as an introductory course in web and interface design. My skill set now includes creating databases with web browser connection and dynamic interfaces, as well as basic visualization techniques (plot, map).

My capabilities of adapting systematics of other structures and use them for my cause, helped tackling this project without having to take courses prior to the task. However, this approach

also bears some disadvantages, mainly missing some minor amount of the fundamentals in web development.

Due to my self-taught approach, I was not able to implement all of the desired features within the time frame. These shortcomings of the software mainly are security design in which I have full confidence, and I am sure, that the structural design behind the interface could be optimized as well. This is due to the fact, that with finishing this project I have learned more and more on the go and now would probably plan and start the project slightly different.

This of course leads to a better understanding on data management in the internet in general. Nevertheless, in order to be able to provide even better solutions, and better time management, it is crucial for me to take courses on web development.

Data structure and interface design now definitely has my attention, and I'm sure that with the necessary mending of basic and acquiring advanced programming skills, I will be able to provide a worthy share towards more intuitive (big) data handling in earth sciences.

References

Düsterhus, A., Rovere, A., Carlson, A. E., Horton, B. P., Klemann, V., Tarasov, L., … Törnqvist, T. E. (2016). Palaeo-sea-level and palaeo-ice-sheet databases: problems, strategies, and perspectives. *Climate of the Past*, *12*(4), 911–921. https://doi.org/10.5194/cp-12-911-2016

Khan, N. (2017). HOLSEA Data Template. *Personal Communication*

Rubert, D. (2015). angular-leaflet-directive. Retrieved from https://github.com/tombatossals/angular-leaflet-directive

Shennan, I., Long, A. J., & Horton, B. P. (Eds.). (2015). *Handbook of Sea-Level Research*. Chichester, UK: John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118452547

Touffe-Blin, J. (2017). angular-chart.js. Retrieved from https://github.com/jtblin/angular-chart.js

Appendix

According to Open Source ideology, full documentation, data, scripts and additional manuals can be found on

https://github.com/jDrechsel/PaleoSeaLevelDatabaseInterface.git

Installing the local version

Setting up the local database requires some preparations:

- First, xampp has to be installed to provide local servers (PHP and MySQL)

  https://www.apachefriends.org/index.html

- Second, the folder Holo10local (git?) has to be copied to the .../xampp/htdocs/

- Third, the database has to be created by loading the script "createDataBaseEMPTY.sql" (or "createDatBasePOP_name.sql") into phpmyadmin. The import tab can be used, to load the *.sql files. Working with phpmyadmin also allows to edit and export data. (However, less intuitive)

- Optional, the database can be populated with samples using one or more of the provided templates ("createDataBasePOP_name.sql") or by exporting the desired dataset(s) from the online version.

A more detailed manual on how to get the local version working can be found on the website and github. Feel free to contact the author (drechseljan@gmx.de or use the contact option on the about page), if there are some problems.