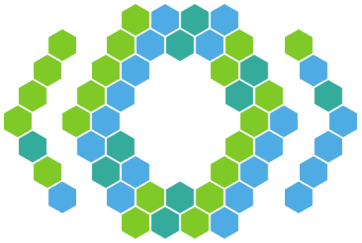

OPEN DATA PORTAL

Deliverable D8.3
Enhanced Portal Demonstrator

Autor: Afshin Amini
Reviewer: Adrian Wilke

Veröffentlichung	Öffentlich
Fälligkeitsdatum	30.06.2019
Fertigstellung	28.06.2019
Arbeitspaket	AP8
Typ	Software Documentation
Status	Final
Version	1.0

Summary:
The OPAL Portal will be used to search for different datasets based on their metadata which is crawled by Squirrel and enriched by several components such as CIVET. To achieve an appropriate user experience, a web user interface composed of server side computations and client side rendering is designed using ReactJS and NextJS. Multiple filters are provided to purify the large quantity of results. In addition, a SPARQL endpoint is also provided for technically experienced users. This document focuses on mockups of page views and used technologies for building the web UI.

Keywords:
OPAL Web UI, ElasticSearch, ReactJS, NextJS, Fuseki, Frontend Technology, Mockups

mFUND
Das Startkapital für die Mobilität 4.0

Bundesministerium
für Verkehr und
digitale Infrastruktur

Inhalt

Introduction 2

OPAL Web Portal Technologies 2

- **ReactJS** 2
- **NextJS** 2
- **Redux** 2
- **Spring-boot** 2
- **ElasticSearch** 3
- **Apache Jena Fuseki** 3
- **HTML5** 3
- **Bootstrap** 3

OPAL portal web user interface 4

- **Startpage view** 4
- **Single dataset view** 6
- **Generalized view on multiple data classes** 7
- **Generalized view on an individual data class** 8

1 Introduction

The OPAL portal is the prototype implementation of the open data portal, including comprehensive search functionality over the integrated metadata and visualizations, e.g., for quality metrics. As discussed in D8.1 there are different technologies for different purposes, and we should select those that fulfill our requirements in OPAL. The structure of the rest of this document is as follows: In Section 2 selected technologies with their benefits for being used in the OPAL are introduced, then in Section 3 schema of what will be designed as the user interface will be presented with a brief discussion on the elements in each page.

2 OPAL Web Portal Technologies

To fulfill the requirements of the OPAL Web Portal based on the existing technologies and frameworks, a technology review was conducted and the following technologies were selected.

2.1 ReactJS

ReactJS[1] is a declarative and component-based JavaScript library for building single page, and interactive user interfaces. It is open-source and revealed by Facebook. ReactJs is rapid in fetching data and updating the respective DOM elements, and also due to its component-based design it is a rich framework for building comprehensive search functionality that is required for the OPAL frontend solution.

2.2 NextJS

NextJS[2] is a Server Side Rendering (SSR) open-source framework that enables to have both SSR and CSR (Client Side Rendering) in a ReactJS application. It has some benefits such as

- Server-rendered by default
- Automatic code splitting for faster page loads
- Simple client-side routing (page based)
- Webpack-based dev environment which supports Hot Module Replacement

These characteristics speed up the general development and runs on the client side.

2.3 Redux

Redux[3] is an open-source javascript library for managing states. It supports om having all states in an object tree in a single store, that makes it more robust and bug free to use and pass data around different components, and enhances debugging processes of the application during development.

2.4 Spring Boo

Spring Boot[4] is used to generate Spring-based projects that are stand-alone and production-grade applications. By default it configured as a Spring project, including an embedded Tomcat Application Server, several POM templates which make the configuration of Maven simpler, and provides several metrics for monitoring the health of the project. The

[1] https://reactjs.org/
[2] https://nextjs.org/
[3] https://redux.js.org/
[4] https://spring.io/projects/spring-boot

deployment is more comfortable in comparison to conventional Spring projects, due to the usage of stand-alone projects.

2.5 ElasticSearch

Elasticsearch[5] is a distributed search engine based on Apache Lucene that provides full-text search engine over the data. To provide fast querying over metadata of data sets, Elasticsearch is a sophisticated search engine as it is well known for providing full-text search, and also it is equipped with some useful tools such as Kibana and Metrics that enables system administrators to monitor the systems health state and also study the data has been stored in it.

2.6 Apache Jena Fuseki

Apache Jena Fuseki[6] is a SPARQL server that contains a user interface for monitoring and administration. It is a sub-project of Apache Jena that provides an API to extract data from and write to RDF graphs and also supports SPARQL 1.1.

2.7 HTML5

HTML5[7] is the latest standard that defines HTML, and has been released in 2008. It is a W3C recommendation since 2014 for web development. It is widely supported in many devices such as all major desktop browsers and mobile browsers.

2.8 Twitter Bootstrap

Bootstrap[8] in and open-source CSS framework which supports latest CSS version (CSS3), and also is a mobile-first framework in which it makes sure to have a stunning and working web UI in different Desktop and mobile browsers.

[5] https://www.elastic.co/
[6] https://jena.apache.org/documentation/fuseki2/
[7] https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
[8] https://getbootstrap.com/

3 OPAL portal web user interface

The OPAL portal is composed of different pages with specialized views of components on each page. The startpage is the first page that user can see when browsing the OPAL portal website (Section 3.1). In addition, a view on individual datasets with details is provided (Section 3.2). For data classes, as licenses, publishers, or keywords, general (Section 3.3) and individual views (Section 3.4) are provided as separate page classes. In addition, embedded links to the SPARQL endpoint are integrated, which enable users to run individual SPARQL1.1 queries. The details of the individual page views are described in the following sections.

3.1 Startpage view

On this page, a search bar and related search domains are provided. The respective results of the search are displayed as a table with specialized filters (see Fig. 1). The result of a query is represented in a progressive way to not only user will experience a smooth browsing but also prevent an overload of browsers when there are enormous amounts of results (e.g. thousands). We integrated a “load 10 more” button that will get next 10 results of the given query. The total number of results are displayed in the top-left corner of the table view. Users are provided with the option to export selected results in different formats, i.e. CSV, TSV. By using a sorting by options users can order the result in different ways, such most relevant, most overall quality score, etc. The filter sidebar shows different filters on theme, keyword, which are used to refine search results. After selecting filters, updated results are shown to the user. For filters that have a lot of possible values (such as keywords), the results are shown in a progressive using a “load 10 more” button.

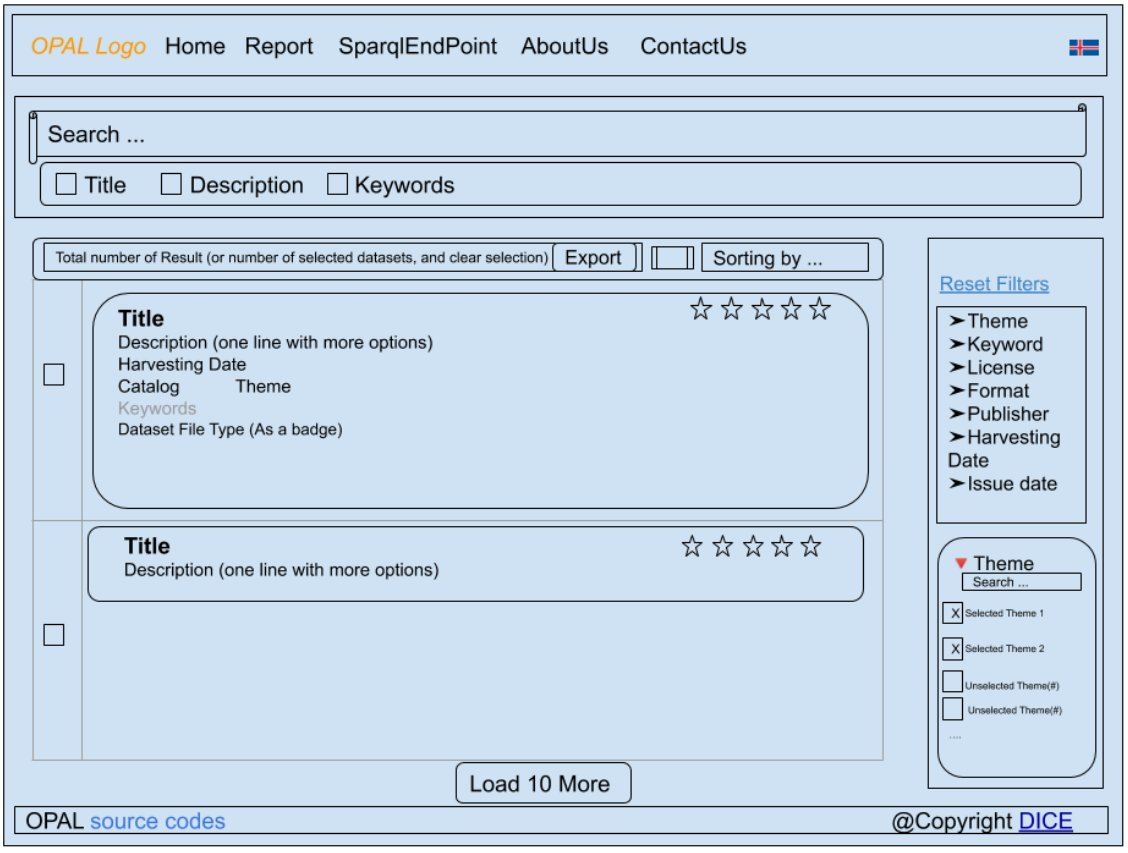

Figure 1: Mockup of the OPAL Home Page

A screenshot of the current state of the startpage is presented in Fig. 2:

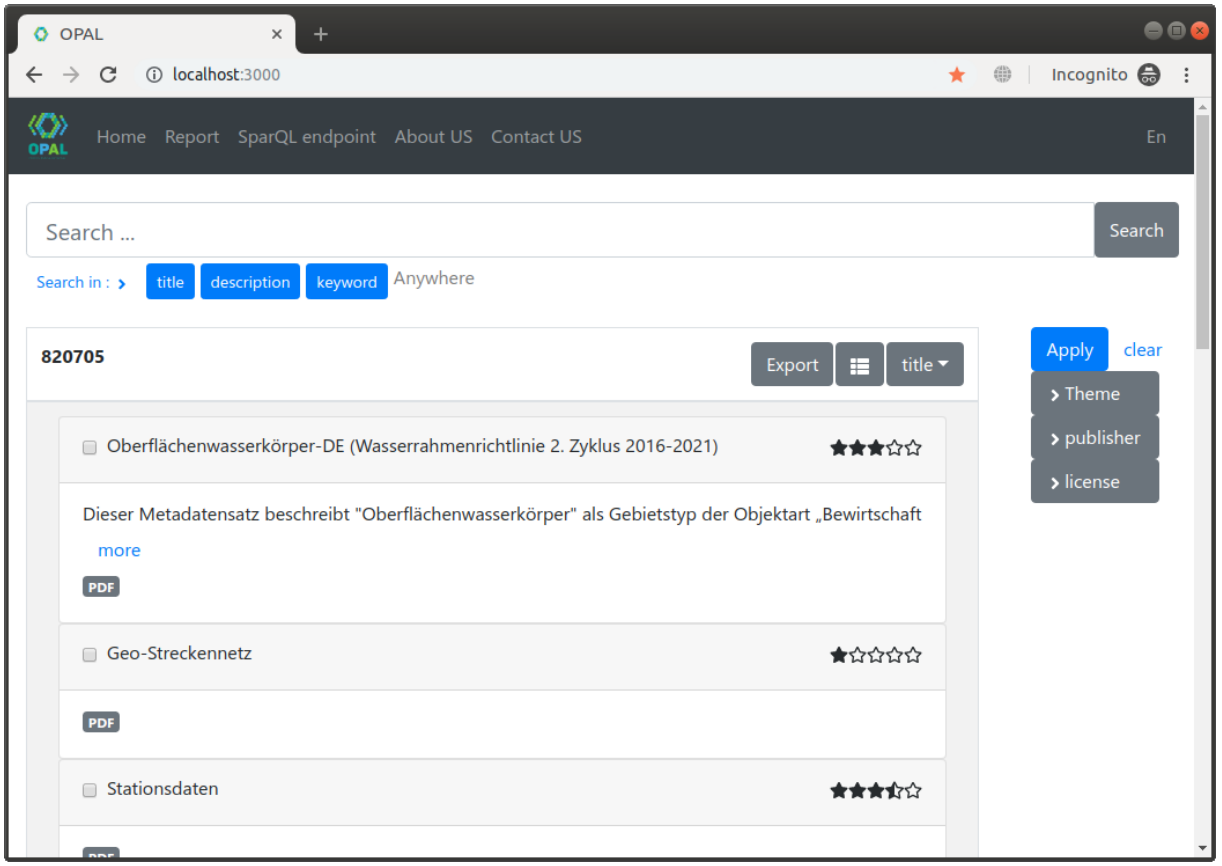

Figure 2: Screenshot of the OPAL Home Page

3.2 Single dataset view

To represent each dataset on a separate page type, its information and related datasets are displayed in a table (see Fig. 3). At the top, title and basic information of the selected data set will be shown. This is enhanced by meta information of related datasets, which is added in a table. Afterwards, the calculated meta data for quality of the data set in different aspects are shown in an additional section. Finally, related datasets are presented along with an additional ability for filtering.

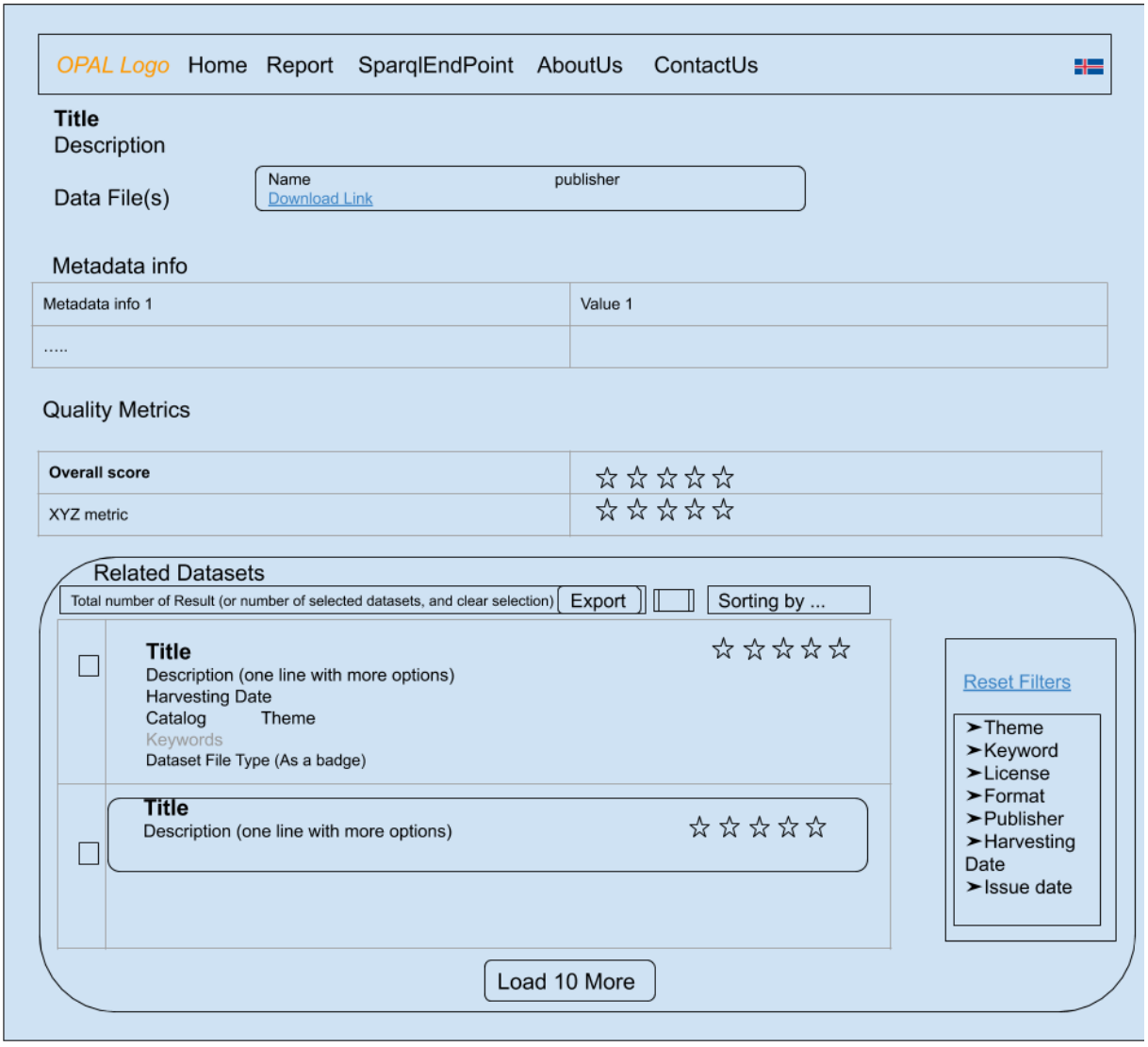

Figure 3: Mockup of the DataSet View

3.3 Generalized view on multiple data classes

Using this page view the user can see all the selected data class, such as licenses, keywords, etc. (See Fig. 4). The results along with the number of datasets for each row are represented. On click on an individual dataset, the users are redirected to the individual view of the selected one. As at the startpage view, for providing smooth browsing, and also prevent overload problems for huge amount of rows, results are displayed in a progressive way.

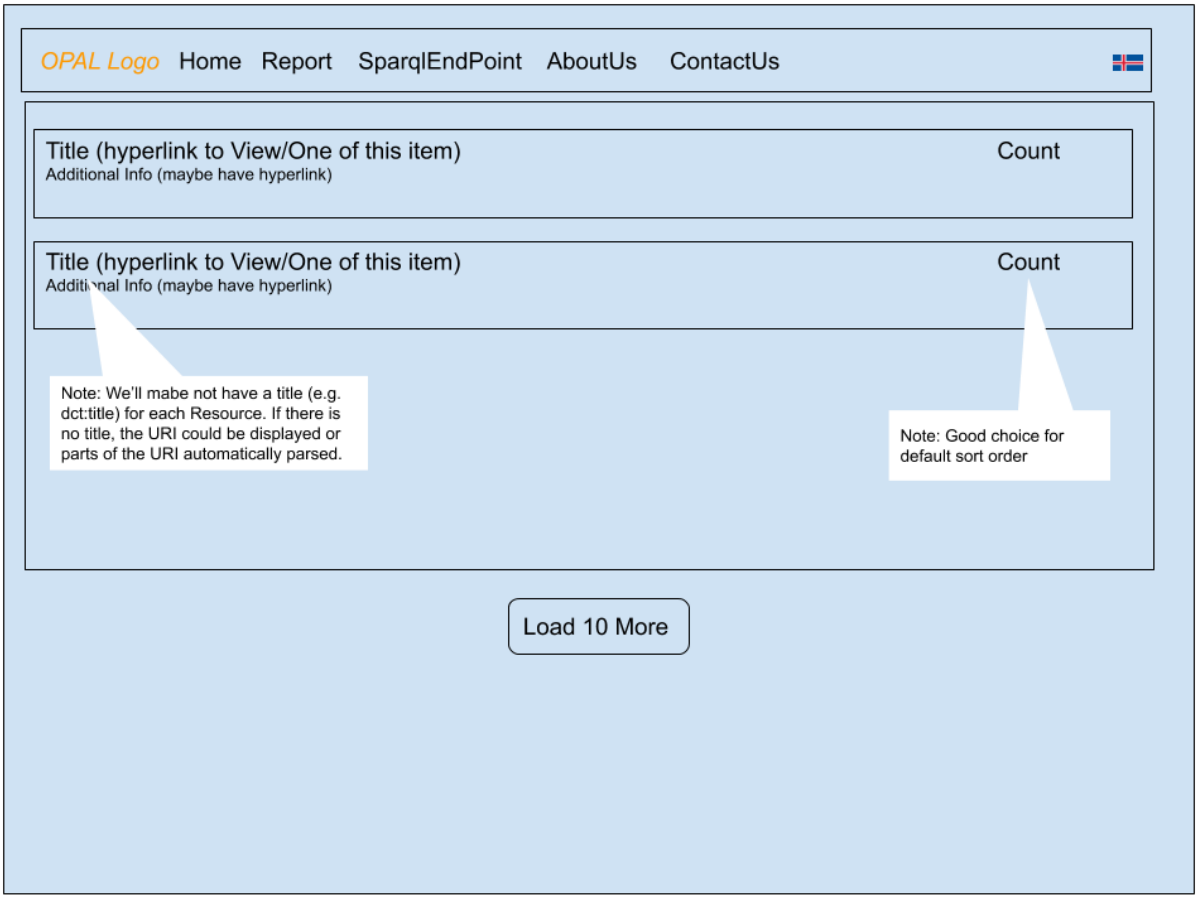

Figure 4: Mockup for the generalized view on multiple data classes

3.4 Generalized view on an individual data class

For each of the data classes, like license, keyword, or publisher, users are provided with independent pages. On top of these, the respective value followed by a table containing all of the data sets related to the value are displayed (see Fig. 5). The table view is the same as table view in the startpage, that is progressive for result, users are provided with an overview of the total number of datasets that contain the related value. In addition, users can order the datasets by selecting predefined sorting orders, and filter the datasets in the table based on different filters.

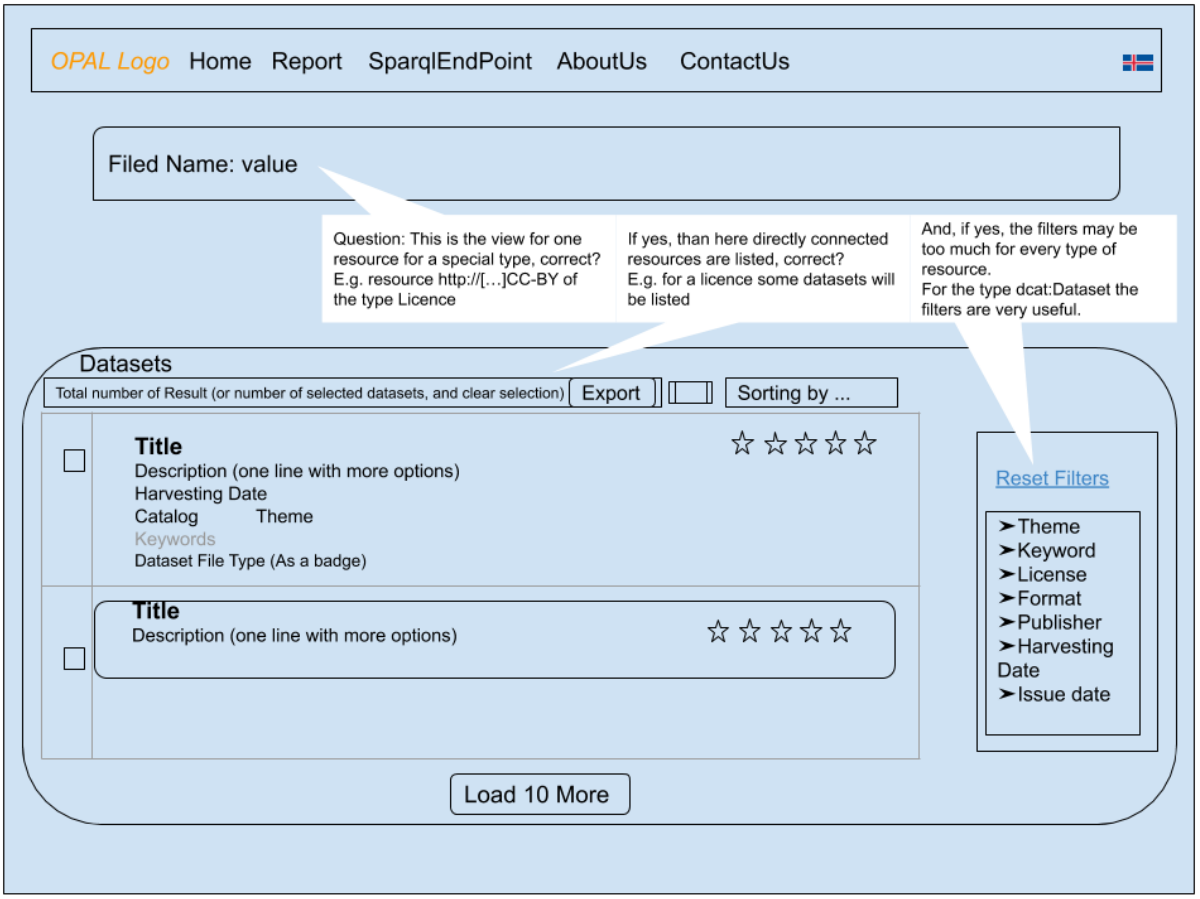

Figure 5: Mockup for the generalized view on an individual data class