Sampo-UI: A Full Stack JavaScript Framework for Developing Semantic Portal User Interfaces

Esko Ikkala [a,*], Eero Hyvönen [a,b], Heikki Rantala [a], and Mikko Koho [a,b]
[a] *Semantic Computing Research Group (SeCo), Aalto University, Department of Computer Science, Finland*
E-mail: firstname.lastname@aalto.fi
[b] *HELDIG – Helsinki Centre for Digital Humanities, University of Helsinki, Finland*
E-mail: firstname.lastname@aalto.fi

Abstract. This paper presents a new software framework, SAMPO-UI, for developing user interfaces for semantic portals. The underlying ideas of the framework are based on the "Sampo" model that contains a sustainable business model for publishing and sharing Linked Data based on a shared ontology infrastructure, the idea of providing the end-user with multiple application perspectives to the Linked Data, and a two-step usage cycle based on faceted search combined with ready-to-use data-analytic tooling. The framework draws from longstanding research of the Semantic Computing Research Group since 2002 on developing semantic portals, which has resulted in publishing a series of "Sampo" portals, mostly related to the Cultural Heritage domain, that have had millions of end-users on the Web. The SAMPO-UI framework makes it possible to create highly customizable and responsive user interfaces that satisfy the requirements for such portals using current state-of-the-art JavaScript libraries and data from SPARQL endpoints, while saving substantial coding effort. SAMPO-UI is published on GitHub under the open MIT License and has been utilized in several internal and external projects.

Keywords: Linked Data, Semantic portal, User interface, Web application, JavaScript, Software framework

1. Introduction: A Model for Semantic Portals

A fundamental underlying idea of the Semantic Web is to share Linked Data (LD) which is harmonized using ontologies. This approach has been proven useful in, for example, the Cultural Heritage (CH) domain in aggregating, harmonizing, and enriching data silos of different galleries, libraries, archives, and museums (GLAM) whose contents are typically heterogeneous and distributed, but often related semantically to each other [1]. Semantic Web content is published for machines via LD services and for human consumption using semantic portals. In general, semantic portals are information systems which aggregate information from multiple sources and publish them using Semantic Web technologies into user interfaces for solving information needs of end-users [2–6]. Such portals, based on knowledge graphs published in LD services, typically provide the end-user with "intelligent" services for data exploration [7] and analysis based on the well-defined semantics of the content. These services may include faceted, ontology-based, and entity-based search engines, semantic browsing based on semantic relations extracted and reasoned from the underlying knowledge graphs, and tooling for data-analysis, visualization, and serendipitous knowledge discovery [8].

To extend the notion of sharing and reusing data on the Web, this paper proposes that one should harmonize and share the way in which semantic portals, especially their user interfaces, are implemented and used, too. It is argued that in this way implementing semantic portals can be made easier for software developers, and from the end-user's perspective, portals based on similar functional logic are easier to learn to use. As

*Corresponding author. E-mail: firstname.lastname@aalto.fi.

an approach towards these goals, the "Sampo"[1] model has been found useful in practise while developing the "Sampo series" of semantic CH portals [8]. The model is based on the FAIR principles[2].

The "Sampo" model includes three parts that specify 1) the "business model" for creating and publishing heterogeneous, distributed LD, 2) the idea of providing the end-user with multiple application perspectives to the contents, and 3) how the application perspectives can be used in two basic steps using faceted search [9] for filtering out data of interest and applying data-analytics tools on it.

These three ideas have been explored and developed before in different contexts. The notion of collaborative content creation by data linking is a fundamental idea behind the Linked Open Data Cloud movement[3] and has been developed also in various other settings, e.g., in ResearchSpace[4]. The idea of providing multiple perspectives to a single database/service has been used in other portals, such as the ePistolarium[5] for epistolary data, and using multiple perspectives have been studied as an approach in decision making [10]. The two step usage model is used in prosopographical research [11] (without the faceted search component).

The novelty of the "Sampo" model comes from combining the three ideas into a whole. Furthermore, the "Sampo" model includes also the idea of using reusable data infrastructure and software tools for implementing the portals, especially the shared ontologies and datasets, such as the national LODI4DH infrastructure [12], and the SAMPO-UI framework for implementing user interfaces, the topic of this paper.

An example of applying the "Sampo" model and SAMPO-UI is the Mapping Manuscript Migrations (MMM) project [13], where three major databases containing information about pre-modern manuscripts were aggregated into a harmonized knowledge graph (data creation and publishing model), and semantic portal for manuscript research were built (multiple perspective interface design with two-step usage cycle). Figure 1 shows how one of the faceted search result visualizations provided by the MMM portal can be used to analyze the general trend of movement of over 200 000 manuscripts from their places of production into their last known locations.

The key contribution of this paper is to provide insights and a reusable solution to the following research question: *"How can user interfaces for semantic portals be built?"*. As a methodological basis, design science [14] is applied. General features and requirements of semantic portal user interfaces are outlined, based on the experience of designing and implementing the "Sampo series" of semantic portals since 2002 [8]. The new SAMPO-UI[6] JavaScript framework is our reusable tool for implementing user interfaces for semantic portals. The framework covers the second and third component of the comprehensive "Sampo" model, as illustrated in Figure 2. Hence, a prerequisite for using SAMPO-UI is that the underlying data is published as knowledge graph(s) in SPARQL end-point(s) using the principles of LD [15]. SAMPO-UI is not targeted for data curation, but for creating user interfaces for existing data.

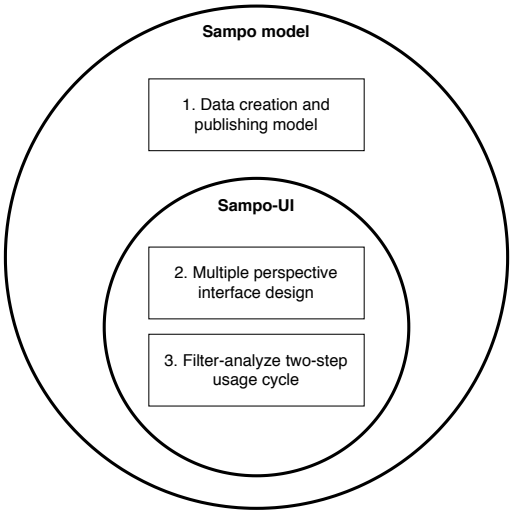

Figure 2. "Sampo" model and SAMPO-UI framework.

This paper is structured as follows. Section 2 reviews existing tools for developing user interfaces for semantic portals. In Section 3, based on experiences gathered in developing the "Sampo" series of portals, we outline a group of core features for semantic portal user interfaces, involving faceted search based

[1] "Sampo" is, according to the Finnish epic Kalevala, a mythical machine giving riches and fortune to its holder, a kind of ancient metaphor of technology.

[2] Findable, Accessible, Interoperable, and Re-usable, cf., https://www.go-fair.org/fair-principles.

[3] https://lod-cloud.net/

[4] https://www.researchspace.org

[5] http://ckcc.huygens.knaw.nl

[6] Information about SAMPO-UI can be found on the homepage https://seco.cs.aalto.fi/tools/sampo-ui; the framework is available under the open MIT License on GitHub: https://github.com/SemanticComputing/sampo-ui.

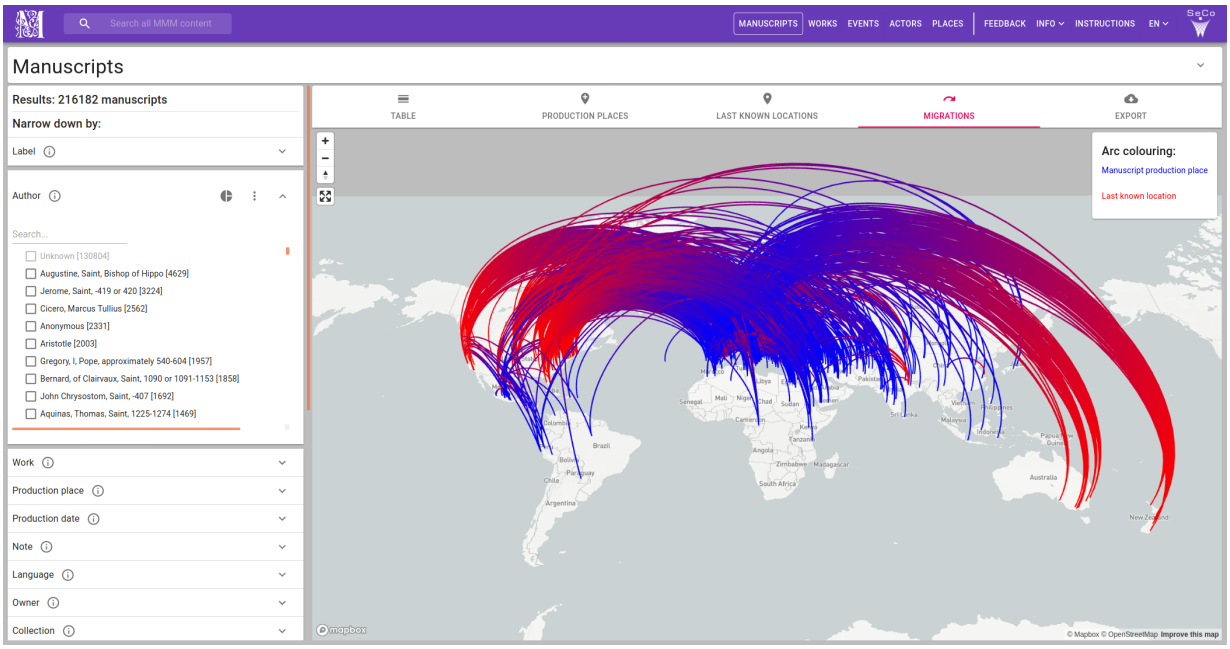

Figure 1. Visualization of the movement of pre-modern manuscripts as arcs from places of production to last known locations using the Mapping Manuscript Migrations Portal. Subsets of manuscripts can be filtered out for visualization using the facets on the left (Label, Author, Work, Production place etc.). The blue end of the arc shows the production place and the last known location is in red (this is not necessarily visible in black and white print). There are different options available for inspecting the filtered result set using tabs on top of the image: Table, Production places, Last knows locations, Migrations (selected in the image), and Export. The Export tab enables the user to export the SPARQL query, which was used for generating the Table result view, into the public SPARQL query service Yasgui. In Yasgui the current result set can be, e.g., downloaded in CSV format for external analysis.

on ontologies as a key ingredient. The new SAMPO-UI framework is presented and compared with existing approaches in Section 4. Experiences of applying SAMPO-UI are discussed in Section 5 through a case study on how the framework was used for building a new user interface of a semantic portal about military history. Finally in Section 6 the evolution of the ideas behind SAMPO-UI is described, after which the contributions, impact, and limitations of the framework are summarized.

2. Related Work for Semantic Portal User Interfaces

User interfaces for LD can be created in various ways, and can employ a variety of user interaction paradigms [16]. In addition to general user interfaces for LD, there is a large variety of custom, localized domain specific applications, tailored to cater users with information needs of a specific domain. This section provides some relevant references that in particular have inspired our work from a software developer's viewpoint; some related works about semantic portals from an end-user point of view were discussed in the previous section.

Khalili et al. [17] discuss the state of end-user application development for LD, and present the Linked Data Reactor (LD-R), which is an open-source software framework for building modern web-based LD user interfaces. LD-R is based on the idea of using existing state-of-the-art solutions, such as the Flux pattern[7] and the React library[8] to provide the software developer with a general "starting base" of a complete full stack JavaScript web application, which can be configured to view, browse and edit LD. The SAMPO-UI framework provides a similar "starting base" for a user interface, which is focused on faceted, ontology-based, and entity-based search interfaces, excluding data editing functionalities.

[7] Flux pattern for building user interfaces: https://facebook.github.io/flux

[8] https://reactjs.org

LD-R provides faceted search functionalities through FERASAT [18], which is a serendipity-fostering faceted browsing environment built on LD-R components and their configurations. The employed design principles: skeuomorphic[9], adaptive, and component-based, have been an inspiration to SAMPO-UI.

SPARQL Faceter [19] is a JavaScript library for building faceted search user interfaces, implemented using AngularJS[10] which is not anymore actively developed. SPARQL Faceter's ideas of using only SPARQL for data retrieval is also adopted in SAMPO-UI, and the other requirements found in the study have guided our work, which takes the same ideas further in a full stack setting, e.g., by providing tools needed for building a complete user interface for a semantic portal from scratch.

Bikakis and Sellis [20] have surveyed LD exploration and visualization systems, and presented some general features and requirements of such systems, which have informed our work. Klímek et al. [21] have surveyed existing tooling for LD consumption for non-technical end-users and presented general requirements for a hypothetical LD consumption platform. Many of the requirements are also relevant for domain-centric semantic portals and they have informed our research.

In addition to the SPARQL-based data handling needs of SAMPO-UI, the W3C's RDF JavaScript Libraries Community Group is creating standards and a collection of libraries for using LD on the Web[11]. SAMPO-UI uses an advanced object mapper to create JavaScript objects by combining SPARQL result rows, which is currently not supported in the aforementioned libraries.

3. Requirements for Semantic Portal User Interfaces

In this section desired features of semantic portals are outlined, based on experiences in developing the "Sampo" portals and from existing research. It has been found useful to provide the user with a variety of different ways to explore the knowledge graphs to find something useful and interesting without initially knowing exactly what to search for. Data exploration models provide the user with an overview first and iteratively enable exploring subsets of the data in more detail [20]. To this end, various search paradigms and interface models can be employed to provide different views to the data [22].

3.1. Search Paradigms and Result Set Visualizations

The search paradigms presented in the following paragraphs are found to be useful in many semantic portals.

Free text search across the whole knowledge graph is the most widely recognized search paradigm. The search can use all textual metadata fields of all entities, or to be narrowed down to, e.g., only use labels, and/or only use entities of specific classes. *Faceted search* [9, 23], known also as view-based search [24] and dynamic hierarchies [25], is based on indexing data items along orthogonal category hierarchies, i.e., facets (e.g., places, times, document types, etc.). In searching, the user selects in free order categories on facets, and the data items included in the selected categories are considered the search results. After each selection, a count is computed for each category showing the number of results, if the user next makes that selection, omitting categories with no hits. The idea of faceted search is especially useful on the Semantic Web where hierarchical ontologies used for data annotation provide a natural basis for facets, and reasoning can be used for mapping heterogeneous data to facets. Faceted search can be implemented with server-side solutions, such as Solr[12], Sphinx[13], and Elastic-Search[14], or in a LD setting using a set of configurable SPARQL queries, as in [18, 19].

In *geospatial search* the user can see resources on a map, and browse and explore them. It should be possible to filter the result set by making a selection on a map with e.g. drawing a bounding box. *Temporal search* can be performed by constraining the result set based on a timeline component or a timespan selector. A timeline can be used to browse and explore the resources. The result set that has been constrained using one or more search paradigms can be shown to the user in many different ways. The user can be provided with options of various result set display meth-

[9] *Skeuomorphism* in user interface design refers to incorporating recognizable, familiar objects to decrease the cognitive load of users and to hide the underlying complexity [18].

[10] https://angularjs.org

[11] http://rdf.js.org

[12] http://lucene.apache.org/solr

[13] http://sphinxsearch.com/blog/2013/06/21/faceted-search-with-sphinx

[14] https://www.elastic.co

ods to choose from, that are considered relevant: *Table* is often an intuitive way of displaying the resulting resources as a simple 2-dimensional table with each resource as a separate row and their most important properties shown as table columns. Another useful tabular format is to use a grid to position and show each result inside a rectangle. *Geospatial visualizations* show the results visually on a map as, e.g., markers, polygons or heatmaps. *Temporal visualizations* display the results on a timeline. *Spatio-temporal visualizations* display the results with interlinked geospatial and temporal components. *Statistical visualizations*, e.g., bar charts, histograms, line graphs, pie charts, and sankey diagrams, are useful if the result set is large, by providing summaries of the results instead of individual resources.

All of the previous display methods need to able to handle large result sets with, e.g., pagination, infinite scrolling, or clustering of individual resources into larger groups.

3.2. Interlinked Portal Perspectives to Data

The structure of the user interface should support having multiple perspectives for data exploration and searching, each of which are built around entities of a specific class. This way distinct perspectives to the underlying data can be built iteratively and individually.

Moreover, each entity of interest in the knowledge graph should have a landing page, which shows the metadata related to the single entity. Additionally, the landing pages provide both internal and external recommendation links to other related entities, e.g., for the landing page of a person, their family, relatives, and friends could be shown as links to the landing pages of the persons in question. Social network visualizations can be integrated to the landing pages of people, to provide a useful and intuitive way of seeing the person in their social and historical context. The URLs of the landing pages must be constructed in a in a systematic way to enable easy linking within the portal and from external applications.

4. SAMPO-UI Framework

This section presents the architecture and design principles behind the SAMPO-UI framework. More specific documentation and instructions for software developers can be found from SAMPO-UI's GitHub repository[15]. A case study of applying SAMPO-UI is presented in Section 5.

4.1. Architecture

The design philosophy behind SAMPO-UI differs from existing frameworks for LD user interfaces. Instead of focusing on providing the software developer with predefined configuration files and options as in, e.g., LD-R and SPARQL Faceter, in SAMPO-UI the center of attention is keeping the additional layer between LD specifics and underlying JavaScript libraries as thin as possible. This provides the developer with full control of writing and extending the actual client-side or server-side JavaScript code when necessary. However, predefined configuration files are used in SAMPO-UI whenever it's possible to employ them without restricting the developer from creating custom solutions.

These choices are based on the experiences of developing SAMPO-UI in conjunction with multiple LD based projects in different domains, where it has become evident that implementing user interfaces for semantic portals requires a remarkably wide range of flexibility to be able to adapt to the particular data models, schemas, search indices, and external APIs in use. Furthermore, as the data models get more complex and domain specific, it becomes virtually impossible to create universally applicable user interface solutions that could be taken into use by employing solely configuration files.

The SAMPO-UI framework provides software developers with a comprehensive set of reusable and extensible components, well-defined application state management, and a read-only API for SPARQL queries, which can be used for creating a modern and responsive user interface for a semantic portal with minimized coding effort. If the existing functionalities of SAMPO-UI are not enough, the architecture is designed to provide the software developer plenty of freedom to incorporate new functionalities by writing JavaScript code, utilizing the underlying libraries or adding new libraries. Much effort has been put in testing different open source libraries and choosing those with the best prospect for long-term sustainability as a basis for SAMPO-UI.

The framework is meant for building full stack JavaScript web applications. The main parts are 1) a

[15] https://github.com/SemanticComputing/sampo-ui

client based on the widely used and established React[16] and Redux[17] libraries, and 2) a Node.js[18] backend build with Express framework[19]. Angular[20] could have been another choice for a basis, but as it is a complete framework per se, choosing React as a basis for the client enforces a more modular structure for SAMPO-UI, in which individual libraries (e.g. for handling the application state or routing) can be replaced by new ones if they become obsolete. On the other hand, this modular structure demands more careful maintenance of the interplay of individual libraries.

Figure 3 depicts the overall architecture of SAMPO-UI. The general idea is that the focus of the client is on displaying data. The business logic of fetching the data from SPARQL endpoints using various search paradigms is placed on the backend. The client makes use of SPARQL end-point(s) by sending API requests to the Node.js backend. Based on the API requests and predifined configurations, the Node.js backend generates the SPARQL queries, sends them to the SPARQL endpoint(s), and maps and merges the raw results rows in the SPARQL 1.1 Query Results JSON Format[21] into a more developer friendly array of potentially nested JavaScript objects.

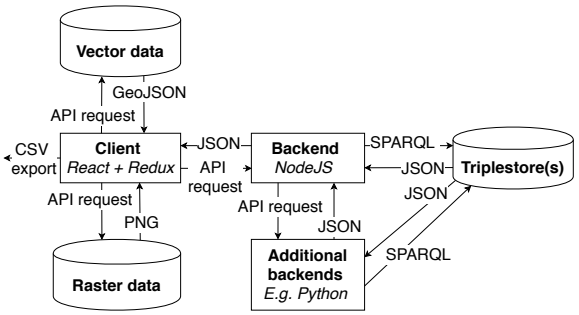

Figure 3. General architecture of SAMPO-UI.

The API endpoints provided by the Node.js backend are described using a document that conforms to the OpenAPI Specification[22]. The same document is used for both documenting the API, and validating the API requests and responses. The documentation is published with the API documentation tool Swagger UI[23]. The API documentation of an example portal[24] built with SAMPO-UI can used for testing the API.

For handling the API requests related to faceted, full text, and spatial search, the Node.js backend includes a set generalized templates[25] for SPARQL queries. In order to connect the Node.js backend to a specific SPARQL endpoint, the developer needs to provide configuration[26] for all desired facets and results sets. The placeholders in the SPARQL query templates are replaced with patterns from respective configuration objects by SAMPO-UI's core functions.

Introducing a Node.js backend to the architecture adds some overall complexity, but the benefits are plentiful. A significant part of the logic related to various search paradigms and the processing of search results can be carried out using pure JavaScript, instead of having to integrate the functionality inside client-side frameworks or libraries, which are known to be deprecated considerably sooner than pure JavaScript code. Additionally, using the backend for SPARQL queries makes it possible to run many computationally expensive operations related to data processing on the server, instead of relying on the varying computational resources of the client. Also querying password protected SPARQL endpoints without a backend is laborious and error-prone.

While the Node.js backend makes it possible to run any relevant JavaScript libraries on the server, the architecture can be extended by connecting the Node.js backend to additional backend services, as illustrated in Figure 3. This is useful, e.g., if Python modules are needed for dynamic processing of SPARQL query results. An example of such extension is the Sparql2GraphServer API[27] which integrates SPARQL queries and their results with the NetworkX[28] Python package for advanced network analysis tasks that are not currently available for JavaScript.

Figure 3 also shows how API requests to external vector and raster-based services for geographical data can be made directly from the client, as long as the re-

[16] https://reactjs.org
[17] https://redux.js.org
[18] https://nodejs.org/en
[19] https://expressjs.com
[20] https://angular.io
[21] https://www.w3.org/TR/sparql11-results-json
[22] https://swagger.io/specification
[23] https://swagger.io/tools/swagger-ui
[24] https://sampo-ui.demo.seco.cs.aalto.fi/api-docs
[25] Generalized templates for SAMPO-UI's core SPARQL queries: https://github.com/SemanticComputing/sampo-ui/blob/master/src/server/sparql/SparqlQueriesGeneral.js
[26] See instructions for configuring the Node.js backend for SPARQL endpoints in SAMPO-UI's readme: https://github.com/SemanticComputing/sampo-ui#configuration-and-folder-structure
[27] https://github.com/SemanticComputing/Sparql2GraphServer
[28] https://networkx.github.io

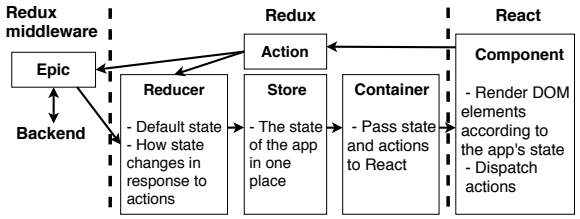

Figure 4. SAMPO-UI client architecture based on the strict unidirectional data flow of Redux integrated with React components.

spective API key does not need be hidden. If the API key need to be hidden, the API requests are routed through the Node.js backend. In SAMPO-UI it is also possible to export the results of the SPARQL queries in CSV format, a feature deemed useful by the end-users willing to analyse the data using additional software libraries and tools, such as spreadsheet programs and R. Additionally, the specific SPARQL query that was used for the search results can be given to the end-user if needed.

The modular architecture of the client is depicted in Figure 4, where boundaries between the main client-side libraries are marked with a dashed line. The state of the application (e.g., the user's facet selections and the search results) is maintained using the strict unidirectional data flow enforced by Redux[29]. To reduce the complexity of handling side effects, asynchronous data fetching is carried out uniformly by a Redux middleware named Redux Observable[30]. Redux Observable provides a base for SAMPO-UI's epics[31], which listen for asynchronous Redux actions, send API requests to the Node.js backend and other external services, implement debouncing when necessary, and handle the possible errors in the API responses.

4.2. *Adaptable Server-Side or Client-Side Faceted Search*

Nowadays the size of the knowledge graphs that a semantic portal needs to be able to process ranges from thousands to tens of millions triples. Furthermore, the data may be available from a single triplestore or from multiple triplestores. For handling these varying settings we have developed two main approaches for implementing faceted search in SAMPO-UI. We call the first approach "server-side faceted search" (ServerFS), where all queries relating to faceted search are sent to the triplestore. With this approach all limits regarding the size of the knowledge graph and the complexity of the queries are imposed by the triplestore and the server used for hosting it.

The most critical requirement for ServerFS is that all underlying data needs to be made available from a single SPARQL endpoint. ServerFS also causes considerable load on the triplestore when the knowledge graph is large or the portal has a usage spike[32], as a selection in one facet causes the recalculation of both the result set and the result counts of all of the active facets.

In some application settings, e.g. in NameSampo portal [27] that re-uses several existing SPARQL endpoints, it is not possible to aggregate all data into a single triplestore. For these settings we have developed a second approach named "client-side faceted search" (ClientFS). The main idea here is that the whole initial result set for faceted search is first fetched into the client as JavaScript objects, after which all faceted search functionalities are implemented using a set of Redux selectors[33].

Based on our experiences, due to the memory limit set by the browser running the client, the initial result set in ClientFS has to have an upper limit of approximately 30 000 instances on modern mobile devices or desktops with more than 4 GBs of RAM. Even though this limitation rules out many large knowledge graphs, we have found ClientFS effective especially when the initial query is a full text query, which is sent to multiple triplestores simultaneously. Then the results from each triplestore are merged and used as the initial result set for ClientFS. This approach is in use, e.g., in the above-mentioned NameSampo portal.

Figure 5 illustrates how the different tasks related to faceted search are divided between the client and the backend, when using either ServerFS or ClientFS. The example configurations of SAMPO-UI include templates for both ServerFS and ClientFS approaches. It is also possible to use the ServerFS and ClientFS approaches simultaneously in a single semantic portal, provided that they are separated into distinct perspectives. This is demonstrated in the example portal[34],

[29] https://redux.js.org

[30] https://redux-observable.js.org

[31] In Redux Observable epics are functions which take a stream of actions as input and return a stream of actions.

[32] If the triplestore is deployed using a Docker container or similar, it is possible to handle the usage spikes with autoscaling [26] carried out by a container-orchestration system.

[33] https://github.com/reduxjs/reselect

[34] https://sampo-ui.demo.seco.cs.aalto.fi

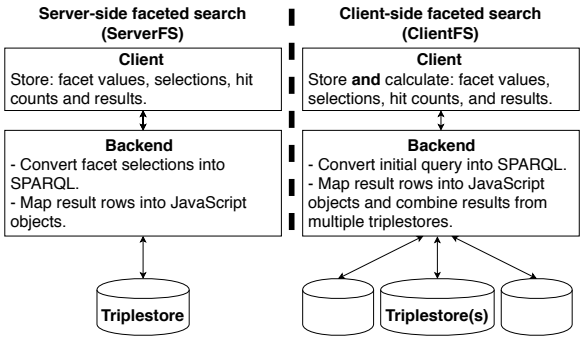

Figure 5. Faceted search architecture options of SAMPO-UI.

which combines perspectives from NameSampo and MMM portals.

Also several additional features for faceted search have been developed in SAMPO-UI, such as ordering of facet values, visualization of facet value distributions (e.g., by a pie chart), selecting multiple disjunctive values in a single facet, and support for arbitrary depth of hierarchies in hierarchical facets.

4.3. User Interface Components

Component-based user interface development is enforced by many state-of-the-art libraries and frameworks for web application development, such as React[35], Angular[36], and Vue.js[37]. By adopting component-based design principles, SAMPO-UI provides a generally applicable structure for a user interface of a semantic portal, while maintaining the possibility to create arbitrary layouts and structures. The responsiveness and accessibility of SAMPO-UI's components is achieved using the Material-UI library[38], which is a React implementation of the established Material Design language[39] developed and backed by Google.

Figure 6 illustrates how the user interface of a semantic portal can be broken down into a hierarchy of reusable components, inspired by the single-responsibility principle [28]. For simplicity, there are only three types of pages: *portal landing page*, *faceted search perspective* and *entity landing page*. All functionalities on these pages are provided by the main SAMPO-UI components shown in Figure 6. All relevant SAMPO-UI components are documented in a separate website[40], where the documentation texts are generated dynamically from the comments in the source code of SAMPO-UI, using the popular open source development tool Storybook[41].

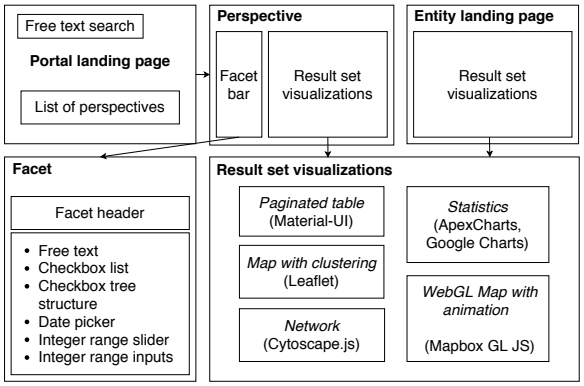

Figure 6. Main SAMPO-UI components for building semantic portal user interfaces. The underlying JavaScript libraries are given in parenthesis for the result set visualization components.

Because the underlying knowledge graph usually contains a diverse selection of differing entities, it has seen fit to divide the portal into multiple perspectives for different entity types. Each perspective typically contains faceted search functionalities configured for the entity type at hand, accompanied by a group of versatile views for visualizing the result set. An important design choice then is to find out which entity types have enough data points and significance to form a base of a perspective.

For example, in the MMM Portal[42], which uses ServerFS, there are five faceted search perspectives build around the following core entity types of the knowledge graph: Manuscripts, Works, Events, Actors, and Places. The ClientFS based NameSampo portal, on the other hand, is built around federated full text search results from multiple SPARQL endpoints, so the structure of the user interface is different. The main search functionalities, full text search and spatial bounding box search, are placed directly on the landing page of the portal. Once the federated query is finished, the user is directed to a single perspective, where the initial result set can be filtered further

[35] https://reactjs.org/docs/thinking-in-react.html
[36] https://angular.io/guide/architecture-components
[37] https://vuejs.org/v2/guide/components.html
[38] https://material-ui.com/
[39] https://material.io/
[40] https://semanticcomputing.github.io/sampo-ui
[41] https://storybook.js.org/
[42] https://mappingmanuscriptmigrations.org

with facets and analyzed using different result views. In both settings the same components for facets and result views shown in Figure 6 are used. This illustrates the reusability and extensibility provided by the component-based design of SAMPO-UI.

5. Case Study: Implementing the User Interface of WarVictimSampo

In this section experiences of applying SAMPO-UI to build user interfaces are presented and reflected using the development work for the user interface of the WarVictimSampo 1914–1922 portal[43] [29] as a case study.

5.1. Background for Developing the Portal

The WarVictimSampo portal was created to replace an older web application that was used to provide access to the War Victims 1914–1922 database[44] maintained by the National Archives of Finland. The database contains detailed information especially about the victims of the Finnish Civil War in 1918[45], including all known death records from that time due to the wars. The data has been used both by researchers of history and by the general public interested in finding out information about their deceased ancestors.

Simultaneously with the development of the new user interface, the database was updated with new victims and converted into a LD knowledge graph. The new portal needed to work with the knowledge graph and enable exploring, visualising, and studying the results in ways that are useful for both researchers and the general public.

The WarVictimSampo portal was opened to the public in November 2019 and gathered nearly 20 000 unique users in the first month. The portal, consisting of a user interface and a knowledge graph, was created reasonably quickly in one year with limited resources, so it was was preferable to use some existing application or framework as a basis for the user interface, instead of starting from nothing. SAMPO-UI was selected because it is based on modern currently maintained technologies, and offers a comprehensive "starting base" of a complete full stack JavaScript web application.

5.2. Setting up the Development Environment

The development started by forking the SAMPO-UI GitHub repository. SAMPO-UI provides a preconfigured environment for full stack JavaScript development. Babel[46] is used for converting the latest features of JavaScript, such as arrow functions and the async/await syntax, into a backwards compatible version of the language for current and older browsers. Webpack[47] handles the automatically restarting development server for the client, and bundling all source code and dependencies into static assets. The Node.js backend is run concurrently with the client, and is automatically restarted using Nodemon[48] when the source code is changed. Uniform coding style is enforced by using the JavaScript Standard Style[49] package.

The user interface of WarVictimSampo was developed on the basis of the default structure and configurations provided by SAMPO-UI, including the landing page of the portal, faceted search perspectives, and entity landing pages (cf. Figure 6). For faceted search, the ServerFS approach (cf. Figure 5) was selected, because the size of the knowledge graph is too big for ClientFS.

5.3. Multiple Perspectives to Data

The WarVictimSampo portal consists of two perspectives for studying the underlying knowledge graph, based on faceted search.

1. **War Victims** The main perspective of the portal includes information about some 40 000 war victims. The information can be searched and visualized in many different ways.
2. **Battles** The Battles perspective is based on the battles of the Finnish Civil War and was added to the application mainly for educational purposes. The perspective includes for example an animation of the battles.

The plan is to add more perspectives later, for example, about the prison camps of the Finnish Civil War.

[43] The portal is published at https://sotasurmat.narc.fi/en. See the homepage https://seco.cs.aalto.fi/projects/sotasurmat-1914-1922/en for more information about the project.

[44] http://vesta.narc.fi/cgi-bin/db2www/sotasurmaetusivu/main?lang=en

[45] The data contains also Finnish victims of the First World War and the Kindred Nations Wars in 1914–1922.

[46] https://babeljs.io

[47] https://webpack.js.org

[48] https://nodemon.io

[49] https://standardjs.com

5.4. Configuring Facet Components

The War Victims of Finland 1914–1922 database has very detailed information about the victims and a piece of information can relate to a person in more than 150 different ways. Different information types include both literal values in text and numeric form, and references to resources. For the best result different types of information require different types of facets.

In the WarVictimSampo portal the user can filter the initial result set with free text facets, date facets, numeric sliders, and flat or hierarchical checkbox facets. All of these facet types could be implemented in a straightforward manner by configuring and deploying the existing SAMPO-UI facet components presented in Figure 6.

5.5. Result Views: Table, Chart, Map, Animated Map, and CSV

After filtering out a result set it can be inspected using several visualization views as tables, pie charts, line charts, animations, and maps. In addition, the result set can be downloaded as a CSV file for external usage.

The paginated table view is the default view to the data and could be adapted easily by configuring the SAMPO-UI's table component. Figure 7 shows the table result view of the War Victims perspective with two facets opened on the left. The death date has been restricted to May 1918 using the Death date facet, and Helsinki has been selected from the Birthplace facet. This means that the current result set includes only those people who died in May 1918 and were born in Helsinki. The result set is shown on the right. First, on top, are the result view tabs that can be used to select which view is used to visualize the result set. Currently the table view is selected and highlighted. Below the result view tabs there is a navigation bar for controlling the pagination of the table. The actual table includes clickable column headings, which be can used to sort the table by each column. The upwards arrow next to the column heading "Name" means that the results are shown in ascending order based on the name of the person. The metadata of a person is represented as a single table row. If one table cell contains multiple values, they are rendered as list. The name of the person is a link that can be clicked to access the entity landing page of the person.

For statistical analysis, a pie chart result view was created for WarVictimSampo using the pie chart component of SAMPO-UI with minor customization. The pie chart can be used to visualize the distribution of distinct values within one facet. While the facets show the number of hits next to each selectable value by default, a pie chart visualization makes those numbers easier to understand. For example in this case pie chart can used visualize and compare the occupation distributions between the parties of the Civil War.

A line chart view based on the React wrapper[50] for Google Charts was created to visualize temporal distributions of death dates, ages at death, and birth years. The line chart view also calculates the average and median of the data values. This coupled with faceted search enables the user to find interesting new things in the data. For example, Figure 8 shows two different line charts generated by the line chart view for comparing the ages of people from one side of the Civil War whose official place of residence was in two different provinces of Finland. In both charts, the party "Red"[51] is selected from the "Party" facet. In the upper chart, Province of Turku and Pori is selected from the "Registered province" facet and in the lower one the Province of Viipuri. There is a big difference in the shape of the charts, and the median age at death for people from the Province of Viipuri is five years greater. Therefore, for some reason, members of the Red party from the Province of Viipuri died a lot older than those from the Province of Turku and Pori. Finding out this kind of serendipitous phenomena in the data for further analysis by close reading shows how faceted search combined with visualizations can be a powerful tool when researching the data.

The core structure provided by SAMPO-UI made it straightforward to connect the pie chart and line chart visualizations to facets and thereby make them interactive. The main challenge was converting and passing the data to the underlying visualization libraries in a form that they expect, which required some programming work.

SAMPO-UI includes a map component based on the Leaflet[52] library. The component was configured for WarVictimSampo to visualize the places of death

[50] https://github.com/RakanNimer/react-google-charts

[51] The Civil War was fought between socialist Reds ("punainen" in Finnish) and conservative Whites ("valkoinen").

[52] https://leafletjs.com

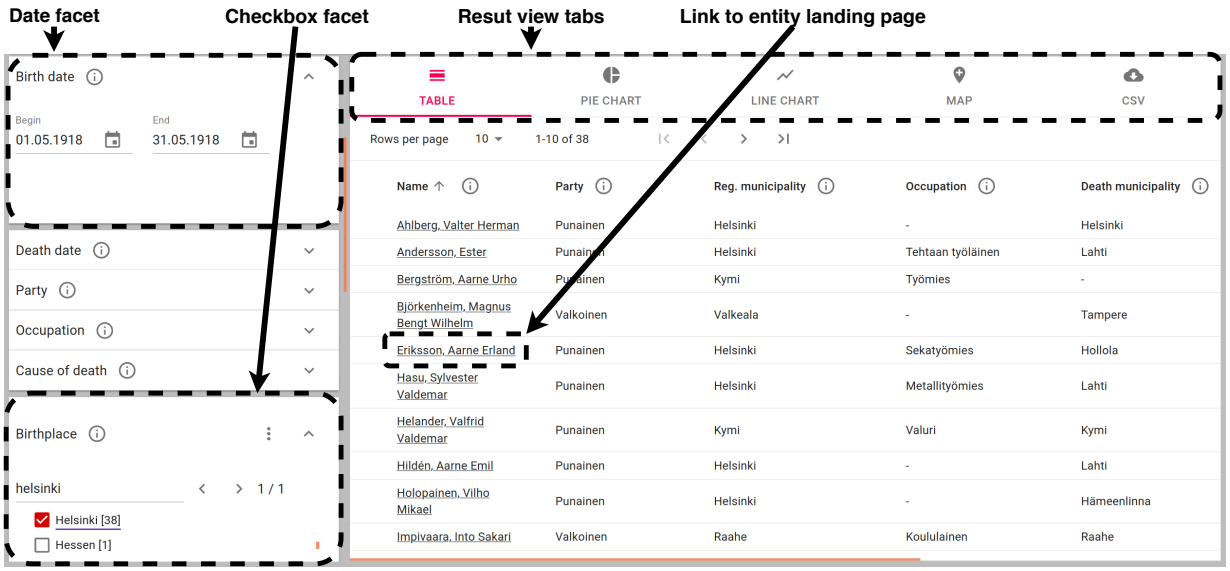

Figure 7. A faceted search view of the WarVictimSampo 1914–1922 portal. By using the facets on the left, the initial result set of some 40 000 war victims have been narrowed down into 38 victims who were born in Helsinki and died in May 1918.

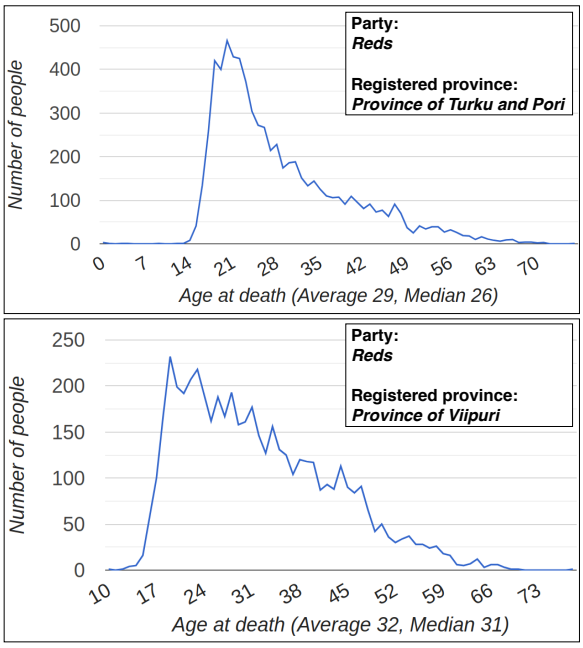

Figure 8. Two line charts generated with the WarVictimSampo portal depicting number of people of certain age at death in the result set. The upper chart depicts people of the Red party from the Province of Turku and Pori and the lower chart depicts people of the Red party from the Province of Viipuri.

as clustered markers on a base map fetched from the Mapbox Raster Tiles API[53].

For representing the course of events in the Finnish Civil War, a new animated map component was developed based on the WebGL-powered[54] deck.gl[55] framework. The animated map component presented in Figure 9 is used for the animation result view of the Battles perspective, where the temporal and spatial metadata of 1182 battles of the Finnish Civil War is rendered as an animation, with playback functionalities at the bottom of the map. The selected day is controlled using the time slider at the bottom. A battle emerges on the map as a red spot when the time-span of the battle is less than three days from the date of the animation. When the difference is larger, the spot remains on the map and turns to black. The spots can be clicked for examining the details of the battles. The animated map component was designed to handle any spatial data with a temporal dimension, so it was decided to add the component into SAMPO-UI's core components for visualising facet results.

The functionality for downloading the current result set of faceted search in table format as a CSV file was done with ready-to-use component of SAMPO-UI. The

[53] https://docs.mapbox.com/help/glossary/raster-tiles-api
[54] https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
[55] https://deck.gl

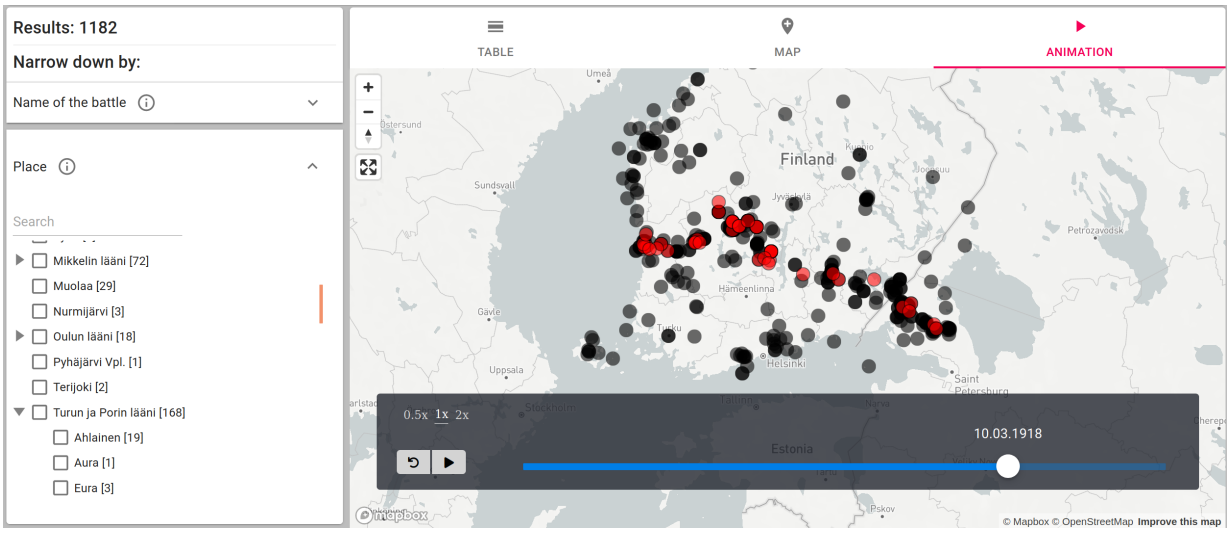

Figure 9. Animation result view of the battles of the Finnish Civil War. As time goes by on the time slider at the bottom, new battles emerge as red spots and turn later into black spots. In the image, the main front line of the Civil War from east to west can be seen. The battles can filtered using the facets on the left.

ability to download the data in this manner for external use in, e.g., spreadsheet programs, was a feature that especially the researchers of history required.

5.6. *Entity Landing Pages*

SAMPO-UI provides a generic functionality to create entity landing pages, which serve as the "homepages" for individual entities, such as people. The landing pages for battles were created using this functionality. As the metadata of the war victims contains over 150 distinct data fields, the generic landing page was extended to create a custom landing page for the war victims, including separate tabs for principal and additional information.

Expressing the sources for all the individual pieces of information was important for the research use of the knowledge graph, and it could be done by using a built-in functionality of SAMPO-UI. When the source for a piece of information is available, the respective component renders it in superscript with codes that can be clicked to navigate to the entity landing page of that specific source, which includes more information of the source. The entity landing pages of the sources were created using the SAMPO-UI's generic functionality for landing pages.

5.7. *Reflections of Using* SAMPO-UI

Using SAMPO-UI as the basis for a new user interface for a semantic portal saved resources and time, and demanded considerably less programming skill than would have otherwise been necessary. Some new client-side and server-side JavaScript code was needed for creating new types of result views to the data. Also, basic understanding of SPARQL and the data model of the underlying knowledge graph was required for writing property paths for the facet configurations and queries for result sets.

Responsive and user-friendly faceted search perspectives to the data could be created iteratively using SAMPO-UI's built-in components and APIs. The user interface can be easily extended, if new classes or properties are added to the knowledge graph in the future. In general, the WarVictimSampo portal has been received well by the researchers of history and can be considered a huge improvement in comparison to the old application.

6. Discussion

This section describes first how the ideas behind the SAMPO-UI framework have evolved. After this, contributions and limitations of SAMPO-UI are summarized and its availability and sustainability is discussed.

6.1. Portals Based on the "Sampo" Model and/or SAMPO-UI *Framework*

Table 1 presents the already published and forthcoming semantic portals that have been implemented using the "Sampo" model and/or SAMPO-UI. The portals categorized as *internal* in Table 1 have been developed by the Semantic Computing Research Group (SeCo)[56], while *external* portals, such as the portal for the Norwegian placename registry Norske Stadnamn[57], are examples of external adoption of the "Sampo" model and/or SAMPO-UI.[58]

The table also shows how the ideas behind the "Sampo" model and SAMPO-UI have evolved gradually since 2002. The idea of combining ontologies and LD with faceted search (that was called then view-based search) for semantic portals was first explored in the MuseumFinland[59] portal that aggregated and published heterogeneous, distributed collection data from several Finnish cultural history museums.

The idea of MuseumFinland was then extended in CultureSampo[60], where the goal was to try to represent and interlink virtually any kind of heterogeneous cultural heritage data, including both tangible and intangible cultural heritage, such as the Kalevala epic, and publish the massive LD using a semantic portal. CultureSampo explored the idea of providing the end-user with multiple perspectives to a knowledge graph[61]. The portal also advocated and demonstrated the idea of using a shared ontology infrastructure, and was based on the national FinnONTO ontology infrastructure[62] that was developed at the same time.

The BookSampo[63] portal, based on rich metadata about all virtually all Finnish fiction literature, was originally part of CultureSampo with a user interface of its own developed by the Finnish Public Library Consortium. This consortium maintains today BookSampo as a separate national library service, including the underlying knowledge graph.

Our work on creating generic user interface tooling for semantic portals started with the SPARQL Faceter tool [19] that was used in creating WarSampo[64] and BiographySampo[65] (and some other applications not included in Table 1). At this point, the idea on integrating semantic portals seamlessly with Digital Humanities tooling and serendipitous knowledge discovery was developed and demonstrated in practise [8]. The first version of SAMPO-UI was developed for the NameSampo portal [27] and has been used in the newer "Sampo" portals after that.

6.2. Contributions

This paper presented general features and requirements for user interfaces of semantic portals and the new SAMPO-UI JavaScript framework for developing such user interfaces based on the "Sampo" model. SAMPO-UI is our proposed solution to the research question *"How can user interfaces for semantic portals be built?"*.

The impact of the framework has been demonstrated through the case study presented in Section 5, the "Sampo" portals already using it, and the several new semantic portal user interfaces that are being built with it in 2020–2022, as shown in Table 1. Our practical experiences on developing several semantic portals in use suggest that the framework is viable and useful for software developers in implementing user interfaces for end-users. Obviously, user interfaces based on a similar functional model are easier to learn to use for the end-user.

From a technical viewpoint, several novelties and useful features are provided in SAMPO-UI for the software developer, such as:

- Ready-to-use components and an API for faceted search based on SPARQL endpoints are available, and also other search paradigms can be used.
- There is support for single and multiple SPARQL endpoints, including password protected SPARQL endpoints.
- Reusable and expendable data analytic views for studying the search results are provided. External

[56] https://seco.cs.aalto.fi

[57] https://toponymi.spraksamlingane.no

[58] More information about the "Sampo" portals and related publications can be found on the homepage: https://seco.cs.aalto.fi/applications/sampo.

[59] Publications and the link to the actual system online can be found at the project homepage: https://seco.cs.aalto.fi/applications/museumfinland.

[60] See homepage https://seco.cs.aalto.fi/applications/kulttuurisampo for further details and publications.

[61] Actually, the term knowledge graph was not in use in 2008 and the SPARQL standard was not available, but CultureSampo included a custom-made triplestore.

[62] https://seco.cs.aalto.fi/projects/finnonto

[63] See homepage https://seco.cs.aalto.fi/applications/kirjasampo/.

[64] See homepage https://seco.cs.aalto.fi/projects/sotasampo/en for further information.

[65] Cf. homepage https://seco.cs.aalto.fi/projects/biografiasampo/en

Table 1

Semantic portals using the "Sampo" model and/or the SAMPO-UI framework. Developing the model started with the MuseumFinland portal in 2002 and developing the SAMPO-UI framework began with the NameSampo portal in 2018. SAMPO-UI is use in five portals on the Web and in five forthcoming portals underway. An extended version of this table including links to the portals and their homepages can be found at https://seco.cs.aalto.fi/applications/sampo.

Portal	**Year**	**Domain**	**Unique users**	**KG triples or size**	**Faceted search engine**	**UI framework**	**Primary data owner**
			Internal portals				
MuseumFinland	2004	Museum collections	34 000	210 986	OntoViews	Apache Cocoon	National Museum of Finland
CultureSampo	2008	GLAM collections	89 000	11 400 000	Custom	Tapestry	Over 20 Finnish Memory Organizations
HealthFinland	2008	Health information	Unknown	6000 documents	Custom	Tapestry	Finnish institute for health and welfare
TravelSampo	2011	Cultural travel	Unknown	17 000 000 places	Custom	Drupal jQuery	Several Finnish Memory Organizations
WarSampo	2015	Military history	655 000	14 322 426	SPARQL Faceter	AngularJS	National Archives of Finland
BiographySampo	2018	Biographies	29 000	5 373 496	SPARQL Faceter	AngularJS	Finnish Literature Society
NameSampo	2019	Place names	35 000	241 068 456	ClientFS	Sampo-UI	Institute for the Languages of Finland and National Land Survey of Finland
WarVictimSampo 1914–1922	2019	Military history	21 000	9 815 265	ServerFS	Sampo-UI	National Archives of Finland
Mapping Manuscript Migrations	2020	Pre-modern manuscripts	2200	22 472 633	ServerFS	Sampo-UI	Schoenberg Institute for Manuscript Studies, Bodleian Libraries, and Institut de recherche et d'histoire des textes
LawSampo	2020?	Law	TBA	TBA	ServerFS	Sampo-UI	Ministry of Justice, Finland
AcademySampo	2020?	Finnish Academic People 1640–1899	TBA	TBA	ServerFS	Sampo-UI	University of Helsinki, Finland
HistorySampo	2021?	Historical events	TBA	TBA	ServerFS	Sampo-UI	Agricola Network of Historians
FindSampo	2021?	Archaeology and citizen science	TBA	TBA	ServerFS	Sampo-UI	Finnish Heritage Agency
ParliamentSampo	2022?	Parliamentary data	TBA	TBA	ServerFS	Sampo-UI	Parliament of Finland
			External portals				
BookSampo	2011	Finnish fiction literature	1 211 640 in 2019	8 313 263	None	Drupal	Finnish Public Library Consortium
Norske stadnamn	2019	Place names	Unknown	217 353 538	ClientFS	Sampo-UI	Norwegian Mapping Authority
Staff portal	2019	Human resources	Unknown	Unknown	ClientFS	Sampo-UI	Lingsoft Ltd.

geospatial data sources in vector and raster formats can be used.
- The framework supports multiple perspectives on a knowledge graph.
- The framework is highly customizable to multiple domains and use cases, and adaptable to different data models.
- The architecture of the framework is modular, and makes it possible to integrate new libraries and functionalities straightforwardly.
- By enforcing the Material Design guidelines as a base, the components support creating user friendly, responsive, and accessible interfaces for all screen sizes.

6.3. Limitations and Future Work

When designing user interfaces for semantic portals, one has to select which search paradigms are supported. A key challenge here is that the search paradigm selection affects the data creation phase and publishing phase, so that ontologies and metadata annotations support the planned user interfaces, and that the needed APIs and search indices are deployed and properly configured. Hence the functionalities provided by the SAMPO-UI components are dependent on the nature of the knowledge graph at hand, and the performance and configuration of the triplestore used for deploying it.

Since a user interface built with the SAMPO-UI framework is a full stack application with a client and a backend, SAMPO-UI cannot be included as an independent library, such as Leaflet, into an existing JavaScript application. Instead, the framework is meant to be used as a basis for a complete user interface of a semantic portal, including a predefined development environment, centralized state management, routing, and other relevant features. This in turn enables building a complete web application from scratch by forking the SAMPO-UI repository, followed by minor adaptations for the knowledge graph(s) in question.

The backend of Sampo-UI could also be used by another client via the well defined API. In order to further improve the reusability of SAMPO-UI, in the future the client and the backend could be refactored and separated into respective packages, and published in the Node package manager Registry[66].

As the API provided by SAMPO-UI's backend is read-only, functionalities related to editing and maintaining knowledge graphs are intentionally left out from the scope of the framework to avoid the complexity and security issues related to user management. Furthermore, supporting editing of LD while maintaining the integrity of knowledge graphs with wildly differing and complex data models, such as the CIDOC CRM[67] and its derivatives, is very challenging, as many of the data models for knowledge graphs are designed solely for presenting the data, not for editing.

SAMPO-UI is built on a group of external dependencies, which are listed in the standard *package.json* file. For keeping the framework up to date, these dependencies need to be updated periodically, while maintaining their mutual compatibility.

Several new features are being planned for SAMPO-UI, based on the requirements of different projects using it. One requested extension for faceted search is "faceted search within a facet", where, for example, instead of filtering the result set of books by authors, one could filter by author's birth date. This would require refactoring the core application state and components of SAMPO-UI so that each facet could have it's own filters for all relevant properties of the facet values. Another development trend is creating new visualizations of result sets. In contrast to modifying the facet components, they are easier to implement individually without touching the core parts of the application's state. For example, visualizing networks as interactive graphs or imprecise temporal information on zoomable timelines, combined with proper clustering algorithms for large result sets would certainly bring out new insights from already existing knowledge graphs.

6.4. Availability and Sustainability

The SAMPO-UI framework is being developed in a GitHub repository[68], which contains example structure, configurations, and documentation for building a complete user interface for a semantic portal from scratch. User interfaces built with SAMPO-UI are being developed in separate repositories, which are created by forking the SAMPO-UI repository. When new functionalities are added to the SAMPO-UI core repository, they can be merged into other respective repositories when needed, to facilitate code re-use. Dependency management and backward compatibility of

[66] https://docs.npmjs.com/using-npm/registry.html

[67] http://www.cidoc-crm.org

[68] https://github.com/SemanticComputing/sampo-ui

SAMPO-UI is ensured by using Semantic Versioning[69].

Sustainability of SAMPO-UI is supported by releasing the framework with the open MIT License, and indeed the software has already been used by external users with some contributions. The existence of the "Sampo" series of portals in use and the large number of quite recently published new portals (cf. Table 1) suggest that the underlying model is feasible and supports development work efficiently. Several new "Sampo" portals are being developed, which means that the SAMPO-UI framework will be supported and developed further by its developer, the Semantic Computing Research Group (SeCo) at the Aalto University and University of Helsinki (HELDIG).

Acknowledgments

The "Sampo" model, portal series, and the SAMPO-UI framework have been developed gradually by numerous researchers during several projects in 2002–2020 funded by over 50 different organizations.[70] The authors wish to acknowledge CSC – IT Center for Science, Finland, for computational resources.

References

[1] E. Hyvönen, *Publishing and Using Cultural Heritage Linked Data on the Semantic Web*, Synthesis Lectures on the Semantic Web: Theory and Technology, Vol. 2, Morgan & Claypool Publishers, 2012. doi:10.2200/S00452ED1V01Y201210WBE003.

[2] O. Suominen, Methods for Building Semantic Portals, PhD thesis, Aalto University, Finland, 2013. http://urn.fi/URN:ISBN:978-952-60-5254-0?locale-attribute=en.

[3] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer and Y. Sure, Semantic Community Web Portals, *Computer Networks* **33**(1–6) (2000), 473–491. doi:10.1016/S1389-1286(00)00039-6.

[4] N. Stojanovic, A. Maedche, S. Staab, R. Studer and Y. Sure, SEAL: a framework for developing SEmantic PortALs, in: *Proceedings of the 1st international conference on Knowledge capture*, K-CAP '01, ACM, 2001, pp. 155–162. doi:10.1145/500737.500762.

[5] E. Hyvönen, Semantic Portals for Cultural Heritage, in: *Handbook on Ontologies*, 2nd edn, S. Staab and R. Studer, eds, Springer, Berlin, Heidelberg, 2009, pp. 757–778. doi:10.1007/978-3-540-92673-3_34.

[69] https://semver.org

[70] Some 400 publications related to this line of research and development are available at http://seco.cs.aalto.fi/publications.

[6] H. Lausen, Y. Ding, M. Stollberg, D. Fensel, R. Lara Hernández and S.-K. Han, Semantic web portals: state-of-the-art survey, *Journal of Knowledge Management* **9**(5) (2005), 40–49. doi:10.1108/13673270510622447.

[7] S. Idreos, O. Papaemmanouil and S. Chaudhuri, Overview of Data Exploration Techniques, in: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, Association for Computing Machinery, 2015, pp. 277–281. doi:10.1145/2723372.2731084.

[8] E. Hyvönen, Using the Semantic Web in Digital Humanities: Shift from Data Publishing to Data-analysis and Serendipitous Knowledge Discovery, *Semantic Web – Interoperability, Usability, Applicability* **11**(1) (2020), 187–193. doi:10.3233/SW-190386.

[9] D. Tunkelang, *Faceted search*, Synthesis lectures on information concepts, retrieval, and services, Vol. 1, Morgan & Claypool Publishers, 2009, pp. 1–80. doi:10.2200/S00190ED1V01Y200904ICR005.

[10] H.A. Linstone, Multiple perspectives: Concept, applications, and user guidelines, *Systems practice* **2**(3) (1989), 307–331. doi:10.1007/BF01059977.

[11] K. Verboven, M. Carlier and J. Dumolyn, A short manual to the art of prosopography, in: *Prosopography approaches and applications. A handbook*, Unit for Prosopographical Research (Linacre College), 2007, pp. 35–70. doi:1854/8212.

[12] E. Hyvönen, Linked Open Data Infrastructure for Digital Humanities in Finland, in: *Proceedings of the Digital Humanities in the Nordic Countries (DHN 2020)*, CEUR Workshop Proceedings, 2020, Accepted.

[13] E. Hyvönen, E. Ikkala, J. Tuominen, M. Koho, T. Burrows, L. Ransom and H. Wijsman, A Linked Open Data Service and Portal for Pre-modern Manuscript Research, in: *Proceedings of the Digital Humanities in the Nordic Countries 4th Conference (DHN 2019)*, CEUR Workshop Proceedings, Vol-2364, 2019, pp. 220–229.

[14] A.R. Hevner, S.T. March, J. Park and S. Ram, Design Science in Information Systems Research, *MIS quarterly* **28**(1) (2004), 75–105. doi:10.2307/25148625.

[15] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, 1st edn, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, 2011. doi:10.2200/S00334ED1V01Y201102WBE001.

[16] C. Bizer, T. Heath and T. Berners-Lee, Linked Data - The Story So Far, *International Journal on Semantic Web and Information Systems* **5**(3) (2009), 1–22. doi:10.4018/jswis.2009081901.

[17] A. Khalili, A. Loizou and F. van Harmelen, Adaptive Linked Data-driven Web Components: Building Flexible and Reusable Semantic Web Interfaces, in: *The Semantic Web. Latest Advances and New Domains. ESWC 2016.*, H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S.P. Ponzetto and C. Lange, eds, Lecture Notes in Computer Science, Vol. 9678, Springer, Cham, 2016, pp. 677–692. doi:10.1007/978-3-319-34129-3_41.

[18] A. Khalili, P. Van den Besselaar and K.A. de Graaf, FERASAT: A Serendipity-Fostering Faceted Browser for Linked Data, in: *The Semantic Web. ESWC 2018*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, eds, Lecture Notes in Computer Science, Vol. 10843, Springer, Cham, 2018, pp. 351–366. doi:10.1007/978-3-319-93417-4_23.

[19] M. Koho, E. Heino and E. Hyvönen, SPARQL Faceter – Client-side Faceted Search Based on SPARQL, in: *Joint Proceedings of the 4th International Workshop on Linked Media and the 3rd Developers Hackshop*, CEUR Workshop Proceedings, Vol 1615, 2016.

[20] N. Bikakis and T. Sellis, Exploration and Visualization in the Web of Big Linked Data: A Survey of the State of the Art, in: *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference*, CEUR Workshop Proceedings, Vol 1558, 2016.

[21] J. Klímek, P. Škoda and M. Nečaskỳ, Survey of tools for Linked Data consumption, *Semantic Web* **10**(4) (2019), 665–720. doi:10.3233/SW-180316.

[22] M.A. Hearst, *Search User Interfaces*, Cambridge University Press, 2009. ISBN 978-0-521-11379-3.

[23] M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen and K.-P. Lee, Finding the flow in web site search, *Communications of the ACM* **45**(9) (2002), 42–49. doi:10.1145/567498.567525.

[24] A.S. Pollitt, The key role of classification and indexing in view-based searching, Technical Report, University of Huddersfield, UK, 1998.

[25] G.M. Sacco, Dynamic taxonomies: guided interactive diagnostic assistance, in: *Encyclopedia of Healthcare Information Systems*, N. Wickramasinghe, ed., Idea Group, 2005.

[26] E. Casalicchio and V. Perciballi, Auto-scaling of Containers: the Impact of Relative and Absolute Metrics, in: *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, IEEE, 2017, pp. 207–214. doi:10.1109/FAS-W.2017.149.

[27] E. Ikkala, J. Tuominen, J. Raunamaa, T. Aalto, T. Ainiala, H. Uusitalo and E. Hyvönen, NameSampo: A Linked Open Data Infrastructure and Workbench for Toponomastic Research, in: *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Geospatial Humanities*, ACM, 2018, pp. 2:1–2:9. doi:10.1145/3282933.3282936.

[28] R.C. Martin, *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall, 2002. ISBN 9780135974445.

[29] H. Rantala, E. Ikkala, I. Jokipii, M. Koho, J. Tuominen and E. Hyvönen, WarVictimSampo 1914–1922: A Semantic Portal and Linked Data Service for Digital Humanities Research on War History, in: *Proceedings of ESWC 2020, Posters and Demos*, Springer-Verlag, 2020, Accepted.