EPISA Platform: A Technical Infrastructure to Support Linked Data in Archival Management*

Sérgio Nunes*, Tiago Silva, Cláudia Martins and Rita Peixoto

INESC TEC and Faculty of Engineering, University of Porto, Portugal, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

Abstract

In this paper we describe the EPISA Platform, a technical infrastructure designed and developed to support archival records management and access using linked data technologies. The EPISA Platform follows a client-server paradigm, with a central component, the EPISA Server, responsible for storage, reasoning, authorization, and search; and a frontend component, the EPISA ArchClient, responsible for user interaction. The EPISA Server uses Apache Jena Fuseki for storage and reasoning, and Apache Solr for search. The EPISA ArchClient is a web application implemented using PHP Laravel and standard web technologies. The platform follows a modular architecture, based on Docker containers. We describe the technical details of the platform and the main user interaction workflows, highlighting the abstractions developed to integrate linked data in the archival management process. The EPISA Platform has been successfully used to support research and development of linked data use in the archival domain in the context of the EPISA project.

Keywords

Linked data, Archives, Platform, Software engineering

1. Introduction

Linked Data is a broad concept describing both a set of design principles and a set of specific technologies that have the goal to improve data management and access, specifically by contributing to data description and data integration. The term linked data was coined by Tim-Berners Lee in 2006 in the context of the semantic web project [1]. Many organizations and projects have since adopted linked data with diversified goals, from making public data openly available [2], to improving data querying and exploration for users [3], or contributing to data interoperability between systems [4].

Archival records management presents itself as a complex information context where linked data has the potential to impact at different levels, from record creation and description, to record access and exploration. The EPISA Project[1] is a Portuguese funded project that explores this opportunity to develop standards, processes, and technologies to support linked data use in the

TPDL2022: 26th International Conference on Theory and Practice of Digital Libraries, 20-23 September 2022, Padua, Italy

*This work is financed by National Funds through FCT - Foundation for Science and Technology I.P., within the scope of the EPISA project - DSAIPA/DS/0023/2018.

*Corresponding author.

ssn@fe.up.pt (S. Nunes)

0000-0002-2693-988X (S. Nunes)

[1] https://episa.inesctec.pt

archival record management context. A key piece of the project is the technical infrastructure that supports linked data storage, management, and overall user interaction.

In this paper we describe the EPISA Platform, the computational infrastructure developed to support the use of linked data in the archival records management context. Among the obstacles hindering the use of linked data technologies is the lack of tools, specifically software and resources [5]. We contribute to this problem by showcasing and describing the design and technical details related to the use of linked data in an archival systems management platform. The EPISA Platform uses the ArchOnto ontology [6] as a data model and is based on open-source software, most notably Apache Jena as a triplestore engine.

The remaining of this paper is organized as follows. In Section 2 we survey existing solutions and examples of computational infrastructures designed to support linked data use in the archival domain. Section 3 presents a high-level view of the EPISA Platform, which adopts a client-server model composed by the EPISA Server, presented in Section 4, and the EPISA ArchClient, a web application designed to support user interaction and presented in Section 5. In Section 6 we highlight two use cases to showcase the application of the linked data paradigm to records management, specifically record creation and description, and record navigation. Finally, the conclusions and future work perspectives are presented in Section 7.

2. Linked Data Platforms

In this section we identify works that detail the technical aspects of solutions developed to support linked data use in the context of archival records management systems. The majority of the existing works focus on exposing information as linked data, and offering views to query and navigate record collections, not focusing on the process of creating linked data.

Wikibase[2] is the software that supports Wikidata, an open, large-scale, collaborative, knowledge base designed to manage "factual information". As of May 2022, Wikidata contains nearly 100 million items[3]. Wikibase is also used to support other knowledge bases, including archival contexts (e.g., German National Library, Europeana)[4]. Diefenbach et al. [7] describe how to use Wikibase as an infrastructure to create domain-specific knowledge graphs. From a technical point of view, Wikibase data is stored on a relational database (MariaDB), indexed for keyword-based search using Elasticsearch, and exported to a triplestore (internal Blazegraph fork) to be queried through SPARQL user interface. In the EPISA Platform we adopt semantic web technologies as central pieces of the infrastructure, specifically we use a triplestore for data storage. One important characteristic of the Wikibase infrastructure is the support for tracking changes and individual contributions.

metaphactory is a commercial software platform designed to support knowledge graph applications [8], including extraction and integration, storage, querying, and data authoring. The metaphactory platform is targeted at expert users (e.g., schema design), end users (e.g., user friendly interaction and access), and also application developers (e.g., build specialized tools). metaphactory uses a triplestore for data storage and uses SPARQL for interaction with

[2]https://wikiba.se
[3]https://www.wikidata.org/wiki/Wikidata:Statistics
[4]https://wikiba.se/showcase

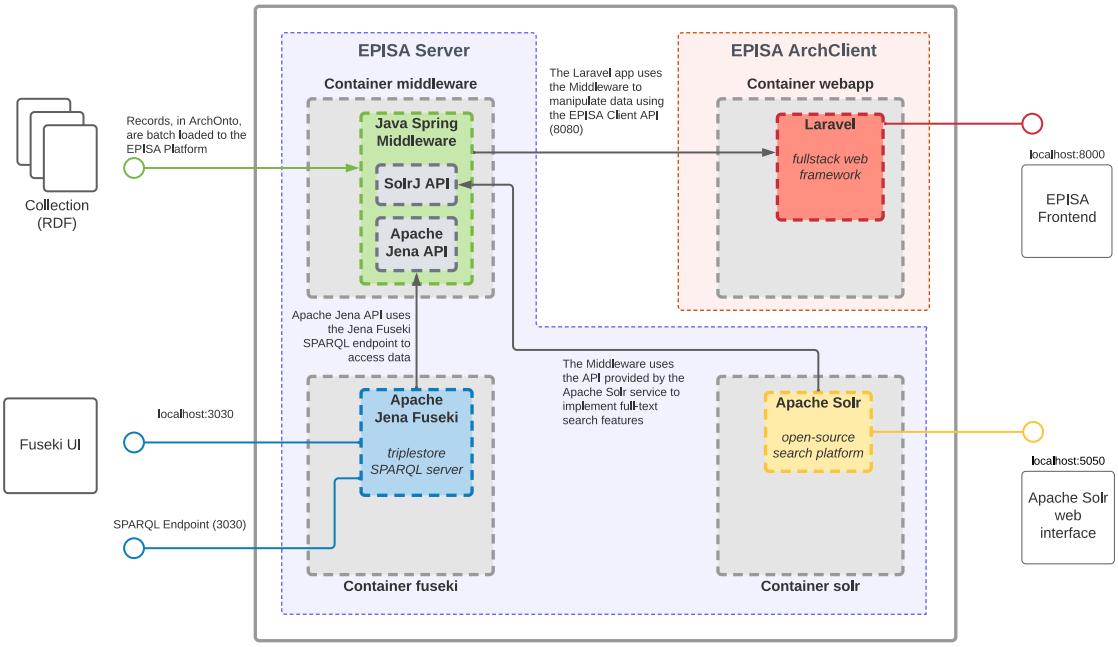

Figure 1: EPISA Platform Docker Environment.

the data layer, making the platform independent of specific database management systems. At the user interface level, a customizable templating mechanism is used. In the EPISA Platform we also use programmable user interfaces to easily accommodate different data representations. methaphactory showcases a wide list of application deployments, including ResearchSpace [9] in the cultural heritage domain.

Our work contributes to the state of the art by presenting and describing a platform to support linked data storage and management, querying and visualization, and data authoring through graphical user interfaces. Our work is supported on open technologies and planned to be released using an open license.

3. EPISA Platform Overview

The EPISA Platform architecture, depicted in Figure 1, consists of two main components – the EPISA Server and the EPISA ArchClient. The platform adopts a client-server paradigm between these components, at a lower level, it follows a microservice oriented architecture as both components constitute a group of independent services.

The EPISA Server is responsible for storing and performing reasoning over the archival data, manage access to the data, and to provide an effective search mechanism. These duties are handled by the different microservices contained in the component. Storage and reasoning is performed by an Apache Jena service, and the search aspects are implemented with an Apache Solr instance. Additionally, a Spring Boot application acts as a middleware and connects to the storage and search services. Ultimately, it provides a gateway for external componets to access

the EPISA Server's features.

The EPISA ArchClient is a web application responsible for providing an interface for archivists to access, manage and describe collections of archival records, using the components implemented by the EPISA Server. The ArchClient is composed of a Laravel application, that serves both the frontend and the backend of the provided web application.

As the platform's underlying architecture follows a microservice strategy, the services are implemented and deployed as individual Docker containers. Each of the afforementioned services is an instance of a Docker image running in its own environment. However, the services share a network that allows them to communicate with each other. This comunication, at the component level, is mediated by the middleware service that provides a REST API to perform all actions and operations that the EPISA-Server yields. Through this API, the ArchClient, as its only client, is able to search for, access, create and curate archival records and the entities that are mentioned in them.

In the following sections we detail both components, as well as their underlying services.

4. EPISA Server

The EPISA Server component comprises three microservices to deliver its several features: Apache Jena Fuseki, a SPARQL server responsible for data storing and reasoning; Apache Solr, the search platform; and a Spring Boot application for managing authorization. This section fully depicts each of these microservices and the middleware that connects them.

4.1. Middleware

The EPISA Server incorporates a Spring Boot application that acts as a middleware to connect the remaining microservices, Apache Jena Fuseki and Apache Solr. Figure 1 demonstrates how the microservices in the EPISA Server are internally connected, and also the connection between the EPISA Server and the EPISA ArchClient. The middleware is the component responsible for managing the interaction with the clients of the EPISA Server component, through the EPISA Client API it implements. Moreover, the middleware also connects to the Apache Solr service through the use of the SolrJ API and to the Apache Jena Fuseki through the several APIs provided by the Apache Jena framework. Details on the Apache Jena Fuseki and Apache Solr services can be found in the next subsections.

As the core element of the EPISA Platform, the middleware needs to complete a sequence of operations and interactions with the other microservices to kickstart the platform and to enable the distribution of the features it maintains. This process is depicted in Figure 2 where we can identify multiple stages: Stage 1 corresponds to the migration of records to the ArchOnto ontology [10]. This process results in a collection of records described with the ArchOnto ontology [11][5]; Stage 2 is the process of importing data to the triplestore. In this stage, the middleware reads the files that specify the ArchOnto ontology and the files that comprise the collections of records previously migrated and loads them to the Apache Jena Fuseki service. The Apache Jena Fuseki service then stores and performs reasoning over the received data; Stage 3

[5] https://purl.archive.org/episa/archonto

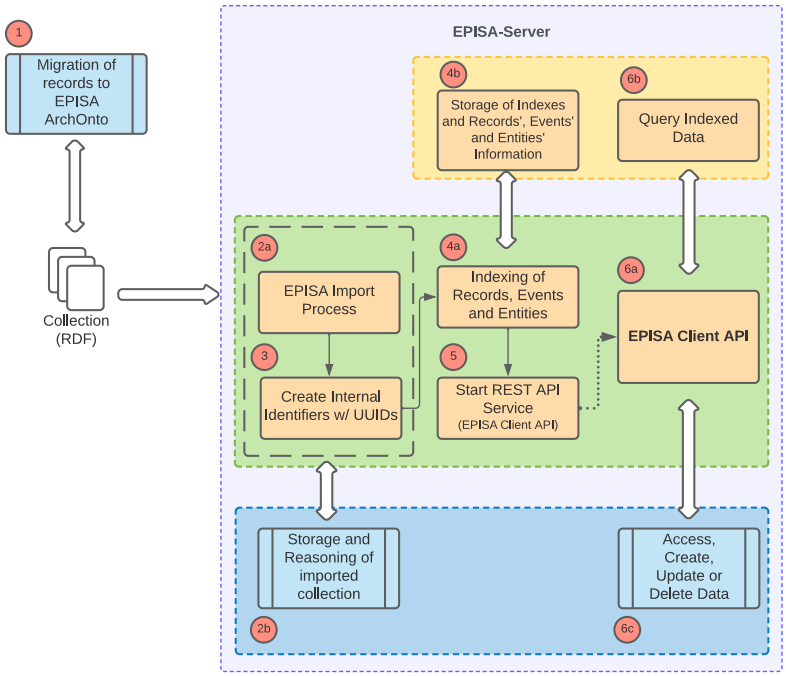

Figure 2: Sequence of operations.

comprises the creation of internal identifiers. Even though the ArchOnto ontology describes a schema for representing identifiers of documents, events, and entities, such identifiers are from within the context of the archival description. For this, they may not ensure the properties needed for the proper functioning of the EPISA Server component. The EPISA Server requires a guaranteed unique and individual identifier for each document, event, and entity object present in the system. To fulfill this condition, the middleware generates a unique identifier to attribute to each document, event and entity using Java's UUID library[6] and appends it to the model. In Stage 4, documents, events and entities are indexed in the Apache Solr service. This process is described in Section 4.3; Stage 5 and 6 represent the initialization of the EPISA Client API and the operations performed upon the arrival of an HTTP request from the EPISA ArchClient.

The EPISA Client API is responsible for making all the features of the EPISA Server available over HTTP. Table 1 contains a short description of the endpoints available in this API. The responses provided by this API are in JSON format. In the case of an archival record, the JSON response is structured into five different sections: identity, context, access and use conditions, linked resources, and linked data. These sections are based on ISAD(G) areas of descriptive information [12]. Regarding the entities, the JSON response contains three zones: identity, linked data, and linked resources. The identity zone includes the fields that identify a document, such as an identifier, title, description level and reference code, or an entity, such as identifier, type and name. The context zone contains information regarding a register's subjects, conservation states, typologies, writings and documentary traditions. The access and use conditions of a

[6]https://docs.oracle.com/javase/8/docs/api/java/util/UUID.html

record identify its physical location, technical requirements related to its physical condition, language and access conditions. The linked resources zone identifies the resources linked to a determined entity or archival record. Lastly, the linked data zone identifies archival records and entities related to a specific archival record or entity.

Table 1
EPISA Client API.

Endpoint	Method	Description
/docs	GET	Get all documents belonging to the platform. Can be filtered according to keywords inserted.
/doc	POST	Add a new record of a document to the system.
/doc/<uuid>	GET	Get the document identified by UUID.
/doc/<uuid>	PUT	Update information of the document identified by UUID.
/doc/<uuid>	DELETE	Delete all the information about the document identified by UUID.
/entities	GET	Get all entities belonging to the platform. Can be filtered according to keywords inserted.
/entity	POST	Add a new record of an entity to the system.
/entity/<uuid>	GET	Get the entity identified by UUID.
/entity/<uuid>	PUT	Update information of the entity identified by UUID.
/entity/<uuid>	DELETE	Delete all the information about the entity identified by UUID.
/events	GET	Get all events belonging to the platform. Can be filtered according to keywords inserted.
/event	POST	Add a new record of an event to the system.
/event/<uuid>	GET	Get the event identified by UUID.
/event/<uuid>	PUT	Update information of the event identified by UUID.
/event/<uuid>	DELETE	Delete all the information about the event identified by UUID.

4.2. Apache Jena Fuseki

As the EPISA Platform brings linked data concepts into the archival domain, all the data from the archival records is stored in the form of RDF triples. A native triplestore was adopted for data storage, specifically Apache Jena[7] was choosen since it is an open-source, high-performance solution, with a frequent release cycle, and supports a rich set all of APIs to process RDF data. For querying and updating the RDF data, Apache Jena Fuseki is being used for exposing the RDF data as a SPARQL endpoint accessible over HTTP, providing REST-style interaction with the underlying data.

Additionally, the Eclipse RDF4J framework[8] is used by the middleware to create SPARQL queries programmatically, particularly the SparqlBuilder API, which eases the process of querying the Fuseki service to obtain data. Moreover, Apache Jena also supports inference over the stored triples, a fundamental process when dealing with RDF data and OWL. Apache Jena comes with several reasoners that differ on the level of reasoning provided. The reasoner currently being used is the OWL Micro reasoner, layered on top of the datasets belonging to the Fuseki service. Table 2 shows the constructs supported by the OWL Micro Reasoner that provide the inference support for the EPISA Platform.

[7] https://jena.apache.org
[8] https://rdf4j.org

Table 2
Constructs Supported by the OWL Micro Reasoner.

Prefix	Constructs supported
rdf	rdf:type
rdfs	rdfs:subPropertyOf, rdfs:subClassOf, rdfs:range, rdfs:domain
owl	owl:intersectionOf, owl:unionOf, owl:equivalentClass, owl:hasValue, owl:Thing, owl:equivalentProperty, owl:inverseOf, owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:SymmetricProperty, owl:TransitiveProperty

4.3. Search

Apache Solr[9] is used to implement the keyword-based search engine in the EPISA platform. A Solr Docker container is initialized with two different cores (i.e., indexes), one to index the archival records and other to index the entities (people, places, organizations, and events). The indexing process is started once the data is loaded to the triplestore and the Fuseki service obtains all the necessary information about each record or entity. When there is the need to search for information, the search keywords are matched against the indexes in the Solr cores to retrieve the matching results. At the user interface level, search results are presented separately in two rankings, one for records, and another for entities.

As the archival records and entities in the EPISA platform are described using the ArchOnto ontology, the indexing process relies on the specificities of that model to provide more accurate results, for example, by indexing specific properties and relations. The information indexed refers to archival records' and entites' identity, linked data and linked resources zones. It also expands ArchOnto properties and classes by parsing and translating them to Portuguese, allowing more complex searches and the use of the Portuguese language.

Figure 3 shows an RDF subgraph with a partial representation of the birth of Person1. In this case, when indexing information about Person1, the linked entity Birth1 will be expanded. By doing so, Person2 and the property *P96 by mother* will be associated to Person1 in the index. Moreover, the properties *P98 by mother* and *P96 was born* will also be translated to portuguese so that the search for the birth or the mother of Person1 can be done in portuguese. In the case of Person2, the reverse will happen and Person1 will also be associated to its index as a child.

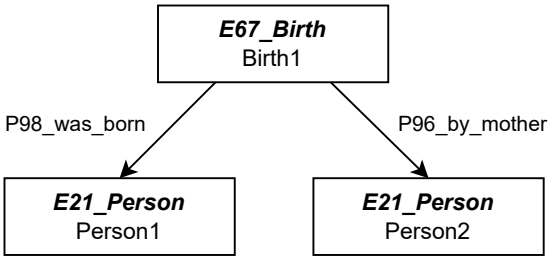

Figure 3: Representation of a birth in ArchOnto (subgraph).

[9]https://solr.apache.org

5. EPISA ArchClient

The EPISA ArchClient is a web application that provides archivists an interface to access, manage and describe collections of archival records and entities such as people, places, groups, organizations, and events supported by the records being described. It intends to exploit the capabilities of connecting information and concepts that linked data concepts and technologies provide, and thus provide increased detail in the representation of archival information.

The ArchClient is implemented using Laravel[10], a full stack, open-source PHP web framework. The option for this framework is justified by the requirement of having a light, low demanding web application on the client side. Therefore, both the frontend and the backend of the provided web application are implemented with Laravel and standard web technologies. The web application implements the Model-View-Controller pattern, a common approach in web frameworks that Laravel's guidelines lightly enforce. Despite this, the application comprises only two main components, the View and the Controller components. In this case, given that the data is held and pulled from the EPISA Server component through HTTP requests to the EPISA Client API, the Model component is not implemented. Instead, the EPISA Server acts as the application's Model by being responsible for, among other tasks, the storage of the data required by the ArchClient. The web application's implemented components are the typical View and Controller components of an implementation of the MVC pattern.

The Controller is assigned with the duty of implementing the logic of the application, more specifically the actions that need to be executed according to the requests and input received from the View. Thus, this component's main tasks are handling and routing the View's requests, validating the input attached to them and performing the required HTTP requests to the EPISA Server component's Client API. The View is responsible for defining the actual frontend of the application. In its implementation in the ArchClient we used a PHP templating engine in Blade for producing and structuring the application's pages and TailwindCSS libraries[11], CSS, and plain JavaScript to style and enhance them.

For the event's view, as the attributes are event type specific, a programmable user interface was developed. This interface constructs its fields according to the data the EPISA Server provides and works for both the event's visualization and the event's creation and edition. It becomes possible using Micro-form[12], a library to translate any datasource into into HTML form elements. The EPISA Server provides a JSON object with the necessary elements and their properties (type, name, verifications, etc) and the EPISA ArchClient interprets it and builds the layout of the interface with the corresponding fields.

6. User Interaction Use Cases

In this section, we describe the user interaction abstractions by presenting two illustrative use cases for the archivist user – search and navigation, and the description process. The adoption of linked data opens up many opportunities to improve data description and access, but user

[10] https://laravel.com
[11] https://tailwindcss.com
[12] https://github.com/marcomilon/micro-form

interaction with linked data, in particular for authoring data, is still a challenge with many open problems [13]. The work presented here is informed by a series of interviews and user tests conducted with professional archivists [14].

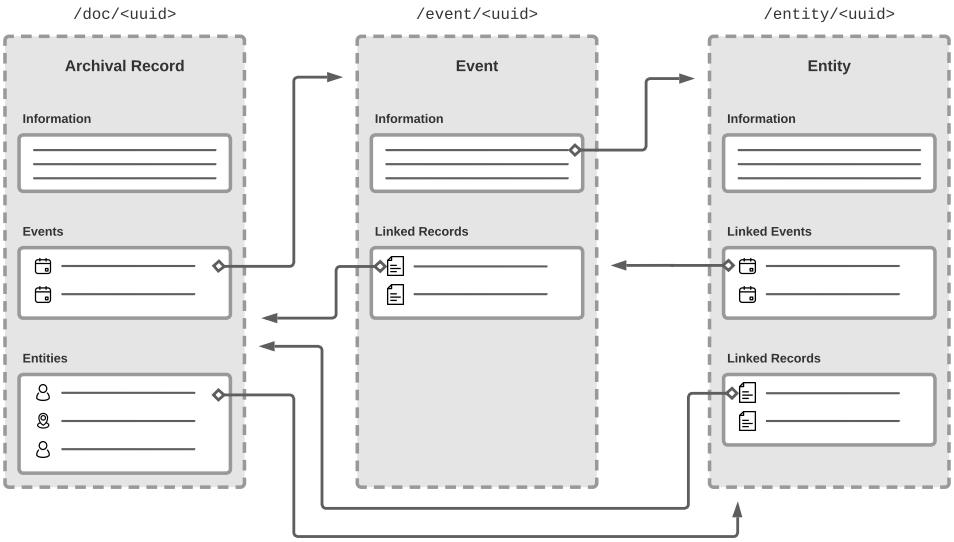

Figure 4: Rich picture illustrating the central information concepts available at the user interface level.

In the EPISA ArchClient, the end user is presented with three central information concepts — the records, the events, and the entities, as illustrated in Figure 4. Although all data is stored in a single underlying graph using the ArchOnto model, these concepts are introduced at the logical level to structure user interaction. The *records* represent the collection of archival documents, described at different levels (e.g., fonds, collections, records). The *entities* represent the concepts (i.e., persons, places, organizations) that are mentioned in the records being described. Finally, the *events* allow the creation of more complex structures linking records and entities — e.g., birth and death events, marriages, places of domicile.

Search and Navigation An archivist can use the EPISA ArchClient to perform a search on the records and entities held by the system. The system provides four categories of search: (i) simple search, where a full-text search is performed with the text term given by the archivist; (ii) hierarchy search, where the archivist can navigate through the hierarchy of records and entities and find and see the relationships between each one of them; (iii) advanced search, where archivists can add to their search, filters with different existing categories; and (iv) structured search, where the archivist can build a complex query to find the desired records/entities. Each individual record, event, and entity work as direct entry points in the EPISA ArchClient, i.e. each has a unique address. Figure 5 shows the record view. From each record, the user can navigate to associated events or directly to specific entities. Similarly, from each entity, the user can navigate to the associated events.

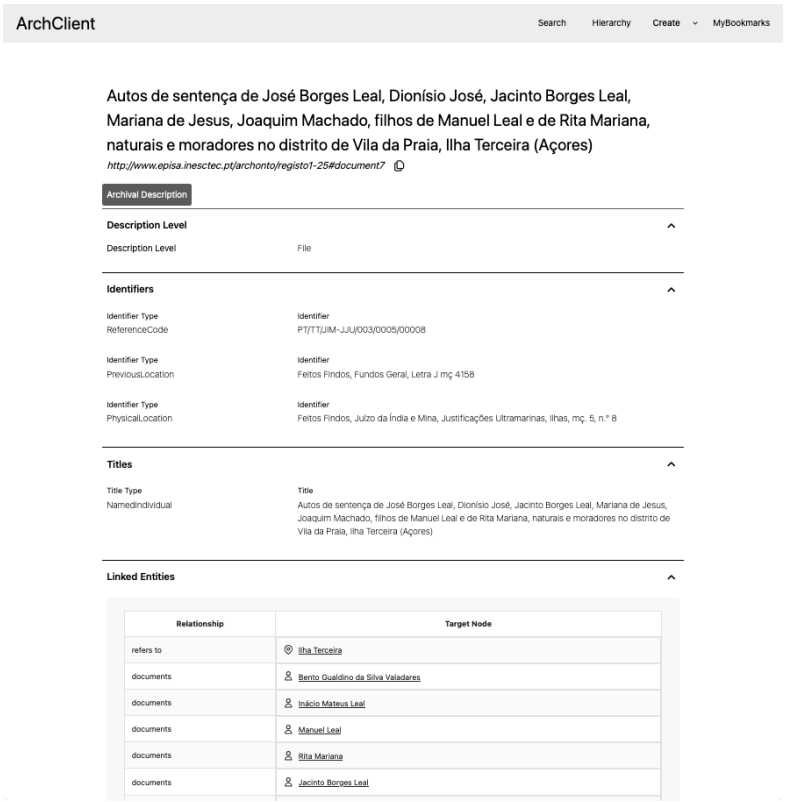

Figure 5: EPISA ArchClient screenshot (record view).

Description Process The description process always starts with the creation of a new archival record. For each record, the standard description properties can be defined (e.g., description level, support, languages, access). Additionally, each record can be associated with multiple events. Events are created using custom user interfaces that can be defined programmatically.

7. Conclusions

We have described the EPISA Platform, a computational infrastructure developed to support archival records management using linked data technologies. The system is currently under active development with internal prototypes being evaluated by professional users. This work contributes to the development and the adoption of linked data technologies by providing details about a real-world implementation in the context of the archival domain. Future work includes the design and implementation of an authorization mechanism to support fine-grained control of data access. Also planned is the design and organization of user studies with professional archivists. These studies are central to understand how the proposed user interface abstractions and overall workflows fit the archivists processes. Additionally, these user studies will also include the evaluation of the keyword-based search – which is expected to greatly impact the way information is retrieved, taking advantage of the links between data items.

References

[1] T. Berners-Lee, Linked Data - Design Issues, 2009. URL: https://www.w3.org/DesignIssues/LinkedData.html.

[2] European Union, data.europa.eu, 2022. URL: https://data.europa.eu/en.

[3] Europeana Foundation, Linked Open Data | Europeana Pro, 2022. URL: https://pro.europeana.eu/page/linked-open-data.

[4] Schema.org, Schema.org, 2022. URL: https://schema.org.

[5] K. Smith-Yoshimura, Analysis of 2018 International Linked Data Survey for Implementers, Code4Lib Journal (2018).

[6] I. Koch, C. Ribeiro, C. T. Lopes, ArchOnto, a CIDOC-CRM-Based Linked Data Model for the Portuguese Archives, in: Digital Libraries for Open Knowledge, Springer International Publishing, 2020, pp. 133–146.

[7] D. Diefenbach, M. D. Wilde, S. Alipio, Wikibase as an Infrastructure for Knowledge Graphs: The EU Knowledge Graph, in: The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings, volume 12922 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 631–647. doi:10.1007/978-3-030-88361-4_37.

[8] P. Haase, D. M. Herzig, A. Kozlov, A. Nikolov, J. Trame, metaphactory: A platform for knowledge graph management, Semantic Web 10 (2019) 1109–1125. doi:10.3233/SW-190360.

[9] D. Oldman, D. Tanase, Reshaping the Knowledge Graph by Connecting Researchers, Data and Practices in ResearchSpace, in: The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II, volume 11137 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 325–340. doi:10.1007/978-3-030-00668-6_20.

[10] D. Melo., I. Rodrigues., I. Koch., Knowledge Discovery from ISAD, Digital Archive Data, into ArchOnto, a CIDOC-CRM based Linked Model, in: Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KEOD, INSTICC, SciTePress, 2020, pp. 197–204. doi:10.5220/0010134101970204.

[11] I. Koch, C. Ribeiro, C. Teixeira Lopes, ArchOnto, a CIDOC-CRM-Based Linked Data Model for the Portuguese Archives, in: Digital Libraries for Open Knowledge: 24th International Conference on Theory and Practice of Digital Libraries, TPDL 2020, Lyon, France, August 25–27, 2020, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2020, pp. 133–146. URL: https://doi.org/10.1007/978-3-030-54956-5_10. doi:10.1007/978-3-030-54956-5_10.

[12] ISAD(G): General International Standard Archival Description, Standard, International Council on Archives (ICA), Stockholm, SE, 1999.

[13] M. Aguiar, S. Nunes, B. Giesteira, A Survey on User Interaction with Linked Data, in: Proceedings of the Sixth International Workshop on the Visualization and Interaction for Ontologies and Linked Data co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, 2021, volume 3023 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 13–28.

[14] C. Guedes, B. Giesteira, S. Nunes, Designing user interaction with linked data in historical archives, J. Comput. Cult. Herit. (2021). URL: https://doi.org/10.1145/3485731. doi:10.1145/3485731.