

	 <p>Transforming Research through Innovative Practices for Linked Interdisciplinary Exploration</p>
[SEPTEMBER 2021]	Advancing Open Scholarship
	D5.5 – REPORT ON THE OPEN ANNOTATION TOOL Version 1.0 – Final PUBLIC
	H2020-INFRAEOSC-2019 Grant Agreement 863420

The project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 863420.

Disclaimer- “The content of this publication is the sole responsibility of the TRIPLE consortium and can in no way be taken to reflect the views of the European Commission. The European Commission is not responsible for any use that may be made of the information it contains.”

This deliverable is licensed under a Creative Commons Attribution 4.0 International License.



Deliverable Name

Project Acronym:	TRIPLE
Project Name:	Transforming Research through Innovative Practices for Linked Interdisciplinary Exploration
Grant Agreement No:	863420
Start Date:	1/10/2019
End Date:	31/03/2023
Contributing WP	WP5
WP Leader:	Net7
Deliverable identifier	D5.5
Contractual Delivery Date: 09/2021	Actual Delivery Date: 09/2021
Nature: Report	Version: 1.0 Final
Dissemination level	PU

Revision History

Version	Created/Modifier	Comments
0.0	Luca De Santis (Net7)	Index
0.1	Duccio Breschi, Niccolò Cardelli, Luca De Santis, Edgar Gomez, Massimiliano Pardini, Marco Zizi (Net7)	Contributions
0.2	Luca De Santis (Net7)	First full version
0.3	Maxime Bouillard (MEOH), Simone Kopeinik (Know-Center), Laurent Capelli (Huma-Num)	Revisions
1.0	Luca De Santis	First released version

Table of Contents

DESCRIPTION OF THE SERVICE	6
<u>INTRODUCING PUNDIT, THE TRIPLE OPEN ANNOTATION TOOL</u>	6
<u>THE IMPORTANCE OF SEMANTIC ANNOTATIONS</u>	10
THE SERVICE IN TRIPLE	11
SOFTWARE ARCHITECTURE	12
<u>AN OVERVIEW</u>	12
<u>PUNDIT ANNOTATOR</u>	15
<u>THE ANCHORING MODULE</u>	15
<u>PUNDIT BACK-END: THE ANNOTATION SERVER</u>	16
<u>ANNOTATION APIs</u>	17
<u>W3C SEARCH ENDPOINT</u>	20
<u>PUNDIT DATA LAYER</u>	25
<u>PUNDIT APP</u>	28
INTEGRATION STRATEGIES OF THE SERVICE IN GOTRIPLE	30
DESCRIPTION OF THE WORK DONE IN TASK T5.5	34
<u>A FULL ARCHITECTURAL REFACTORING</u>	34
<u>NEW FUNCTIONALITIES IN PUNDIT</u>	37
<u>TAGGING SUPPORT</u>	37
<u>SEMANTIC ANNOTATIONS</u>	38
NEXT STEPS AND EVOLUTIONS	39
REFERENCES	42

Table of Figures

<i>Figure 1 - Pundit's main services</i>	6
<i>Figure 2 - Pundit in action</i>	7
<i>Figure 3 - Social features in Pundit</i>	8
<i>Figure 4 - Semantic Annotations in Pundit</i>	9
<i>Figure 5 - The new Pundit architecture</i>	14
<i>Figure 6 - Annotation Server's APIs deployed on the Amazon AWS cloud</i>	17
<i>Figure 7 - Structure of an annotation according to the W3C Web Annotation Data Model</i>	26
<i>Figure 8 - Pundit data model of the main data store on Elasticsearch</i>	27
<i>Figure 9 - MariaDB data model in Pundit</i>	27
<i>Figure 10 - The Pundit App with the list of the user's notebooks</i>	29
<i>Figure 11 - Linking user accounts between GoTriple and Pundit</i>	32
<i>Figure 12 - MyGoTriple "What's new" page with Pundit notifications</i>	33
<i>Figure 13 - An example of a wrong anchoring behaviour on a modern website by the old Pundit</i>	35
<i>Figure 14 - The redundancy of the old Pundit protocol</i>	36
<i>Figure 15 - The architecture of the previous version of Pundit</i>	36
<i>Figure 16 - A synthesis of the main problems that afflicted the previous version of Pundit</i>	37
<i>Figure 17 - Tagging in action in Pundit</i>	38
<i>Figure 18 - A mock-up of the future implementation of semantic annotations in Pundit</i>	39

Acronyms

AGPL	GNU Affero General Public License
API	Application Programming Interface
AWS	Amazon Web Services
CRUD	Create, Read, Update, Delete
DOM	Document Object Model
EGI	European Grid Infrastructure
EOSC	European Open Science Cloud
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JWT	JSON Web Tokens
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RDS	Amazon Relational Database Service
RFC	Request for Comments: RFC documents contain technical specifications and organizational notes for the Internet.
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WA	Web Annotation
WP	Work-package
WP3	Work-package 3 – “Co-design and user research”
WP4	Work-package 4 – “Integration and building of TRIPLE platform”
WP5	Work-package 5 – “Development and integration of innovative services”
WP7	Work-package 7 – “Innovation, Exploitation and Sustainability”
WP8	Work-package 8 – “Communication and Dissemination”
XPath	XML Path Language
YAML	Yet Another Markup Language: YAML is a human-readable data-serialization language commonly used for configuration files

Publishable Summary

We present here the work done in task T5.5, dedicated to the development and the integration of the Open Annotation Tool. Its purpose is to provide GoTriple's users with a service that they can directly exploit in their daily activities of research: it allows in fact to “take notes” (highlights, comments, tags but also semantic annotations) on web content, in particular on web pages, seen via a standard browser, and, soon, also on PDF documents.

This tool, which is one of TRIPLE's Innovative Services, wasn't developed from scratch but it leveraged the almost decennial experience of TRIPLE's partner Net7 with its web annotation tool Pundit. During this first part of TRIPLE, a large effort was dedicated to significantly refactor Pundit's technology, whose state, at the beginning of the project, was outdated. New features have been also developed and the integration with the GoTriple Discovery Portal fulfilled: with Pundit, users can “take notes” on the content discovered through GoTriple and receive in their MyGoTriple page the main notifications regarding their annotations, for example if they have received a reply or a like (or dislike).

This document starts by describing the characteristics of Pundit, especially its most distinguished feature which is the support of semantic annotations, and its positioning in the context of the TRIPLE project. Its updated architecture is presented in detail, together with the technical strategy that has been designed and implemented for integrating Pundit in GoTriple. A report of the development work done in T5.5 in the past months follows. Finally, an outline of the multiple, and diverse, activities to carry on in the next period of TRIPLE is provided. In particular, areas where task T5.5 efforts will be focused include developing new annotation functionalities, increasing users' adoption, distributing the new Pundit software code with an Open Source license and identifying exploitation models and sustainability strategies.

1 | DESCRIPTION OF THE SERVICE

1.1 Introducing Pundit, the TRIPLE Open Annotation Tool

The Open Annotation Tool is one of the Innovative Services of the TRIPLE project. It consists of a web-based tool that allows users to create annotations on web content (web pages and, soon, PDF documents).

A simplified definition of a web annotation is “a user note on a web page”. Everybody is familiar with the idea of “taking notes” on paper, e.g. on books, articles or magazines: we highlight or underline the important parts of the text and add comments on the border of the page or in other “white spaces” of the document. This activity is in fact often referred to as “marginalia”.

Web annotators enable to do the same on content served on the web, in particular on web pages accessed through a standard browser.

The TRIPLE Open Annotation Tool is based on Pundit (<https://thepund.it>), a system whose development has been carried on by TRIPLE partner Net7 since 2012. The main services offered by Pundit are represented in Figure 1.

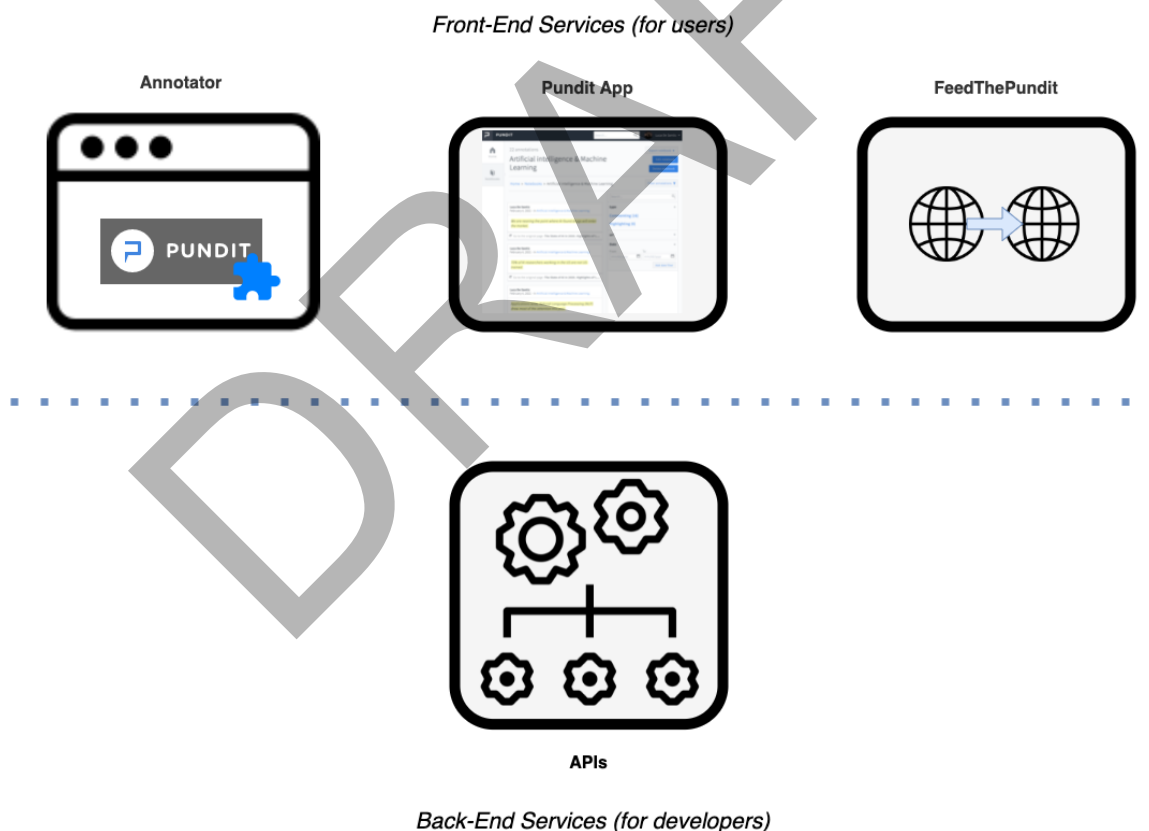


Figure 1 - Pundit’s main services

To start with, the tool used to do the annotations is the **Annotator**, which is implemented as an extension of the web browser Google Chrome. Anyone can search for “Pundit” in the Google Chrome Web Store¹ and install this extension.

In order to use the tool, it is necessary to register on the **Pundit App**, either by directly inserting a few data (e.g. email, password, name...) or by using an external Identity Service like Google, Facebook or EGI AAI Check-In. The Pundit App is not only a registration service but it provides a dashboard where users can see and manage in a centralised place all the annotations that they have done “on the web”. Here they can search and filter annotations with a faceted interface and also can export them in a variety of standard/open formats, including Microsoft Office and Open Office files (.xlsx, .docx and .odt) and, for a technical audience, JSON-LD files fully compliant with the W3C Web Annotation Data Model specification [1].

For those who don’t want to install the Chrome Extension or prefer to use another web browser, the **FeedThePundit** “proxy service” will be available soon². It allows users to open web pages from every browser, also from mobile devices, with Pundit being automatically activated.

Finally, for developers and system integrators, a set of **APIs** are also publicly available, that allows the development of custom integration scenarios with the Pundit service.

The tool at present can be freely used by everyone as a cloud service managed by Net7: a part of its codebase is also available as open source software with the AGPL 3.0 licence. Net7 is analysing the release of the service with a Freemium model, that is, free to use for basic functionalities while for more advanced and professional features the user must pay a recurrent subscription. This is in line with the general sustainability strategy for TRIPLE.

As said, Pundit allows users to “take notes” on web documents (e.g. a web page or a PDF file), in the form of **highlights** of text parts or of **comments**, applied either on a selection of the text or on the whole document.

It is also possible to create annotations to **tag** a selected text or an entire web page. This creates a free-form personal category that can be reused in other annotations: tags in fact can be also applied on other annotations and not only created stand-alone.

DOAB provides and operates the OPERAS “Certification Service”, which certifies open access (OA) book publishers, based on their practices, in particular their peer review (PR) procedures and policy. The service is intended to certify publishers at the institutional and individual publication level. It thereby supports transparency in book publishing — among readers and the service providers working with

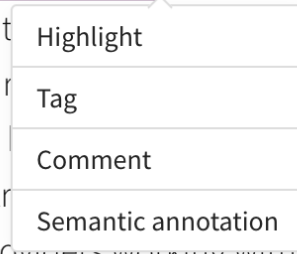


Figure 2 - Pundit in action

¹ <https://chrome.google.com/webstore/>

² At present this service is under refactoring and it is not available yet for the general public.

On existing annotations, **social features** can be applied. This is illustrated in Figure 3 and includes: i) replying to the annotation, ii) liking, disliking or even reporting an annotation to the Pundit administrator.

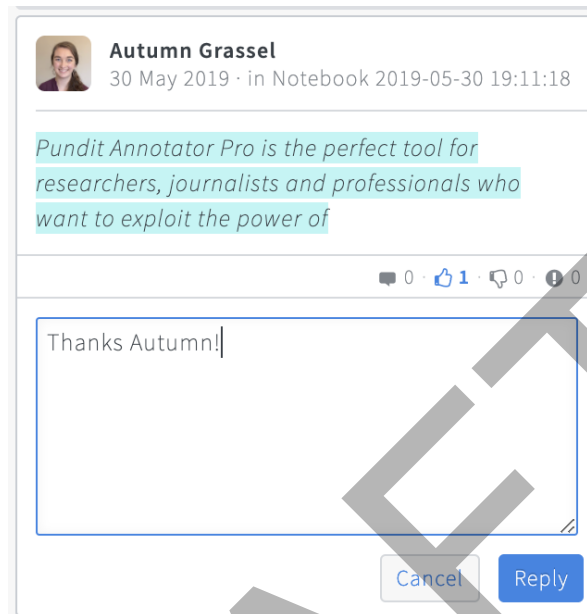


Figure 3 - Social features in Pundit

Pundit annotations are collected in **Notebooks**: users can have as many notebooks as they like and can make them Public or Private: all annotations contained in them can be, respectively, freely seen or not in their original web pages by using Pundit. Amongst the new features currently in development, there is also the possibility to share Notebooks with other users.

In this sense, the tool is not dissimilar to several other web annotation tools in the market, such as Hypothes.is³ or Weava⁴. On the other hand, Pundit has a very distinguished feature, that is the support of **semantic annotations**. With them it is possible to create an annotation as an RDF triple, composed by

subject - predicate -> object

where:

- *subject* is a fragment of text or the whole web document
- *predicate* defines a formal relationship amongst the subject and the object, according to the “rules” specified in a Semantic Web ontology
- *object* can be: i) a literal (e.g. a string); ii) an URI identifying an external resource or an entity defined in a Semantic Web ontology; iii) another fragment of text, possibly of a different web document.

³ <https://hypothes.is/>

⁴ <https://www.weavertools.com/>

While this functionality is of course not for the laypeople, it can prove extremely beneficial for professional users like scholars, who need to create “notes” in a formal and structured way.

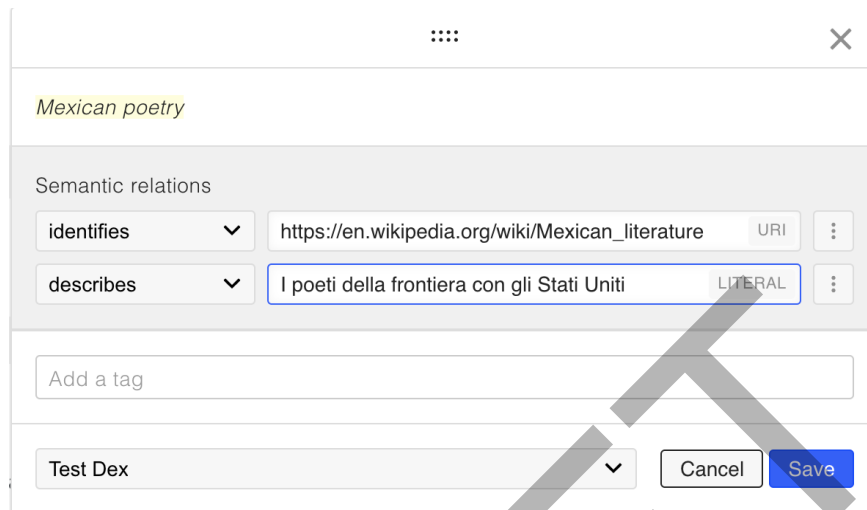


Figure 4 - Semantic Annotations in Pundit

In Figure 4, for example, we have two semantic annotations with:

- the text fragment “Mexican poetry” as the subject of both triples
- the predicates “identifies” and “describes”⁵
- respectively a Wikipedia entity and a text literal as objects.

The advantages that semantic annotations can offer to a professional audience are explained in section 1.2: here it is important to point out that from the very beginning Pundit supports this feature, since the tool has been always designed having scholars, and more generally, a professional audience in mind. Semantic Web technologies were in fact the topic of the SEMLIB research project [2], where the foundation of Pundit was laid.

The most “common” features, like highlighting and commenting, were introduced later in the Pundit development (around 2015), when, in the course of the StoM research project⁶ [3] an exploitation model for Pundit was devised: it revolved around the idea of having a free “basic” version of the tool and a “pro” one, which also includes the possibility to do semantic annotations.

Net7 expects to refine the idea of a Freemium model for Pundit and to implement it eventually. In this regard, the company benefits from its experience in TRIPLE and the collaboration with the WP7 team, which is studying sustainability models for the project’s outcomes. However, its exact path is still to be identified. As an example, from the experience gained so far, the idea of having two different versions of Pundit hasn’t proved to be valid and that’s why only one version of the Annotator is currently in development in TRIPLE.

⁵ respectively `rdfs:isDefinedBy` and `cito:describes`

⁶ <https://www.stom-project.eu>

Although Pundit, as a public web annotator, has been around for almost a decade, its technology was outdated when the TRIPLE project began. The work done in Task T5.5, as explained in detail in the following chapters, has been concentrated on a major refactoring of all of its components, together with the addition of new features. This work hasn't been concluded yet: we describe here the present state of the service, while in the final chapter a plan for the next period of work in TRIPLE is presented.

1.2 The importance of Semantic Annotations

Annotations can be simply seen as “attaching data to some other piece of data”, like a document for example. The advantage of a semantic annotation is to have this data formally defined and machine-understandable: this offers better possibility for search, reuse and exploitation of the annotations performed by users.

A simple but effective definition of a semantic annotation is proposed in [4], where it is stated that “semantic annotation is about assigning to the entities in text links to their semantic descriptions”.

The key for providing effective “semantic descriptions” lies in:

- the careful definition of ontologies, that is the vocabulary of terms, classes, predicates and properties according to which semantic annotations are performed by users. They can be seen as the “grammar” to properly create a semantic annotation.
- the use, as much as possible, of standard ontologies, whose meaning is therefore well-known and accepted. This enables the reuse of the annotations defined using them.
- the exploitation of Linked Data in performing annotations, e.g. as “target” (objects) of the triple. In a nutshell, Linked Data is about publishing structured data on the web. Datasets as Geonames or Wikipedia (or better, Wikidata and DBpedia, their Semantic Web equivalents) provide both huge, general purpose, sets of entities and terms, that can be referred in the annotation process, and families of well-known ontologies, that are so common to be considered standard and as such easily understandable by semantic-aware software agents.

The importance of semantic annotations in text, and of the use of Linked Data in particular, can be better appreciated through an example. Consider the following three sentences:

*On the morning of **September 11** Australian swimming legend Ian Thorpe was on his way to the World Trade Centre for a business meeting.*

*Born on **September 11** writer/director Brian De Palma's career began to take off in the 1970s with the horror classic *Carrie*, based on a Stephen King novel.*

*On **September 11**, President Salvador Allende of Chile was deposed in a violent coup led by General Augusto Pinochet.*

Even if they all contain a reference to the same day (September 11), its actual semantic is very different: the first one is an Event (in 2001), the second a Birth date (1940) and finally the last one refers both to a Death date (Allende's) and to an Event (Pinochet's Coup d'état in Chile in 1973). A simple textual indexing of these three texts would identify as the same concept the "September 11" fragments, without any understanding of their meanings, their contexts and the actual years they refer to. On the contrary, by simply annotating them with a link to the corresponding entities of Wikipedia⁷ and the use of specific predicates⁸, one can automatically infer all the previous information plus much more data.

Pundit allows users to easily create semantic annotations on web documents and also link them to standard Linked Data repositories, all with an intuitive and easy to use interface.

2 | THE SERVICE IN TRIPLE

The idea of incorporating a web annotation tool in TRIPLE is to provide users with a service that they can directly exploit in their daily activities of research. The main focus of this project is in fact the development of an effective discovery platform for SSH: some of the other Innovative Services developed in WP5 (e.g. the Recommender and the Visualization and Discovery Services) are designed to actually enhance the discovery experience but they do not provide any "active" tool for the users.

With the Open Annotation Tool users can "take notes" on the content discovered through GoTriple and receive in their MyGoTriple page the main notifications produced by Pundit, for example if their annotations have received a reply or a like (or dislike).

As explained in technical details in chapter 4, Pundit and GoTriple have an independent user base: some users might be registered in Pundit, others in GoTriple and some in both services. In the latter case the two registrations are completely independent, albeit the use in both systems of the EGI AAI Check-In service for user registration and authentication may enable single sign-on scenarios.

In order to create an effective connection between the two systems, it has been decided to create a workflow through which a registered GoTriple user can link her Pundit account. When this link has been created, all notifications produced by Pundit will also be shown on the MyGoTriple page of the user.

Of course, Pundit might also be used to "take notes" directly on GoTriple pages. The possibility to "embed" the client in the GoTriple front-end, in order to use it on the Discovery Platform without the need to install the Chrome Extension, is currently under evaluation by WP4 and WP5 teams.

⁷ Namely:

- September 11 attacks - http://en.wikipedia.org/wiki/September_11_attacks
- Brian De Palma - http://en.wikipedia.org/wiki/Brian_De_Palma
- 1973 Chilean coup d'état - http://en.wikipedia.org/wiki/1973_Chilean_coup_d%27%C3%A9tat

⁸ Eg. `dbpedia-owl:eventDate`, `dbpedia-owl:birthDate`, `dbpedia-owl:deathDate`

Of course these integration scenarios are just the starting point for the first public version of GoTriple: other more advanced possibilities will be analysed in the rest of the project, by also considering feedback from users, assessed either automatically with web analytics tools to monitor the real usage of the various features, or by executing user research actions in collaboration with WP3.

3 | SOFTWARE ARCHITECTURE

3.1 An overview

As it will be explained in detail in chapter 5, Pundit, in these first months of TRIPLE, has been the subject of a major refactoring that involved all its main parts.

The **Annotator** code in fact has been completely rewritten from scratch for several reasons. First of all the previous version was based on AngularJS, the first release of this framework which is incompatible (and therefore cannot be updated automatically) with the new versions of this technology⁹.

Moreover the anchoring logic, that is the “mechanism” by which annotations can be correctly attached to the part of an HTML page, was not efficient and especially wasn’t resilient to the changes of the structure of the page. For example, a new graphic design could cause a failure in recognising the text fragment where the annotation has to be attached, creating an “orphan” annotation. The anchoring module was completely rewritten by using the open source code of the Hypothes.is web annotation tool. The final result has been again released publicly as open source¹⁰.

The most important update was done on the **back-end of Pundit**, that is on the *Annotation Server* and its *APIs*. The previous version of Pundit was based on a monolithic Java servlet application, which performed all the business logic of the service, including the authentication and registration of users. The old Pundit was deployed on a private server on Net7’s server farm, hosted by the OVH international provider.

Various were the limits of this implementation: first of all, it has been demonstrated that having a monolithic server application is a bad design choice for a web service, since it makes maintenance and updates more complicated. As stated in a seminal article in Martin Fowler’s blog in fact “...with a monolith any changes require a full build and deployment of the entire application. With microservices, however, you only need to redeploy the service(s) you modified. This can simplify and speed up the release process” [5].

Thus, the decision was to redevelop, again from scratch, the Annotation Server following a microservice architecture, using a lean technology easy to be deployed on the cloud. Since the Annotator is a JavaScript based app, it was decided to develop the Pundit back-end in Node.js,

⁹ To emphasize this incompatibility, from version 2 the framework has been renamed from AngularJS to Angular.

¹⁰ <https://github.com/net7/pundit-anchoring/blob/master/README.md>

an open-source JavaScript development platform. Also, to make the service more robust, scalable and secure we chose to host it on the Amazon AWS cloud.

The Annotation Server is therefore a collection of services, implemented as Node.js apps deployed on Amazon AWS as Lambda applications¹¹, sharing a part of their codebase, and exposing their functionalities through APIs.

The new Pundit **data layer** has been updated as well. In the old version of the tool, data was stored in an RDF Triple Store, by natively using a semantic Data Model inspired, but not completely compliant, to the W3C Web Annotation Data Model. In order to make searches more efficient, annotation data was also copied to the Apache Solr Text Search Engine.

While a relational database (MariaDB, fully compatible with most renowned MySQL) is still in use to maintain user's data (profile, authentication information, preferences), annotation data is now stored on Elasticsearch.

Albeit the use of Elasticsearch as the primary data source might seem risky at first, the team has considered the following aspects for taking an informed decision:

- there has been a significant effort to make Elasticsearch as reliable as possible, to be safely used as the primary datastore¹²
- Elasticsearch has excellent back-up and snapshot functionalities, which can ensure a high level of reliability to the service. Also, regular extra back-ups, implemented as full export on the file system, have been added to guarantee a supplementary security measure against data corruption and loss
- the Elasticsearch used by the new Pundit is based on a native Amazon AWS service, a choice that improves resiliency and security
- this choice allows the implementation of queries that also exploits text search functionalities of Elasticsearch, making them more useful and powerful.

This general refactoring, both on the Annotator and on the Annotation Server side, suggested to redesign the communication protocol amongst them as well, which is now based on REST APIs that exchange data in JSON format in a quite compact way, reducing therefore the “payload” of the message and therefore saving on bandwidth: in order to further reduce the payload dimension, the APIs use HTTP(S) data compression.

A public search API has been also implemented with the double intent of:

- providing an intuitive and modern way for developers to retrieve annotations by using the GraphQL query language [6]
- to retrieve data in a JSON-LD format, totally compliant with the W3C Web Annotation Data Model [1].

Through this API it is possible to retrieve either public annotations, that is those contained in public notebooks, but also, if the user successfully authenticates, those in her private notebooks. Data is inherently returned in Linked Data format, maximizing its possible reuse.

¹¹ As defined in Wikipedia “AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services.” See https://en.wikipedia.org/wiki/AWS_Lambda

¹² See <https://www.elastic.co/guide/en/elasticsearch/resiliency/current/index.html>

Finally, also the **Pundit App** has undergone a full refactoring, albeit from a user design viewpoint its interface hasn't changed much from the previous version. From a technological perspective this webapp has been implemented in PHP by using the Laravel framework. Its main purpose is twofold: on the one hand it manages user registration and authentication; on the other it provides Pundit users with the web dashboard where they can see, search for and export all the annotations they did on the web and stored in their notebooks.

In particular for authentication, it has been decided to use the JWT technology [7], an open, industry standard for securely representing authentication and authorization claims between applications. It allows the implementation of a real sessionless environment, particularly suitable for a microservice architecture in which horizontal and dynamic scalability of the services is required. Basically after having successfully authenticated, the Annotator receives a JWT token, signed and encrypted by the Laravel service. At each subsequent request to the APIs, the JWT is automatically sent: the AWS API Gateway service, which mediates the requests to the Annotation Server, verifies if the JWT token is valid and only in this case forwards the request to the specific API.

A general overview of the new Pundit architecture is shown in figure 5.

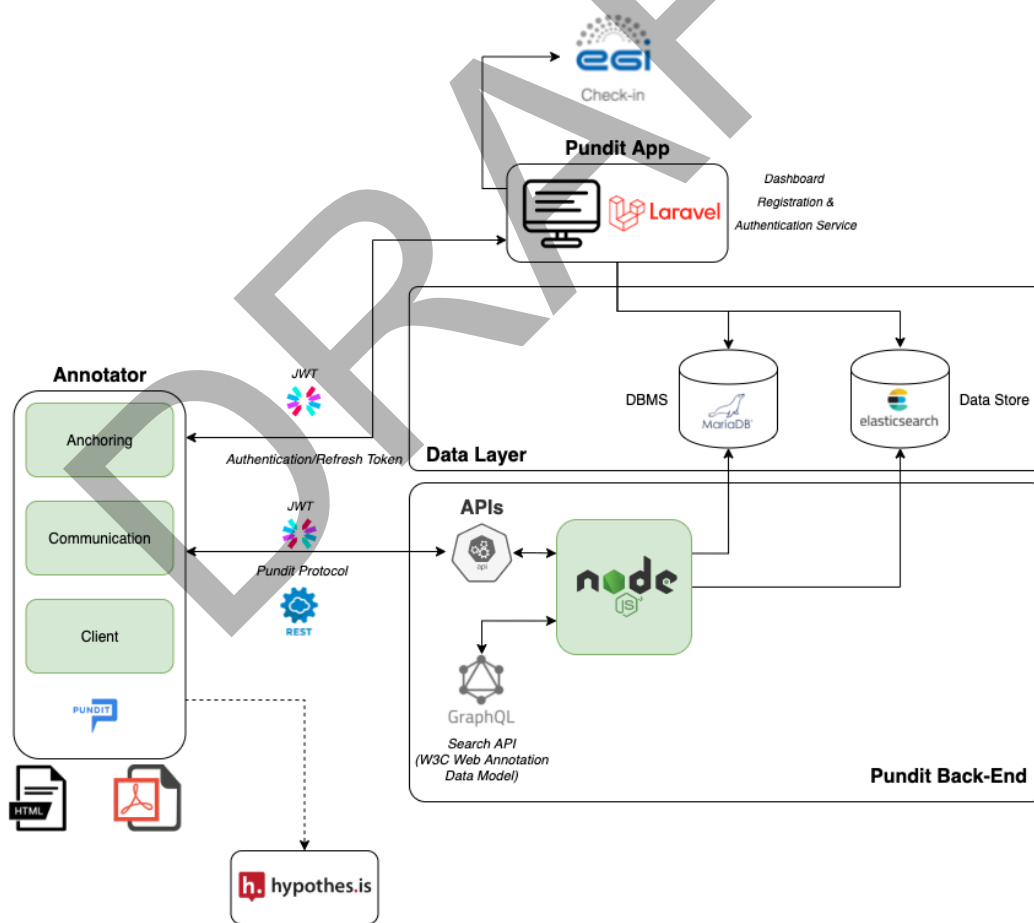


Figure 5 - The new Pundit architecture

3.2 Pundit Annotator

The Pundit Annotator is a web application that can be used as a Chrome browser extension or can be even embedded in any website. It has been developed in the TypeScript language (a strict syntactical superset of JavaScript) by using the Angular¹³ and RxJS¹⁴ frameworks. It is structured in three main modules:

- **Anchoring:** the anchoring module is responsible for matching and highlighting the annotations made by users to the structural elements of a web page (or of a PDF document) and highlighting them. This module is based on the anchoring logic initially designed by the Hypothes.is team for their web annotation tool, a choice that can also ease the interoperability between Pundit and the Hypothes.is annotator. Given its importance, a more detailed description of this module is provided in chapter 3.2.1.
- **Communication:** this module handles all web requests between Pundit's client and the Annotation Server. It uses Axios¹⁵ as its promise-based HTTP client.
- **Client:** it is the module the users directly interact with. Its main component is the sidebar that is visible at the right side of every web page once Pundit is open. Inside the sidebar a user can see all the annotations done in the page, both hers or the public ones, made by other users.

In order to measure the quality of the product and the ease of use of all its parts, the Mixpanel¹⁶ analytics service has been integrated to gather usage data. This allows Net7 to automatically detect any possible issue, also concerning the user experience, and to solve it as needed.

Other external libraries used in the Annotator are Tagify¹⁷ (tagging system for annotations), Draggable¹⁸ (which allows the dragging of the annotation editor inside the browser) and Webpack¹⁹, as the build tool.

3.2.1 The anchoring module

It is a TypeScript rewrite of the following three anchoring packages used in Hypothes.is:

- dom-anchor-fragment²⁰
- dom-anchor-text-position²¹
- dom-anchor-text-quote²².

¹³ <https://angular.io/>

¹⁴ <https://rxjs.dev/>

¹⁵ <https://github.com/axios/axios>

¹⁶ <https://mixpanel.com/>

¹⁷ <https://github.com/yairEO/tagify>

¹⁸ <https://github.com/bcherny/draggable>

¹⁹ <https://github.com/webpack/webpack>

²⁰ <https://github.com/tilgovi/dom-anchor-fragment>

²¹ <https://github.com/tilgovi/dom-anchor-text-position>

²² <https://github.com/tilgovi/dom-anchor-text-quote>

The objective of this module is to provide the user with a simple API to attach web annotations to an HTML or a PDF document.

Several *DOM Anchoring Strategies* have been implemented, including:

- *Fragment*

It is the fastest anchoring strategy if the document structure hasn't radically changed after the annotation was made. It simply applies the "start" and "end" XPath to the current DOM of the web document and attaches the annotation between them.

- *Text Position*

This strategy is useful when the document never changes (e.g. for PDF files). In this case we try to attach the annotation based on global character offsets, then we verify that the matched text is the same as the one stored in the annotation's body.

- *Text Quote*

This is the fallback strategy when both of the above methods fail. It tries to locate the annotated text through a fuzzy search around the expected position in the document. If a "similar" string, within a predefined threshold, is found, the annotation is attached to it.

3.3 Pundit Back-End: the Annotation Server

The Annotation Server implements the business logic for managing users' annotations. It consists of a set of Lambda applications developed with Node.js in TypeScript and deployed on the Amazon AWS Cloud.

Its functionalities are exposed via APIs, in particular to the Annotator. The access to these APIs is mediated by the AWS API Gateway service which is also in charge of verifying the permission policies to invoke them: in fact, if a certain API call is available only for authenticated users, the presence and the validity of the JWT token is verified through an ad-hoc authorizer function (*JWT Authorizer*).

All these lambdas share a set of common libraries, referred to as *Core* in the diagram that follows.

In the next two chapters a description of these APIs is provided, with a dedicated focus on the W3C Search API.

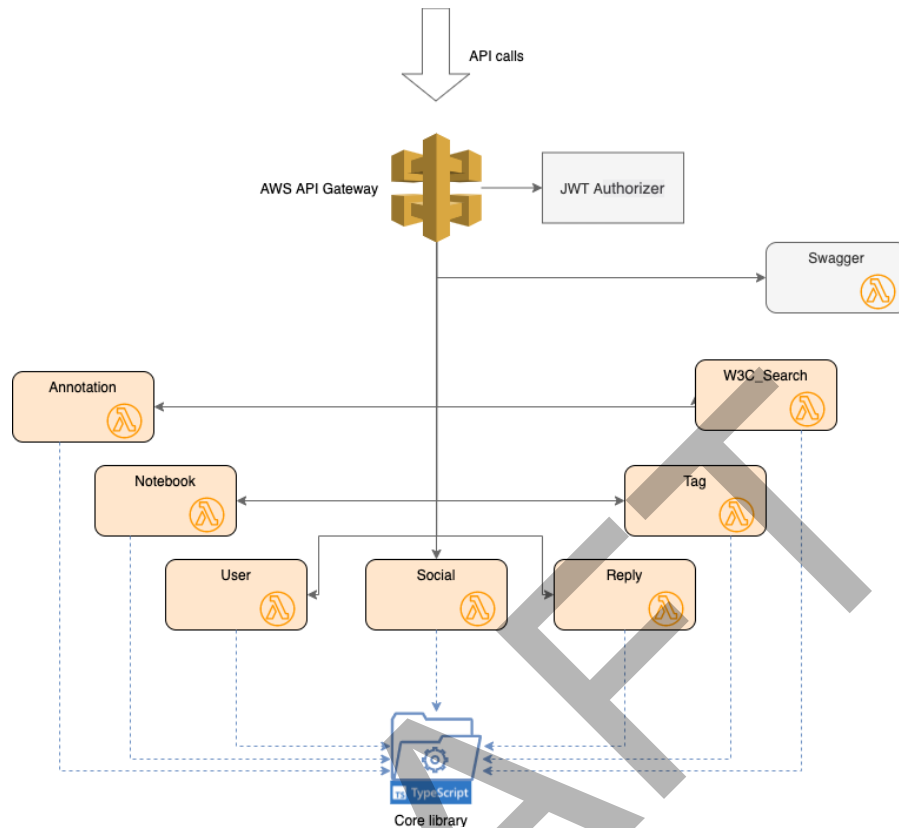


Figure 6 - Annotation Server’s APIs deployed on the Amazon AWS cloud

3.3.1 Annotation APIs

The Annotation APIs are a group of AWS lambda applications that exposes web services to:

- retrieve, save and edit annotations, notebooks, users and social features
- obtain annotation data in different formats (JSON, JSON-LD, HTML) through a URI dereferencing mechanism [8], compliant with the requirements of the Semantic Web.

Moreover the public API has been implemented to provide documentation about the rest of the Annotation Server APIs in a format compliant with the Swagger²³ OpenAPI specification.

In order to invoke the APIs, the user must be authenticated with the Pundit Laravel Authentication service, to obtain a valid JWT token which must be passed with each request.

In some cases the JWT token can be optional: this means that it is possible to invoke some APIs without a token and still get a valid response but limited to “public” data (e.g. to annotations contained in public notebooks).

In the following table the main APIs are schematically enlisted.

²³ <https://swagger.io/>

API Path	HTTP Method	Parameters	Description
annotation	POST	New annotation data	Creates a new annotation and returns its ID
annotation/id	GET	None	Returns the annotation identified by the ID
annotation/id	PUT	Annotation changes	Updates the annotation identified by the ID
annotation/id	DELETE	None	Deletes the annotation identified by the ID
annotation/search	POST	The parameters for searching annotations	Returns the annotations that match the search parameters
notebook	POST	New notebook data	Creates a new notebook and returns its ID
notebook/id	GET	None	Returns the notebook identified by the ID
notebook/id	PUT	Notebook changes	Updates the notebook identified by the ID
notebook/id	DELETE	None	Deletes the notebook identified by the ID
notebook/search	POST	The parameters for searching notebooks	Returns the notebooks that match the search parameters
user	POST	New user data	Creates a new user and returns the ID
user/id	GET	None	Returns the user identified by the ID
user/id	PUT	User changes	Updates the user identified by the ID
user/id	DELETE	None	Deletes the user identified by the ID
social/do	POST	Annotation ID plus new social feature data (e.g. Like/Dislike/Report)	Creates a new social feature associated to the specified annotation and returns its ID
social/undo	POST	The ID of the social feature to remove	Deletes the social features with the specified ID
social/id	GET	None	Returns the social feature identified by the ID

reply	POST	Annotation ID plus new reply data	Creates a new reply associated to the specified annotation and returns its ID
reply/id	GET	None	Returns the reply identified by the ID
reply/id	PUT	Reply changes	Updates the reply identified by the ID
reply/id	DELETE	None	Deletes the reply identified by the ID

The Lambda applications make use of the TSOA²⁴ framework to create HTTP(S) REST endpoints compliant with the OpenAPI v. 3 specification [9].

The source code folder is divided as follows:

- **app**: contains the APIs sources
- **custom-authorizer**: contains the source code of the JWT token validator
- **test**: contains unit tests for the APIs
- **server**: contains the TSOA configurations.

API sources in the app folder are divided as follows:

- **core**: contains the model and the definition of request and response formats
- **presentation**: contains the endpoints. It is the part of the logic that handles requests and responses
- **application**: contains the main Pundit business logic
- **infrastructure**: manages communication with external services, for example with Elasticsearch.

Application deployments of the Annotation Server code is done on the Amazon AWS cloud by using Serverless²⁵, a JavaScript framework that simplifies the creation of serverless applications for different cloud environments. The framework uses configuration files written in YAML to describe the architecture. It is also possible, through the serverless-offline plugin, to have an on-premises solution, useful for local development and debugging.

Finally, the following configuration files are also present in the code repository:

- **serverless.yml**, which defines resources for Annotation Server CRUD APIs and token validation
- **serverless-uri.yml**: it defines the resources needed for dereferencing APIs

²⁴ <https://github.com/lukeautry/tsoa>

²⁵ <https://serverless.com/>

- stage.yml, prod.yml and local.yml, the configuration files for stage, production and development.

3.3.2 W3C Search endpoint

The Annotation Server exposes to developers a flexible, public access, GraphQL-compatible endpoint [6] to easily search and browse annotations created with Pundit, which respects the data model of the W3C Web Annotation specification.

It is quite surprising in fact that this standard doesn't include a description of how annotations can be searched. The W3C Web Annotation Protocol in fact is only concentrated on how to implement standard CRUD operations on back-end servers, but completely misses even the most simple "search annotations" use case, that is "retrieving all annotations done on a certain page identified by this URL".

Another limit of the W3C Web Annotation specification is its verbosity. It makes sense therefore to think about a flexible API that allows the selection of the elements to be returned in the search response. This is the reason why the GraphQL query language has been chosen for implementing this API.

At the same time, in order to encourage reuse, it has been decided to return data in a format completely compatible with the W3C Web Annotation Data Model: in particular the API returns Annotation entities contained in a sequence of Annotation Pages, all formatted in JSON-LD. JSON-LD is a modern RDF serialization format that has been chosen by the W3C research group for the official Web Annotation Data Model recommendation.

This API has been therefore designed with these goals in mind, even if some minor compromises had to be made.

First of all, it doesn't fully respect the constraints imposed by the GraphQL specification, for which the response must be constructed through a map with the mandatory element "data" and optional fields "errors" and "extensions". Since including these elements would violate the Web Annotation Data Model we decided to:

- include them in the GraphQL schema, since they are mandatory for this query language
- ignore them in the response, which is a fully valid Web Annotation Page.

The Search Endpoint is built as part of the Annotation Server and uses the same technologies (TypeScript programming language, tsoa framework for Node.js, deployment as AWS Lambda application), together with the GraphQL.js²⁶ library to validate and execute GraphQL queries.

If user-submitted requests are expressed in a valid format, according to the GraphQL schema, they are translated into Elasticsearch queries. This way, the API retrieves the annotations by leveraging the same search logic used in the Annotation Server APIs and formats them in a JSON-LD Web Annotation Page. Search results are paginated and can be navigated by changing the page parameter of the request.

²⁶ <https://graphql.org/graphql-js/>

The Search Endpoint can be used:

- interactively with a GraphQL client (such as the Altair GraphQL Client²⁷), which also allows the user to easily access and navigate documentation about the request (GraphQL queries) and response formats (JSON-LD Web Annotation Data)
- non-interactively, by sending GraphQL-compatible requests over HTTP(S) (POST requests).

The endpoint accepts three kinds of requests:

1. HTTP POST request with a search GraphQL query as body which is a GraphQL-compatible request over HTTP(S). The response returned is a JSON-LD Annotation Page, as defined in the Web Annotation Data Model, with only the selected fields and containing the annotations that match the search. The response complies with the constraints imposed by the W3C Web Annotation Data Model
2. HTTP GET request. The response will be a JSON-LD Annotation page with all the fields and all the annotations contained in the page identified by an ID. This is a feature that can be exploited to retrieve cached queries.
3. HTTP POST request with an Introspection GraphQL query as body. The response will be a JSON containing the GraphQL Schema Documentation.

By default, the annotations returned to the user with this search endpoint are only those contained in public notebooks. The user can optionally specify in the HTTP request header a valid JWT token generated by authenticating with the Pundit authentication service: this way, the Search Endpoint returns all the annotations of the Pundit user, public or private, plus the public ones made by others which match the specified query parameters.

Here is an example of a valid GraphQL query and of the corresponding response returned by the W3C Search endpoint.

Request:

```
POST https://api.thepund.it/w3c/annotation?page=1
{
  search {
    id
    type
    next
    items {
      id
      type
      created
      creator
      motivation
      generator
    }
  }
}
```

²⁷ <https://altair.sirmuel.design/>

```
body {
  type
  value
}
target {
  source
  selector {
    type
    ... on TextQuoteSelector {
      suffix
      prefix
      exact
    }
    ... on TextPositionSelector {
      start
      end
    }
    ... on RangeSelector {
      startSelector {
        type
        value
      }
      endSelector {
        type
        value
      }
    }
  }
  ...
}
```

Response:

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "urn:urn-5:OGU2ZGM5MTY2Nz1mMjM4ZDQ5OGV1NDA2ZTBkOWZkZTc=:1",
  "type": "AnnotationPage",
  "next": "urn:urn-5:OGU2ZGM5MTY2Nz1mMjM4ZDQ5OGV1NDA2ZTBkOWZkZTc=:2",
  "items": [
    {
      "id": "https://purl.thepund.it/w3c/annotation/pg4MjHsBqgKWRiZgleDd",
      "type": "Annotation",
      "created": "2021-08-28T09:17:20.217Z",
    }
  ]
}
```

```
"creator": "https://purl.thepund.it/w3c/agent/15",
"motivation": "commenting",
"generator": "pundit-client",
"body": {
  "type": "TextualBody",
  "value": "Da usare per risolvere il problema del coldstart?"
},
"target": {
  "source": "https://github.com/juanjoDiaz/serverless-plugin-warmup",
  "selector": [
    {
      "type": "RangeSelector",
      "startSelector": {
        "type": "XPathSelector",
        "value": "/div[4]/div[1]/main[1]/div[1]/div[1]/div[1]/h1[1]/span[1]/a[1]"
      },
      "endSelector": {
        "type": "XPathSelector",
        "value": "/div[4]/div[1]/main[1]/div[1]/div[1]/div[1]/h1[1]/strong[1]/a[1]"
      }
    },
    {
      "type": "TextPositionSelector",
      "start": 3792,
      "end": 3841
    },
    {
      "type": "TextQuoteSelector",
      "suffix": "\n \n \n\n\n\n      \n\n      \n\n",
      "prefix": " \n      \n \n \n ",
      "exact": "juanjoDiaz\n \n /\n \n serverless-plugin-warmup"
    }
  ]
}
},
{
  "id": "https://purl.thepund.it/w3c/annotation/7Wv3gnsBS8MhFsgCljva",
  "type": "Annotation",
  "created": "2021-08-26T14:57:49.270Z",
  "creator": "https://purl.thepund.it/w3c/agent/7214",
  "motivation": "commenting",
  "generator": "pundit-client",
```



```
"body": {
  "type": "TextualBody",
  "value": "yes but how would that be arranged - visas - what countries"
},
"target": {
  "source":
  "https://www.theatlantic.com/p/a/2013/06/the-case-against-universal-service/2/",
  "selector": [
    {
      "type": "RangeSelector",
      "startSelector": {
        "type": "XPathSelector",
        "value": "/div[1]/main[1]/article[1]/section[1]/div[11]/p[1]"
      },
      "endSelector": {
        "type": "XPathSelector",
        "value": "/div[1]/main[1]/article[1]/section[1]/div[11]/p[1]"
      }
    },
    {
      "type": "TextPositionSelector",
      "start": 19771,
      "end": 19885
    },
    {
      "type": "TextQuoteSelector",
      "suffix": "10) Of course, a cosmopolitan, c",
      "prefix": " and I don't think it should -- ",
      "exact": "wouldn't the nation and the world benefit more? I think so. "
    }
  ]
  ...
}
```

Users can optionally specify the following filters in the search:

- **annotationId**: to search only annotations with a specific id
- **target_source**: to search only annotations made on the specified URI
- **motivation**: to search only annotations with a specific *motivation*, as indicated in the W3C Web Annotation Data Model specification. Valid values for Pundit are: commenting, highlighting, linking and tagging.
- **collectionIds**: to search only annotations contained in one or more annotation collections (for Pundit a collection is a Notebook)

- **dateFrom, dateTo:** to search only those annotations that satisfy the specified date condition
- **size:** to limit the number of annotations returned.

Here is an example of a query to search all annotations with a specific target and created from a certain date onwards:

Request

```
POST https://api.thepund.it/w3c/annotation?page=1
{
  search (target_source:"https://amazon.com", dateFrom: "2021-06-01T12:00:00Z"){
    id
    type
    next
    items {
      id
      type
      created
      creator
      motivation
      generator
      body {
        type
        value
      }
      target {
        source
      }
    }
  }
}
```

3.4 Pundit Data Layer

The Pundit data layer uses a “polyglot persistence” approach, in which “a variety of different data storage technologies (is used) for different kinds of data” [10].

In particular the main primary data source, where annotation data is maintained, is managed on Elasticsearch, while user data, in particular those that might change often (profile information, authentication credentials) are contained in a MariaDB relational database.

For both Elasticsearch and MariaDB the corresponding AWS services are used (respectively Amazon OpenSearch Service and Amazon Relational Database Service).

Figure 8 and 9 show the logical views of the data models on these two repositories.

Starting from Elasticsearch, here data is contained in these five different “document types”:

- **User:** here the user data that is needed when showing an annotation is maintained, and therefore copied from the corresponding table of the MariaDB RDBMS. This includes the full name and the url of the thumbnail depicting the user’s avatar.
- **Notebook:** as said, the notebook is the container of the annotations done with Pundit. Therefore every user has at least one notebook and every notebook has an *owner*. Amongst the next steps for Pundit, it has been planned to implement the possibility for users to share notebooks with others, either in *read-only* or also in *write mode*.
- **Annotation.** All annotations are stored here. Every annotation has exactly one creator (*userId*) and is contained in only one notebook (*notebookId*). The structure of this document has been designed with respect to the W3C Web Annotation Data Model which states that an annotation should be generally divided in three parts, as shown below in figure 7:
 - a general set of metadata that describes the *annotation*
 - the *body* with the real “content” of the annotation
 - the *target*, which identifies the “object” that has been annotated, in particular a text fragment of a web document or the whole document itself.
- **Tag** is used to maintain the “free-text categories” chosen by a user when annotating. To simplify the tagging process, a user receives suggestions, through autocomplete, of the various tags she has previously used.
- **Social**, for the social interactions (replies, likes/dislikes/report) done on a specific annotation.

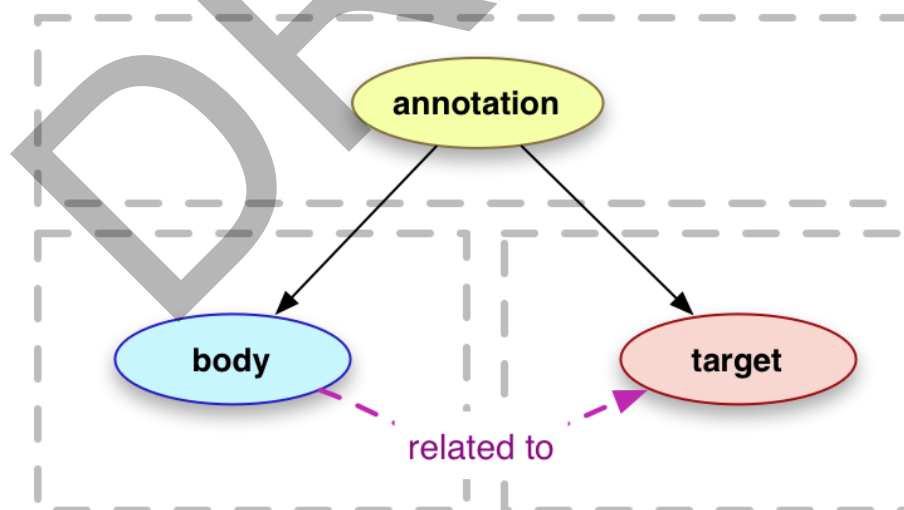


Figure 7 - Structure of an annotation according to the W3C Web Annotation Data Model

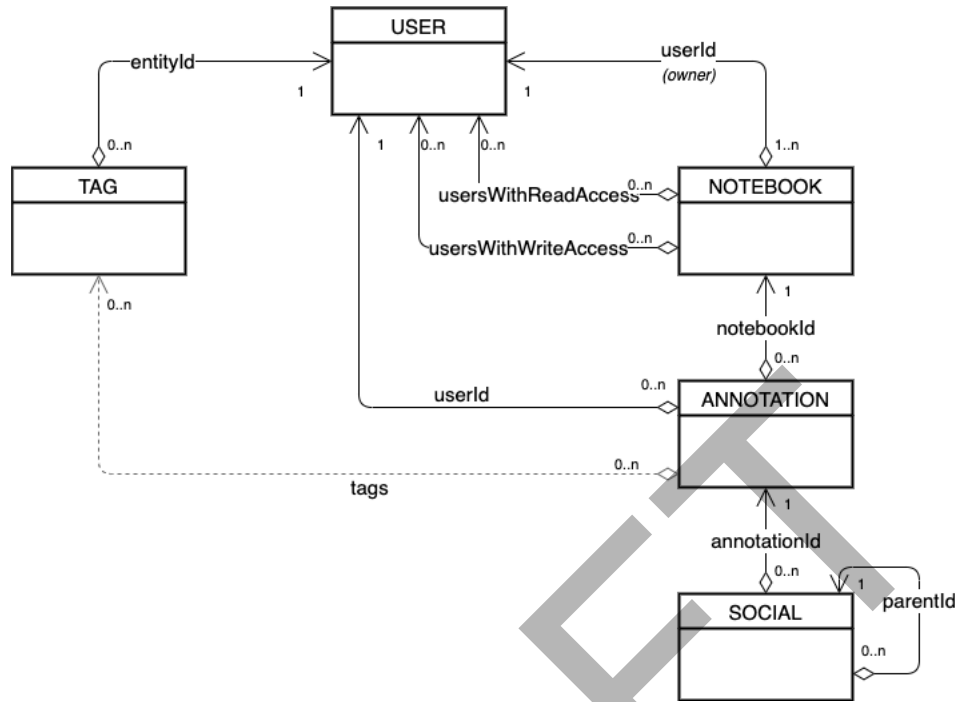


Figure 8 - Pundit data model of the main data store on Elasticsearch

The Entity Relationship Diagram below shows the entities maintained in the MariaDB RDBMS.

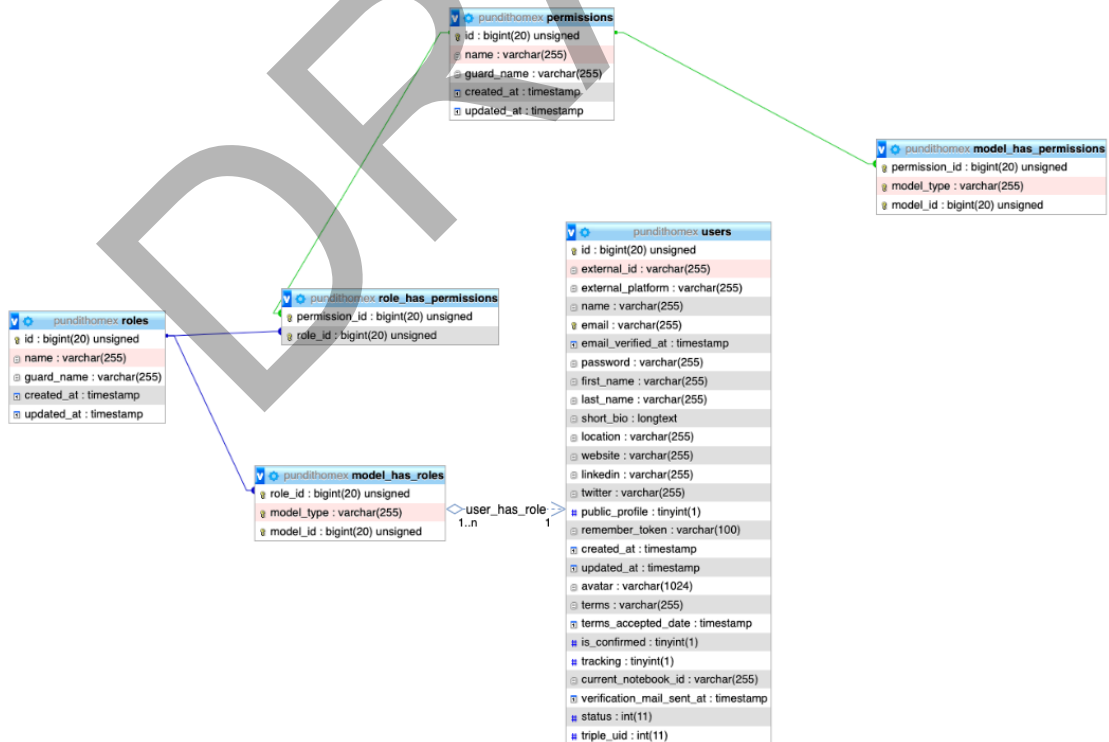


Figure 9 - MariaDB data model in Pundit

Basically here (see figure 9) three kinds of data are maintained:

- **profile information** describing the user in fields, such as *first_name*, *last_name*, *email*, *short_bio*, *location*, *website*, *linkedin*, *twitter*,... of the **users** table. *public_profile* is a boolean that allows to specify if the user wants to show to others the full profile data or not
- information about **roles and permissions**. At present, two possible roles for users have been defined, *basic* and *admin* (see tables **roles**, **model_has_roles**, **permissions**, **model_has_permissions** and **role_has_permissions**). More roles will be defined with the introduction of the Freemium commercial model for Pundit
- **data used for authentication**, in particular if the user has directly registered on Pundit (and in this case her credentials are maintained in the **users** table) or if an external Identity Management system is used. In the latter case the fields *external_id* and *external_platform* of the **users** table are not NULL: at present Pundit supports Google, Facebook and EGI AAI check-in for registration and subsequent authentication of users
- **user preferences** which can change frequently, for example the ID of the current notebook to use when annotating (*current_notebook_id* field). They too are maintained in the **users** table.

3.5 Pundit App

This component has been developed in PHP with the Laravel²⁸ 8 and the Livewire²⁹ frameworks, while the MariaDB database, as described before, is used to store user data.

The deployment, as for the rest of the Pundit architecture, is done on Amazon AWS.

The Pundit App serves two main purposes:

- implementing the dashboard, where users can easily access their annotations
- managing users' authentication: it is therefore the Identity Management service of Pundit.

The Pundit App code is organised in:

- a model done with simple PHP objects that represent the annotation, the notebook and the social feature (like/dislike/report/reply)
- a controller to customise Laravel's authentication by extending the *AuthenticatesUsers trait*³⁰ with the business logic needed for logging-in the Pundit user
- a specific model for the user, that generates the JWT token needed to call the Annotation Server APIs. Moreover it updates the user data contained in Elasticsearch
- a controller for the login API used by the Pundit Annotator. This component and the controller cited before share some methods, such as those needed to support authentication via external services (Google, Facebook or EGI AAI Check-In)
- a controller for the annotation and for the notebook view

²⁸ <https://laravel.com/>

²⁹ <https://laravel-livewire.com/>

³⁰ A trait is a mechanism for reusing code in a software program developed in languages like PHP

- a model class that implements all the business logic needed to call the Annotation Server APIs to retrieve annotations and notebooks
- Livewire components for some of the interactions on the front-end views, especially the ones needed for the notebook CRUD.

When the user accesses the dashboard, she can see a view of all her notebooks (see image below): by clicking on one of them, she accesses all the annotations stored in it.

A search service, with a faceted filtering system, allows to query and retrieve the user's annotations by invoking the Annotation Server APIs.

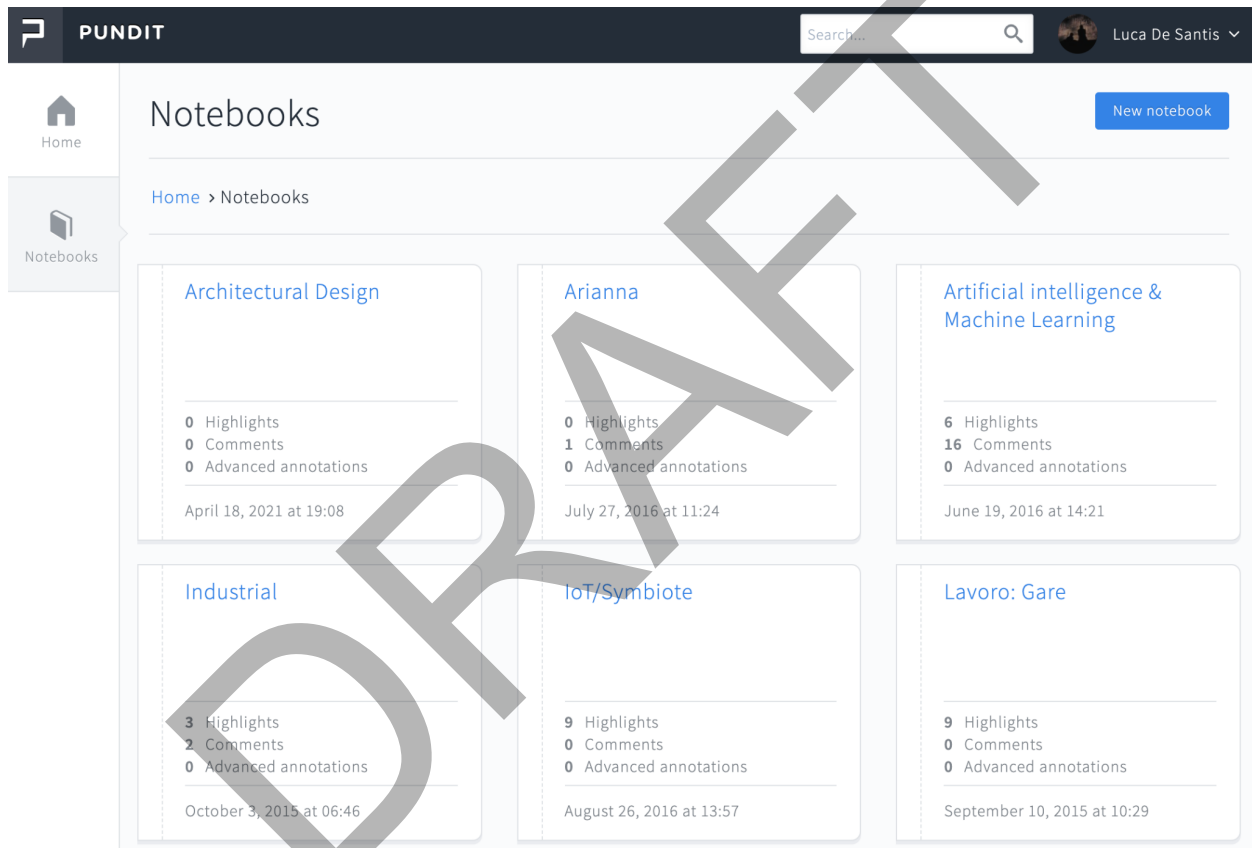


Figure 10 - The Pundit App with the list of the user's notebooks

As far as the Identity Management functionality is concerned, as mentioned before, authentication in Pundit is based on a JWT token.

We have managed to add some custom claims on the token, in order to pass specific information between the Annotator and the APIs, as shown below:

```
{
  'id' => <id>,
  'username' => <username>,
  'thumb' => <avatar>,
  'iss' => <url_of_issuer>,
  'aud' => 'pundit-client',
  'has_accepted_terms' => <true_false>,
  'is_verified' => <true_false>,
  'current_notebook' => <current_notebook_id>,
  'triple_uid' => <triple_uid_if_connected>
}
```

After its generation on the Pundit App, the token is signed by the service's private key, while the Annotation Server (or better, the JWT Authorizer integrated with the AWS API Gateway service) validates it with the corresponding public key. If the token is valid, the API returns the information required, otherwise it returns an error.

The token's duration is set to only one minute, but it can be refreshed as many times as needed (unless the user logs out) for 60 days. The refresh token needed to generate a new JWT token is stored in a HttpOnly cookie, which is not accessible through client code, to improve security. It is important to point out that this strategy is completely compliant with the JWT design patterns for enforcing security on a microservice-based architecture.

Utility shell commands have been also implemented in Laravel, namely:

- `checkVerification`: it is periodically invoked through a cronjob in order to send an email to the users who haven't confirmed their email address yet
- `importSymfonyUsers`: one time script created to import the old Pundit users in the new system.

Finally, an administration back-end interface, implemented with Backpack³¹, is also available to manually manage users if the need arises.

4 | INTEGRATION STRATEGIES OF THE SERVICE IN GOTRIPLE

Finding the right strategy to integrate the Open Annotation Tool in GoTriple has posed some challenges, both from a technical viewpoint and from a usability perspective.

To start with, Pundit is an "external" Innovative Service that (as others, like the Trust Building System) exists in an independent way in respect to GoTriple. Pundit, which can be used only after registration, has been in use for almost a decade and already has a small yet significant

³¹ <https://backpackforlaravel.com/>

base of registered users. It provides its own methods for registering to the service: through direct registration or by using external identity providers like Google, Facebook and the EGI AAI Check-in.

GoTriple will be accessible also to anonymous users, while certain advanced services (e.g. the possibility to reclaim as her/his own the profile of a researcher) will be available only after registration.

The fact that both systems have integrated the EGI AAI Check-In service can enhance the general usability, since it can allow single sign-on when passing back and forth amongst the two platforms. However, it doesn't solve the fact that their user bases are completely independent. As a consequence, their possible integration can be only accomplished through a well-defined federation scenario.

It is important to point out that this need is not limited to Pundit. Some other innovative services, that we can define as "Federated Services", share the same problem, in particular the Trust Building System (developed in WP5 task T5.3) and possibly other third party applications that will be selected as a result of WP5 task T5.1.

The strategy that has been identified, and described herein, defines a standard for integrating these Federated Services within GoTriple.

It is based on a minimum set of requirements which include:

- the federation of accounts amongst the two systems
- the visualization of the Federated Services notifications in the MyGoTRIPLE page.

The first point is about allowing GoTriple users to create a link with their Pundit accounts. This is the necessary first step to enable any other (and more sophisticated) integration scenarios.

Starting from the MyGoTriple personal page a user can select the option to link her Pundit account: after that, she will be redirected to a page on Pundit (and more generally on the Federated Service side) to authenticate and to confirm the wish to link these two accounts. In case of success, the user is redirected to the MyGoTriple page and a confirmation message is shown.

In order to accomplish this link from a technical viewpoint it is necessary to exchange the user's ID on these two systems, while ensuring the maximum privacy and security.

The solution that has been envisioned is to use a JWT token with the user's credential to be passed amongst GoTriple and Pundit. The needed passages, as shown in figure 11, are explained in the following list:

- the user clicks on a "Add Pundit" button on her MyGoTriple page
- the GoTriple front-end invokes an API on its back-end to receive a JWT token with the user's TRIPLE ID, encrypted and signed with the GoTriple private key
- the GoTriple front-end commands a redirect to a special Pundit page, adding the JWT token as parameter
- Pundit decrypts the JWT token and let the user authenticate
- After the successful authentication on Pundit, the user needs to confirm her wish to link her Pundit and GoTriple accounts

- After her confirmation, Pundit stores locally the user's TRIPLE ID. Then generates a new JWT token with the user's Pundit ID, encrypts and signs it with the GoTriple public key
- Pundit then commands the redirect on MyGoTriple passing the JWT token as parameter
- The MyGoTriple front-end invokes an API on its back-end with this JWT token. This API decodes the token and stores the Pundit ID of the user in the GoTriple profile.
- A confirmation message is shown to the user.

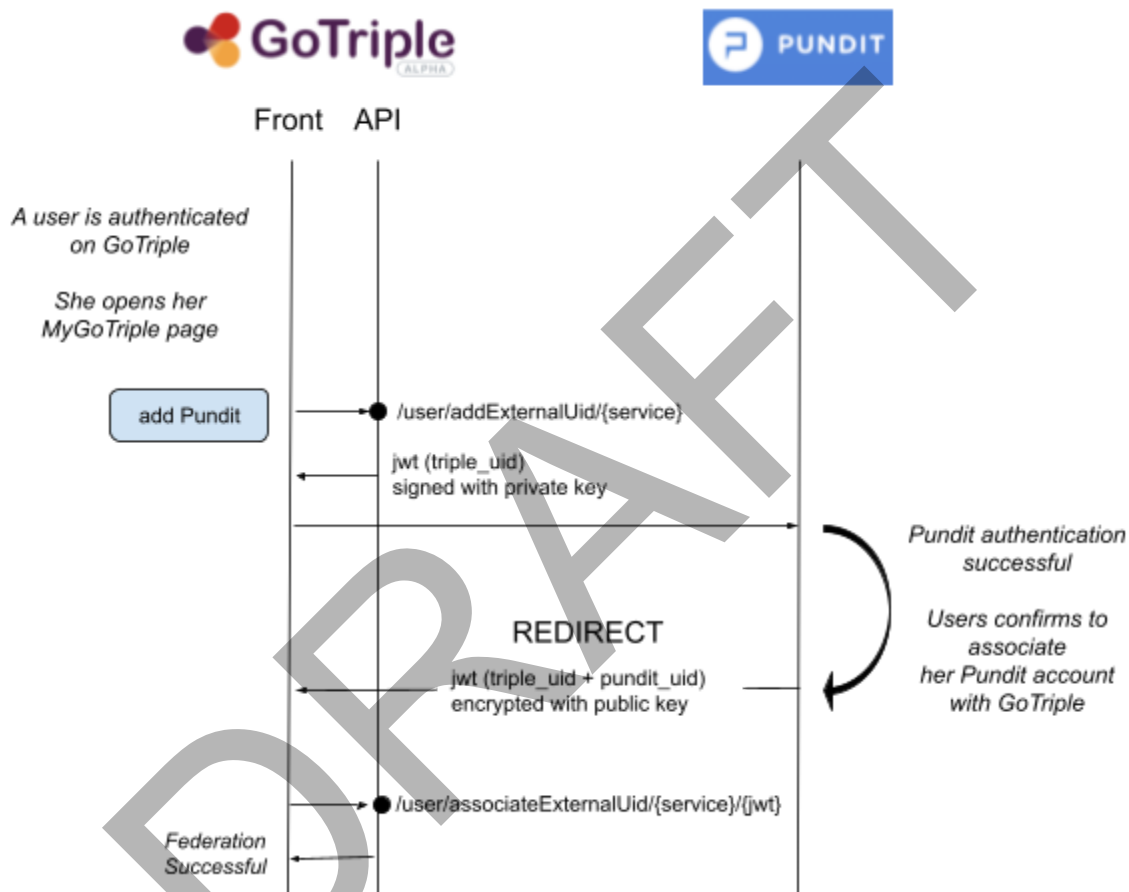



Figure 11 - Linking user accounts between GoTriple and Pundit

After the connection of accounts has been instantiated, it is possible to implement the actual integration amongst the services.

In particular it becomes possible to visualize specific notifications sent by Pundit (and more generally speaking, by a Federated Service) on the MyGoTriple page. This is illustrated in figure 12.

My GoTriple

Overview **3** Messages Collections **5** Recommendations **4** Saved Searches



Massimiliano Spinosa
Università di Pisa, Department of Earth Sciences
GoTriple member since: 17 Nov 2020

Profile completion 70% complete

Users who complete the profile significantly increase their chances for discovery and collaboration. [Update Profile](#)

! We have found publications that may belong to you. [Add them to your profile now.](#)

What's new All Feeds

- TBS** 8 Sep 2021 - 9:45
Steve Horowitz has commented on your post
[Go to the post](#)
- 6 Sep 2021 - 15:31**
Alan Brinkmann has replied to your message.
[See reply](#)
- TBS** 4 Sep 2021 - 11:43
Leonardo Vitali has replied to your comment
[Go to the post](#)
- TBS** 4 Sep 2021 - 9:12
Martin Cooper introduced you to David McLahan
[Join the conversation](#)
- 2 Sep 2021 - 7:45**
Philippe Gautret likes your Pundit annotation on "Femicide as act and process: a geography of ..."
[See annotation](#)
- TBS** 1 Sep 2021 - 12:45
Leonardo Vitali featured you!
[See it on the TBS](#)
- 24 Ago 2021 - 7:00**
Eric Thomson sent you a message.
[See Message](#)
- 24 Gen 2021 - 7:00**
"Femicide as act and process: a geography of ..." was cited in a publication.
[See Citation](#)
- 23 Gen 2021 - 18:00**
Robert Gilbert started following you.
[See user's profile](#)
- 24 Gen 2021 - 7:45**
Philippe Gautret has invited you to collaborate on The spatial heterogeneity of factors of femicide... with Pundit
[Go to document/page](#)

Figure 12 - MyGoTriple "What's new" page with Pundit notifications

GoTriple does not make any assumption about the "semantics" of these notifications: it is the responsibility of Pundit and of the other Federated Services to notify the Discovery Platform with the "messages" to visualize in the MyGoTriple page.

When a specific event occurs that requires a user notification, a Pundit service of the Annotation Server checks if the user is also a GoTriple user. In this case it sends a message to the notification endpoint implemented on the GoTriple side.

This notification mechanism is still under development at the time of writing.

The list of events that Pundit will notify to GoTriple, for all those users who reconciled their accounts, follow.

Notification	Description
Reply to annotation by user	A user has replied to one of your annotations
Like/Dislike/Report to annotation by user	A user has liked/disliked or reported one of your annotations.
Mention in comment <i>(not available at the time of writing)</i>	A user has mentioned you in a comment.
Invitation to collaborate on document <i>(not available at the time of writing)</i>	A user has invited you to collaborate with Pundit on a document or page
New public annotation on a page you annotated	A user has added a new public annotation to a page that you already annotated

5 | DESCRIPTION OF THE WORK DONE IN TASK T5.5

5.1 A full architectural refactoring

As said, Pundit has a long history of development, since its very first release dates back to almost ten years ago (see [11] and [12]). When the TRIPLE project started, it already had a relatively significant amount of users (almost 4.400 who annotated at least once with this tool), although their actual number was decreasing.

Several were the issues that affected the technologies behind Pundit at the time and consequently the experience of the users.

First and foremost, the Annotator was based on the obsolete AngularJS technology; also, from time to time it didn't work, partially or even at all, on modern web sites.

An example of how badly the "old" version of Pundit behaved on some websites is illustrated in the image that follows, which shows on the Isidore.science web site how the behaviour of the service was not the one to be expected.



Documents ▾

Comment

Highlight

Laibach : danse avec le totalitarisme

This UX component...

Fiche du document

Auteur

Jean-Louis Vaxelaire

Date

16 juin 2020

Résumé

Ce qu'on appelle la **musique rock se divise** pour les spécialistes en être fortement opposées à d'autres. Les reprises sont courantes c des groupes qui ne jouent rien d'autre, mais elles relèvent le plus ne sont pratiquement jamais dépassées (il n'existe qu'une poignée et inversement). La reprise ...

...should appear here!

Figure 13 - An example of a wrong anchoring behaviour on a modern website by the old Pundit

In addition, the back-end of the service, the *Annotation Server*, was based on a monolithic architecture developed in Java which had become very hard to maintain and inefficient in terms of performance. It consisted of a monolithic Java Servlet that encapsulated the whole business logic for the annotations management. A large part of its codebase was very old with some newer parts that have been updated during the course of the StoM research project. The newer parts of the Annotation Server code were developed to cover two main requirements:

- providing the support for the W3C Web Annotation Data model. The former version of the Annotation server used an older version of the standard (Open Annotation).
- managing the indexing of annotations in the Solr search server, to support faster and more flexible queries.

From a deployment viewpoint, the Annotation Server and Solr ran as Java servlets on Apache Tomcat.

Also data management strategies, based on an RDF Triple Store (Ontotext's GraphDB) where annotations and notebook data were stored, proved not to be optimal. It used a semantic model inspired, but not perfectly compliant, with the W3C Web Annotation Data Model. As a matter of fact the choice of using an RDF graph database didn't make sense, because none of the specific functionalities that this kind of technology guarantees (e.g. graph navigation through data, inferencing/reasoning, etc.) was used (or needed) in Pundit.

The Annotation Server exposed all its functionalities through a REST endpoint. In particular the protocol used by the Annotator to communicate with the APIs was really redundant, with a lot of duplications and unnecessary data inserted into the payload. The difference between what was sent and what was really useful is demonstrated in figure 14.



Figure 14 - The redundancy of the old Pundit protocol

The system at the beginning of the TRIPLE project presented the architecture shown in the diagram that follows.

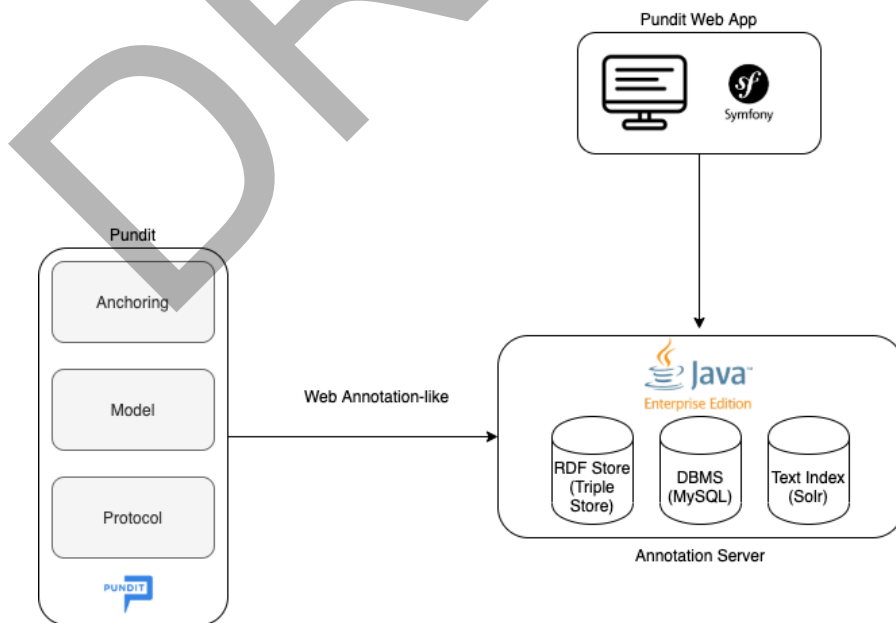


Figure 15 - The architecture of the previous version of Pundit

A list of the main technical problems that this architecture encompassed is summarised in figure 16.

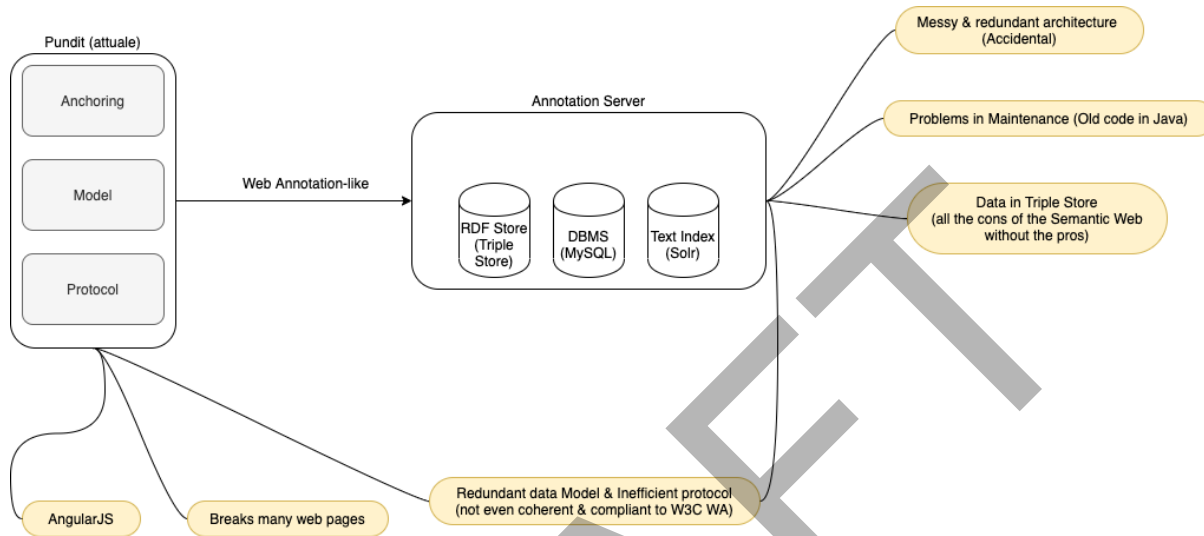


Figure 16 - A synthesis of the main problems that afflicted the previous version of Pundit

All this explains the reason why a full refactoring of Pundit, to arrive at the current version described in chapter 3, was needed.

When the new architecture was ready in March 2021, a full import of the users of the old Pundit and their annotations was performed, in order to allow them to use the new system.

However, the work in task T5.5 in this first part of TRIPLE didn't stop here, since new features have been implemented in Pundit while more are in the roadmap for the upcoming months. Some of the main new functionalities are described in section 5.2, while chapter 6 outlines the next steps foreseen for Pundit, also regarding the technical developments of the service.

5.2 New functionalities in Pundit

5.2.1 Tagging support

This new functionality was introduced as an agile way for users to classify their annotations. In this sense, it can be considered as the lightest form of semantic annotation that Pundit provides.

Users can add tags to their comments, to their semantic annotations but also can create an annotation only to add tags.

The tag is an alphanumeric string of up to 128 characters that the user can freely create. In order to limit the number of tags, an autocomplete feature is provided in the Annotator.

As soon as the user activates Pundit, all her tags are automatically loaded and, through autocomplete, suggested when tagging.

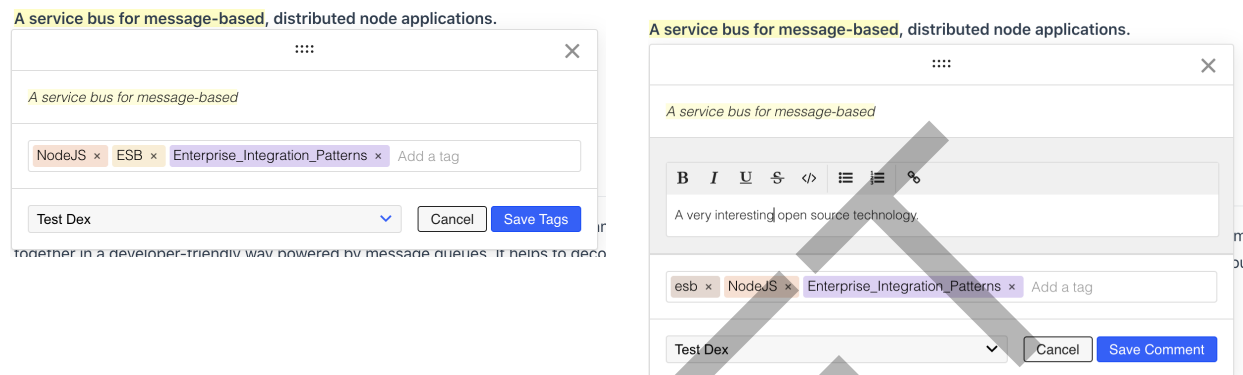


Figure 17 - Tagging in action in Pundit

5.2.2 Semantic annotations

While this is not a new feature “per se”, the new version of Pundit offers a completely redesigned paradigm to create an annotation in the form of a semantic triple.

This triple is formed by:

- subject
 - selected text in the current document
 - whole document/web page
- predicate
 - defined in an ontology and selected from a drop-down list
- object
 - literal: free text
 - URI: a correct URI (not necessarily HTTP urls) inserted as free text. It is assumed that correct URIs are only those with these possible prefixes: http, https, urn, mailto, file
 - date: year or month or day or hour
 - URI of a LOD repository, selected through a search by using a specific connector (e.g. Wikidata, Geonames, ...)
 - a class, defined in an ontology and identified by a URI

- an entity, defined in an ontology and identified by a URI
- text fragment, of the current or of other documents.

The current implementation of Pundit already supports the possibility to create triples with literals, URIs and dates, while in the upcoming months it will be expanded to allow the more complex forms of semantic annotations described above.

It is important to point out that the full support of semantic annotations, including the possibility for users to access already defined advanced ontologies but also to define their own, might be one of the possible premium features, available only to paying customers of Pundit.

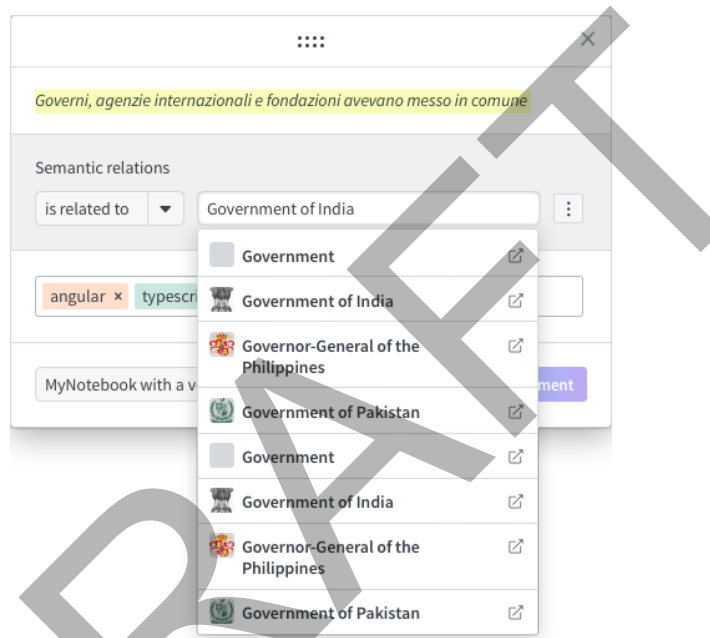


Figure 18 - A mock-up of the future implementation of semantic annotations in Pundit

6 | NEXT STEPS AND EVOLUTIONS

A significant amount of work has been carried out in T5.5 in this first part of the TRIPLE project. On the one hand, a new version of Pundit has been released, to solve all the “technical debts” of the previous version. Moreover, its integration with the GoTriple platform has been implemented and it is about to be released together with the official launch of the Discovery Platform.

Of course the work doesn’t end here, since there are multiple, and diverse, activities to carry on in the following months.

In particular, three are the areas in which the effort will be focused in T5.5:

- new Pundit functionalities
- community building

- exploitation models and sustainability.

As far as the first point is concerned, the T5.5 team is working on several **new Pundit functionalities**, which are expected to be released, in an incremental fashion, as soon as they are ready.

First of all, there is the *finalization of the Semantic Annotations* implementation, as described in the previous chapter. We plan to offer multiple functionalities in this area including: the possibility to let users choose from multiple ontologies, the definition and management of personal ontologies and the availability of automatic connectors for several Linked Data repositories.

Annotation of PDF documents and not only HTML pages will be soon available, together with the possibility to *share notebooks*, in read-only or read-write modes, with other users.

Next, the *FeedThePundit Proxy service* will be reactivated, to allow the use of Pundit with browsers different from Google Chrome and also with mobile devices.

We plan to allow *interoperability with annotations made with the Hypothes.is web annotator*: this will allow Pundit users to see and interact with annotations done with this very popular web annotation tool, in addition to those created with Pundit.

Finally, *more advanced integration strategies with GoTriple* will be designed and implemented, also according to the feedback received by the users of both systems.

Community Building will assume multiple forms: on the one hand there is the need to make the number of Pundit active users grow, also by creating communication and promotion strategies within TRIPLE, involving therefore the WP8 team.

Moreover the proper distribution and subsequent promotion of the source code of Pundit as open source must be properly addressed. At present a significant part of its codebase is freely available on Github through an AGPL 3.0 license: in the perspective of defining a Freemium commercial model, it makes sense to separate some components and to distribute them in a private form, while all the basic modules, that could allow developers to build the code and create a perfectly complete annotator based on Pundit, should be left freely accessible to the public.

The latter point introduced the final part of the work that will be performed in T5.5, which is related to the identification of **exploitation models** and, more generally, on how to reach financial **sustainability** for Pundit. In this regard, discussions have already started with the WP7 team.

Two are the most interesting actions identified so far: as said, a Freemium model seems to be the most feasible solution for Pundit. Its users therefore will continue to freely annotate the web, while through the payment of a fee, they will be able to exploit all the advanced functionalities offered by the service (possibly with different levels of use, according to the amount of money that is paid - e.g. a silver/gold/platinum level).

Also, Net7 is currently evaluating the option to onboard Pundit in the EOSC Catalogue, in the hope to give it a boost in visibility and adoption. At present what has delayed this process is the definition of the exact commercial model for the service. In fact, this is an information that must

be specified in the “Resource Access Mode” attribute of the EOSC Catalogue, even if none of the possible values (Free, Free conditionally, Peer-reviewed, Paid, Other), but the generic “other”, covers exactly the typical Freemium scenario.

As shown, there are multiple activities still to perform in T5.5, focused on the improvement of the Pundit user experience, the increase of its adoption and, finally, the definition of a strategy for its sustainability: a full update of this ongoing work will be provided in the D5.7 deliverable (*Additional services updated*) at month M40.

DRAFT

7 | REFERENCES

- [1] W3C Web Annotation Data Model specifications, <https://www.w3.org/TR/annotation-model/>
- [2] SEMLIB - Semantic tools for digital libraries, European Research project, 2011 - 2012, <https://cordis.europa.eu/project/id/262301>
- [3] StoM - SemLib to Market, European Research project, 2014 - 2016, <https://cordis.europa.eu/project/id/606010>
- [4] Kiryakov A. et al., *Semantic Annotation, Indexing, and Retrieval*, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 2, Issue 1, pp. 49-79, 2004
- [5] Fowler M. et al., *Microservices*, 2014, <https://martinfowler.com/articles/microservices.html>
- [6] GraphQL - A query language for your API, <https://graphql.org/>
- [7] JSON Web Token (JWT) RFC, <https://datatracker.ietf.org/doc/html/rfc7519>
- [8] DereferenceURI, W3C, 2012, <https://www.w3.org/wiki/DereferenceURI>
- [9] OpenAPI v.3 specification, <https://spec.openapis.org/oas/latest.html>
- [10] Fowler M., *Polyglot Persistence*, 2011, <https://martinfowler.com/bliki/PolyglotPersistence.html>
- [11] Morbidoni C. et al., *Pundit: Semantically structured annotations for web contents and digital libraries*, Proceedings of the 2nd International Workshop on Semantic Digital Archives, 2012
- [12] Fonda, S. et al., *Pundit: Augmenting web contents with semantics*, Literary and Linguistic Computing n. 28, 2013