OLKi

Open Language and Knowledge for Citizens

Université de Lorraine, LORIA laboratory
Nancy, France

Pierre-Antoine Rault <par@rigelk.eu>
Christophe Cerisara <cerisara@loria.fr>
Table of contents

Project Statement

Related Works

- Reddit/Twitter 6
- ResearchGate, Mendeley, Academia 7
- Academic Torrents 8
- The Dat Project & Science Fair 9
- Other Open Source Platforms 10

Technical Study
Project Statement

In a time of increasing distrust and defiance of citizens towards artificial intelligence, the scientific community is not left untainted by the scandals of the industry and must provide an answer to a justified concern of the society. OLKi want to take part in that nascent effort by ensuring datasets and particularly linguistic datasets that are the essential resource of machine learning techniques can be easily hosted and interacted with for the whole scientific community.

The OLKi project will connect researchers, scientists and citizens through the Fediverse. To do so it will extend the ActivityStreams language to support scientific research datasets and their related activities.

The Fediverse is originally a social-network-oriented text communication platform that has emerged as a citizen initiative in 2017. It has quickly grown, both in number of users (more than 2.5 million users in November 2018) and supported functionalities, including for instance blog-like platforms (Plume), image sharing (Pixelfed), video sharing (peertube), audio sharing (funkwhale), etc. all while maintaining the basics for compatibility and interoperability (Pleroma being one of the main examples). One of the main strengths of the Fediverse is that all of these features are interconnected together through open standards, enabling users of one feature to access other features as well. A major side-effect is that web applications don’t have to compete with each other but can understand each other (with a degree of compatibility). It might seem insignificant since it just amounts to the classic gains of using open standards. However in the case of social interactions, it allows even small and novel software to free themselves from Metcalfe’s law by gaining support from existing communities.

The OLKi project will further enhance the Fediverse with a new functionality oriented towards scientific communication and research results dissemination. This will fill the current need of a dedicated global and open communication platform between researchers, but also between scientists and citizens. In that sense it will be more than a platform to host datasets and provide features that current dataset hosting platforms lack.
Related Works
Beyond science influencers, a different form of scientific debate.

The temporal, attention-grabbing nature of Twitter posts makes them ill-suited for nuanced, in-depth scientific discussions.

Twitter allows users to post short messages (originally up to 140 characters, increased to 280 characters since November 2017) that can be read by any other user. Users can elect to follow other users whose posts they are interested in, in which case they automatically see their followees’ tweets; conversely, users can be followed by other users, in which case their tweets can be seen by their followers. No permission is needed to follow a user, and reciprocation of following is not mandatory. Tweets can be categorized (with hashtags), repeated (retweeted), and shared via other social media platforms, which can exponentially amplify their spread and can offer links to websites, blogs, or scientific papers.

There are scientific advantages to using digital communication technologies such as Twitter. Scientific users describe it as a means to stay abreast of new scientific literature, grant opportunities, and science policy, to promote their own published papers and exchange ideas, and to participate in conferences they cannot attend in person as “virtual delegates”. Twitter can play a role in most parts of the life cycle of a scientific publication, from making connections with potential collaborators, to collecting data or finding data sources, to dissemination of the finished product.

Instead of waiting for the old letters to the editor, scientists can go to Twitter for rapid critique of their research. Gathering papers on a subject is also seemingly easy, with crowdsourced scraping of the latest, most noteworthy papers of a field due to the number of scientists naturally on Twitter.

"Scientists on Twitter: Preaching to the choir or singing from the rooftops?" by Isabelle M. Côté and Emily S. Darling makes a compelling case that the modern generation of scientific scholars have shifted to social platforms to exchange with and around scientific material. While ill-equipped to allow the full palette of scientific discourse, Twitter has a massive network effect that is both helping inreach and outreach of their field of research.
ResearchGate, Mendeley, Academia

Social networks made for researchers only.

Academic social networks support activities across the research cycle, from getting job suggestions, sharing and reading full-text papers to following use of your research output within the system. We looked at detailed functionalities offered by ResearchGate, Mendeley and Academia.edu.
Academic Torrents

A platform to organise datasets replication.

Claiming to host and ensure replication of more than 33.85TB of datasets as of 2018, Academic Torrents is run by a 501(c)3 non-profit corporation called the Institute for Reproducible Research. Since 2013, it has gained support from multiple high-profile universities and research institutions.

It consists of a website consuming a simple API to list torrents of datasets. A simple commenting and reporting system exists, but is clearly not the focus of the platform, which is centered on transport via BitTorrent and showing the current peers replicating a given dataset.

Where data is stored. Datasets are never stored on the platform. Using any BitTorrent client, you can download from other peers. A typical upload scenario is not uploading any data on the platform, but just a `.torrent` file containing the processed metainformation to join the swarm and retrieve the dataset chunks.

How data is accessed. Relying on BitTorrent and storing no data on the platform means the browser cannot get the data itself, nor via the platform. This limits interaction with and around excerpts of the dataset, or insights about the dataset from the platform.

How data is controled. Being centralized around a single website and a single BitTorrent tracker (the entity allowing easier peer discovery), content can be put down easily. While not a bad thing per se, it is essentially giving a one-size-fits-all editorial and moderation policy for research.

How data is discovered. All datasets to be searched are known by one instance of the platform that lists them. Indexing and searching through them is trivially achieved. A simple listing focusing on the quality of replication (as is the main focus of the platform) allows to decide from dataset to dataset. No editorial works seems to be done at the moment, and dataset titles are copying the torrents' main folder name at best. A full text search is then done on that title and the category of data specified.

Pros

- Very good interoperability
- Light maintenance costs

Cons

- No interaction with data
- No exchange around the data
- Not open source
The Dat Project & Science Fair

Designed for civic and research data storage.

The Dat Project revolves around data sharing and explicitly aims at doing better than Academic Torrents. dat:// is a peer-to-peer protocol for building decentralized tools. It is a community-driven project (as opposed to IPFS, a similar VC-backed project) centered around a protocol that is transport-agnostic and whose key concepts are allow robust archival and access of data.

Having a history of how files have changed is essential for effective collaboration and reproducibility. Git has been promoted as a solution for history, but it becomes slow with large files and a high learning curve. BitTorrent is good with large files but inefficient at providing random access to data in larger datasets, which is crucial for sparse access.

An excellent documentation of the Dat protocol main operations is available here.

Pros (Science Fair)

- Light maintenance costs
- Data can be viewed

Cons (Science Fair)

- No exchange around the data
- Only available via a desktop app
Other Open Source Platforms

Open source platforms using a more traditional model of service to the user: local storage and/or listing local resources only and still little to no interaction.

**Meta-Share (FP7 & ICT PSP)**

[github.com/metashare/META-SHARE](https://github.com/metashare/META-SHARE)

Hierarchical federation of servers, that bear different roles (pertaining to storage or indexation).

Infrastructure of networked repositories that is transparently viewed as one global collection of resources.

Uses a central node as a looking glass/portal to search datasets.

 Doesn’t directly support [OAI-PMH](https://www.openarchives.org/oai/), but explicitly puts responsibility on the instance owner to allow it.

Made using Python.

**Dataverse (Harvard)**

[dataverse.org/software-features](https://dataverse.org/software-features)

No federation of servers (so discovery is local-only), but interoperable formats and standards.

Integration with a lot of providers.

No support for OAI-PMH.

Made using Java.

**Zenodo (CERN)**

[github.com/zenodo/zenodo](https://github.com/zenodo/zenodo)

No federation of servers (so discovery is local-only), but interoperable formats and standards.

Uses heavy dependencies and frameworks (Invenio, Elasticsearch) raising the bar for deployment.

Tightly coupled with its storage infrastructure, and all data is stored at CERN (see their infrastructure).

Supports [OAI-PMH](https://www.openarchives.org/oai/).

Made using Python.
Software Requirements Specification
SOFTWARE REQUIREMENTS SPECIFICATION

for

the OLKi platform

Version 0.1

Prepared by Pierre-Antoine Rault

LORIA

March 26, 2019
"For Artists and Hackers, Academics, and Admins. For Humans who Grok stuff. Those self taught, and persistent. For those who build for Us:
Tools to own and run. For People in Community: Who will not be a Commodity, For Our Internet, And all who sail on her. Useful may it be.
Save time may it much."

Peter Gossner, October 2018, Fediverse Glossary
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Purpose</td>
<td>17</td>
</tr>
<tr>
<td>1.2 Document Conventions</td>
<td>17</td>
</tr>
<tr>
<td>1.3 Intended Audience and Reading Suggestions</td>
<td>17</td>
</tr>
<tr>
<td>1.4 Project Scope</td>
<td>17</td>
</tr>
<tr>
<td>1.5 Project Deliverables</td>
<td>18</td>
</tr>
<tr>
<td>1.6 References</td>
<td>19</td>
</tr>
<tr>
<td><strong>2 Overall Description</strong></td>
<td>21</td>
</tr>
<tr>
<td>2.1 Product Perspective</td>
<td>21</td>
</tr>
<tr>
<td>2.2 Product Functions</td>
<td>21</td>
</tr>
<tr>
<td>2.3 User Classes and Characteristics</td>
<td>23</td>
</tr>
<tr>
<td>2.4 Operating Environment</td>
<td>24</td>
</tr>
<tr>
<td>2.5 Design and Implementation Constraints</td>
<td>24</td>
</tr>
<tr>
<td>2.6 User Documentation</td>
<td>25</td>
</tr>
<tr>
<td>2.7 Assumptions and Dependencies</td>
<td>25</td>
</tr>
<tr>
<td><strong>3 External Interface Requirements</strong></td>
<td>26</td>
</tr>
<tr>
<td>3.1 User Interfaces</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Hardware Interfaces</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Software Interfaces</td>
<td>26</td>
</tr>
<tr>
<td>3.4 Communications Interfaces</td>
<td>27</td>
</tr>
<tr>
<td>3.5 Administration Interfaces</td>
<td>27</td>
</tr>
<tr>
<td>3.6 Moderation Interfaces</td>
<td>27</td>
</tr>
<tr>
<td><strong>4 System Features</strong></td>
<td>28</td>
</tr>
<tr>
<td>4.1 Scientific Watch</td>
<td>28</td>
</tr>
<tr>
<td>4.1.1 Use Case</td>
<td>28</td>
</tr>
<tr>
<td>4.1.2 Functional Requirements</td>
<td>28</td>
</tr>
<tr>
<td>4.1.3 Description and Priority</td>
<td>28</td>
</tr>
<tr>
<td>4.2 Ease of data versioning and diffusion control</td>
<td>29</td>
</tr>
<tr>
<td>4.2.1 Use Case</td>
<td>29</td>
</tr>
<tr>
<td>4.2.2 Functional Requirements</td>
<td>29</td>
</tr>
<tr>
<td>4.2.3 Description and Priority</td>
<td>29</td>
</tr>
<tr>
<td>4.3 Communication around the publication</td>
<td>29</td>
</tr>
<tr>
<td>4.3.1 Use Case</td>
<td>29</td>
</tr>
<tr>
<td>4.3.2 Description</td>
<td>31</td>
</tr>
</tbody>
</table>
## Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-A. Rault</td>
<td>March 19, 2019</td>
<td>initial version</td>
<td>0.1</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Purpose

The present document specifies parts of a computer system, hereinafter called ”OLKi platform” or ”platform”, as well as the distribution services that might be added to it, and the ecosystem of federated platforms that it wishes to integrate with.

The platform is a web service intended to run on a standard tiered server: behind a web server, and associated with a database and cache system. The parts of the platform that are discussed in this document cover both its web client and its server runtime.

The first release of the platform is projected to have the version number 1.0. This document describes its functionality.

1.2 Document Conventions

Defined terms are highlighted with bolding. Requirements will come with priority to indicate in which order they will be implemented. Versions might be released with only some of the total list of requirements implemented.

1.3 Intended Audience and Reading Suggestions

The present document is mostly describing the technical requirements, and as such is intended for developers and system administrators (system administrators can directly go to the Deployment Requirements section 5.7 page 36 for notes specific to their mission).

While most concepts are explained on first use, a prior knowledge of what a tiered application is1, what the fediverse is2, and what platforms we already surveyed in our initial reports3 is advised.

1.4 Project Scope

In a time of increasing distrust and defiance of citizens towards artificial intelligence, the scientific community is not left untainted by the recent scandals and must provide an answer to a justified concern of the society. OLKi wants to take part in that nascent effort by ensuring datasets and particularly linguistic datasets that are the essential resource

1https://12factor.net
2https://fediverse.gitlab.io/en/fediverse
3https://framagit.org/synalp/olki/documents/initial-report/releases
of machine learning techniques can be easily hosted and interacted with for the whole scientific community, with the twofold objective of transparency and understandability of research data.

The OLKi project will connect researchers, scientists and citizens through the Fediverse. To do so it will extend the ActivityStreams language to support scientific research datasets and their related activities for the web, and create the OLKi platform to feature it.

The Fediverse is originally a social-network-oriented text communication platform that has emerged as a citizen initiative in 2017. It has quickly grown, both in number of users (more than 2.5 million users in November 2018) and supported functionalities, including for instance blog-like platforms (Plume), image sharing (Pixelfed), video sharing (PeerTube), audio sharing (funkwhale), etc. all while maintaining the basics for compatibility and interoperability (Pleroma being one of the main examples). One of the main strengths of the Fediverse is that all of these features are interconnected together through open standards, enabling users of one feature to access other features as well.

A major side-effect is that web applications don’t have to compete with each other but can understand each other (with a degree of compatibility). It might seem insignificant since it just amounts to the classic gains of using open standards. However in the case of social interactions, it allows even small and novel software to free themselves from Metcalfe’s law by gaining support from existing communities.

The OLKi project will further enhance the Fediverse with a new functionality oriented towards scientific communication and research results dissemination. This will fill the current need of a dedicated global and open communication platform between researchers, but also between scientists and citizens. A need that is currently (partially) covered by sites such as Twitter or ResearchGate, although we argue they are unfit for the role. In that sense it will be more than a platform to host datasets and provide features that current dataset hosting platforms lack.

### 1.5 Project Deliverables

The OLKi platform will release:

- a landing page and technical blog, already available at https://olki.loria.fr/platform/
- a project account for announces on Mastodon, already available at https://mastodon.etalab.gouv.fr/@scifed
- a complete source of its software(s)
- a pre-compile frontend if need be for ease of deployment
- a `.deb` compatible with Debian stable
- a YunoHost definition for YunoHost support (using the previously defined `.deb`
- a Docker image based on Debian stable (using the previously defined `.deb`)
1.6 References

The present document refers to multiple documents that the reader might want to be familiar with:

- https://webfinger.net/andwww.rfc-editor.org/rfc/rfc7033.txt
- https://www.w3.org/TR/activitypub/
- https://www.w3.org/TR/activitystreams-vocabulary/
- https://www.w3.org/TR/activitystreams-core/
- http://www.openarchives.org/OAI/openarchivesprotocol.html
- http://microformats.org/wiki/dry
- https://indieweb.org/principles
- http://www.rssboard.org/rss-specification
- https://www.w3.org/annotation/ and its refinement https://www.w3.org/TR/wpub-ann/
  https://w3c.github.io/web-annotation/selector-note/index-respec.html#FragmentSelector_def is also extremely important.
- https://www.w3.org/TR/ldn/
- https://12factor.net/
- https://www.divio.com/blog/documentation/
- https://json-ld.org/
- https://w3c-dvcg.github.io/ld-signatures/
- https://graphql.org/
- https://opensocial.github.io/spec/2.0.1/Social-API-Server.xml#ActivityStreams-Service
- http://robustlinks.mementoweb.org/
- https://oauth.net/
- http://dev.opml.org/
The following are related to general design and/or UX:

- https://2017.ind.ie/ethical-design/
- https://www.w3.org/WAI/standards-guidelines/wcag/

The following are related to diffusion protocols that might be used at a later stage:

- https://datprotocol.github.io/how-dat-works/

The following are related platforms that feature some of the previously cited technologies:

- https://web.hypothes.is/
- https://zenodo.org/
- http://www.meta-net.eu/meta-share/architecture
- https://academictorrents.com/
2 Overall Description

2.1 Product Perspective

The need for a scientific platform like OLKi arises with the ever increasing use of social platforms by both researchers and citizens in general, in spite of their lack of scientific tools. The OLKi platform aims to close that gap, and allow everyone on open social networks to benefit from tools that matter when discussing around research material but more generally any corpora complex enough to require targeted annotation and referencing.

A common example is for paper reviews to be made in the form of Twitter threads with image captures text passages, or Reddit threads discussing the publication of a corpus. Mixed populations answer and bring value to the discussion, but more than that they take part in the most accessible form of vulgarisation. This allows to bring people closer to the scientific material itself, a necessary thing especially at times when the outcome of a research can quickly get overblown if one is only reading the abstract - or worse its retranscription on TV.

We argue the need is even more urging when it comes to machine learning: announces prophetising strong AI and their consequences are made daily and their form doesn’t allow the proper hindsight to judge of their pertinence or react with anything but fear, lest enough of their content is available for review by everyone. OLKi takes that mission to build bridges between traditional scientific platforms and their content, and traditional social networks, so that science is not anymore a distant reverie only accessible to scientists.

2.2 Product Functions

OLKi takes its root in an environment where data and algorithm replicability is key to understanding, and the platform will be no different. All of its functions will base off open source and open standards. The resulting platform itself will be libre, free software. The social and scientific platforms OLKi interacts with will also have to be open or at least give an open API.

Basic functions of OLKi amount to indexing content uploaded by its users in the form of research corpora (any document, text or dataset).

- This content is indexed and searchable. Other content of the same type (corpora) originating from federating servers will also be indexed and searchable (see
how OLKi federates and how OLKi displays content). Other content harvested (scraped) from ArXiV and HAL will also be indexed and searcheable.

- This content is uploaded by trusted users freely, or else it goes through a ”pending review” from a moderator or administrator of the server before going public.
- This content can be interacted with via a rich comment (regular Note comment with additional annotation syntaxes) freely or only by chosen accounts.
- Accounts, IPs and full domains can be banned of interacting by an instance administrator or at the account level. Since having content to reference can quickly turn into a heavy moderation task,

**Content discovery**

*Foreword:* content discovery is important in any platform. It drives the relationship with most platforms, and is often presented first to the user. OLKi is no different, and content discovery plays a big role in some of the use cases/features later detailed in chapter 4. We thus have to make assumptions as to how users will apprehend the OLKi platform when coming to the main page, detailed below.

Instances are assumed to be hosting content of a group of researchers. Their work cover a limited number of topics that will be chosen by the instance administrator as being "representative" of the "focus" of an instance’s users. This will impact the listing of content displayed by default in the general listing of all content. By "general content”, we mean a listing that mixes local, federated and harvested content. Let us call this listing "Global view”. As opposed to a "Personal view” that lists only persons/groups you follow as a user and that is only available when logged-in, the "Global view” lists corpora that can be seen from the point of view of the server, but filtered down to the selected set of topics of focus of the instance. It gives an insight of what is accessible for review to users and visitors alike.

Local content is favored in the display with reserved spaces when possible, and filtering options allow to show local content only. A dedicated view listing local research teams wishing to be listed is available as another view. This ensures an instance of OLKi can be used by its users to showcase their work.
**User and Teams profiles**

Users have a profile with a name, picture, description, and list of works (corpora). They can constitute Teams which also have a name, picture and description. Teams are groups of local users that by default do not show any information but the corpora and members its members wish to showcase.

**Corpus page and interaction mechanics**

Opening a corpus switches to a view (≈ page) split in two panes: on the left the title, description, metadata and list of files, and on the right a pane listing/suggesting interactions with the corpus (for instance comments, statistics about the data, etc.).

![Corpus page and interaction mechanics diagram]

**Navigation**

- **Browse** | **Profile** | **Upload**
- **view navigation**
- **login/notifications**
- **current view actions**

**2.3 User Classes and Characteristics**

The OLKi platform pertains to the needs of the following initial user populations (which incidentally are the classes of users corresponding to each scientific group involved in the OLKi project - from the point of view of their usage later identified in the use cases in chapter 4, page 28):

1. general computer science researchers
2. machine learning researchers
3. natural language processing researchers
4. researchers in epistemology
5. researchers in linguistic

6. researchers in mathematics

We note computer science researchers often have less use for a platform to host corpora when most of their data amounts to code that is already hosted on dedicated software forges (web-based collaborative software platform for both developing and sharing computer applications). They will be more interested in scientific watch features.

Other classes of users might be later taken into account, but for now we will only count citizens in general. We consider they won’t bother creating an account on the platform for a single review (especially since instances will be institutional and closed to registrations at first) but still might want to reference or annotate documents. Users not creating an account can still interact with the public features of the platform; they are called visitors.

2.4 Operating Environment

The platform is meant to be run on small to medium servers, as can be typically found in most virtual private server providers’ offers. It requires to run more than a simple web server but also a runtime of the selected language. Typical shared hosting with no possibility to run anything beside PHP behind a web server will prevent running the platform, and deployment cannot be expected to be done with just copying files via FTP.

The runtime will expect existence of a PostgreSQL database running locally and accessible via socket or local TCP connection, as well a Redis service under the same circumstances.

The runtime will bind on the loopback interface only, and on a configurable port. It is expected the system administrator configures

2.5 Design and Implementation Constraints

The implementation of the ActivityPub federation must respect the Mastodon dialect of ActivityStreams, and be able to send/receive compatible Note objects. Custom objects part of the platform’s own ActivityStreams dialect will eventually need work to integrate with other platforms like Mastodon, but do not constitute a requirement.

To be deployed by institutions, the platform will have to support LDAP/CAS registration.

The LORIA will maintain an instance for their own use and an instance for each organization member of the OLKi project. They will constitute a prime example of the functionalities featured by the platform, and be listed on the public list of demonstration instances, even though they are production instances. A proper demonstration instance will be put up as well along them, and will have a staging role in the continuous deployment operations of the software.
2.6 User Documentation

Along with the platform, a documentation for a) end users, b) developers and c) system administrators will be deployed at https://olki-docs.loria.fr. While considered both developer and third-party developer documentation, API documentation will also receive extra care and be available online.

2.7 Assumptions and Dependencies

The OLKi platforms aims to be able to run on its own without any hard dependency to external services. However, some components that tie to external and centralised platforms like arXiv, HAL or ORTOLANG will affect its usability in case they go down or change their APIs (even though they are deemed stable since we will be harvesting metadata of their newest publications using OAI-PMH). In case they go down, the value of some features (i.e.: scientific watch) will be diminished but not rendered unavailable.
3 External Interface Requirements

3.1 User Interfaces

The platform will be exposed as a web service consumed by a web client in the form of a web page. A login will be required for authenticated services, but part of the web interface will be usable with no account on the platform, so as to expose reference and annotation visualisation beyond simple quotes or images in conventional comments on federated third party platforms.

3.2 Hardware Interfaces

The platform runtime acts as a web server that binds to the loopback address on a port set by the system administrator. It answers to requests on that port only.

The platform will store files locally.

3.3 Software Interfaces

To run, the OLKi platform relies on the following external dependencies:\(^1\):

- PostgreSQL >= 9.6 to run the backend (model storage)
- Redis >= 2.8.18 to run the backend (request caching)
- Python >= 3.7.x to run the backend (runtime for API service)
- NodeJS >= 8.x to run the frontend (runtime for API consumption)

Support will be ensured for Debian stable and, if different, YunoHost stable.

Behaving a web server, the runtime will initiate a long-running connection to the Redis and PostgreSQL backends immediately upon start. It then answers to requests made to its local port. It is meant to be forwarded requests by a proper web server that handles TLS termination in its stead.

The platform is a long running process that is supposed to be run as a system service, of which we plan to provide service files for deployment.

The platform will store files locally, so any FUSE-mounted directory or remote device can be used as a storage directory for files. Different storage directories are considered by type of content.

\(^1\)by "external dependencies", we refer to dependencies that have to be installed by the system administrator manually when installing the application
3.4 Communications Interfaces

Deployment-wise: the platform will require an e-mail account to send notification mails from. The service being an HTTP API and its web client, it will require no less than a web browser (no assumption is made as to the compatibility ensured for web browsers, but as no outstandingly modern format or feature is planned client-wise, the compatibility should be broad by default). The web client will communicate with the web server via the ActivityPub client standard for social activities, and regular GraphQL operations otherwise.

User-wise: the platform will require an e-mail account to send notification mails to, and a password to login with.

3.5 Administration Interfaces

Administration tasks cover functional tasks (choosing what optional functions to enable), and hybrid (functional and moderation-related) tasks like deciding who to federate with. As many tasks as possible are done from the web interface, although some might require intervention from the system administrator.

We acknowledge that while most small instances are managed by an admin user that is also the system administrator, institutional instances will see a clear distinction most of the time. We then need to allow the admin user to do most of the task, to reduce the need for the system administrator to intervene.

3.6 Moderation Interfaces

Admin user can do all moderation tasks, but moderator users dedicated to the task are desirable to reduce the load. They can mute or ban users or whole instances. Normal users can mute or ban users or whole instances from interacting with them. Teams can mute or ban users or whole instances from interacting with them. In that case the users in charge of a team decides for the mute and ban actions.

Content to be moderated can be submitted to an admin/moderator by users via reports if a user deems the whole instance is at risk of further interacting with the user. A report is done on an activity. Upon report the activity is pinned for moderation and placed in a review queue. Content uploaded by new users will also automatically be put in a review queue. The goal is to ensure new users are trustworthy.
4 System Features

Completing the overview of the platform functions in section 2.2, we detail the main use cases the platform was designed for.

4.1 Scientific Watch

Keeping in touch with the most recent publications, a specific scientific community, and more generally getting notified about them, them being a social group, tag (think of categories or hashtags), or generic selector.

4.1.1 Use Case

Some scientific fields see a lot of publications daily, and keeping track of each and every single one is a tedious task. Some centralized tools aim to reduce the churn, but they are poorly integrated in our social lives. ArXiV for instance doesn’t provide a conform RSS feed, a standard which is not even supported by the main centralizing social platforms.

For instance, the field of computer science has dozens of new papers daily, and filtering them with consistent tools across platforms like ArXiV or HAL could help. More than that, filtering could be done in a more sophisticated way than what these platforms offer: with a model running on metadata. That kind of tag could help researchers tune their feed to their liking.

4.1.2 Functional Requirements

We note that the targeted services claim support for OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting), which allows us to harvest metadata from them via a single, standard client implementation. We also note that supporting that standard allows us to potentially harvest from 4022+ OAI conforming repositories (as of early 2019)\(^1\).

4.1.3 Description and Priority

The feature is deemed important to help scientific users see an immediate added value to their scientific workflow, and place the platform as a place of choice to keep visiting (even though the implementation provides interoperability with off-loading feeds to other tools of choice). It will not affect all fields of research the same, but some will definitely need such a feature to keep track of papers. It also correlates with the import of third-party

\(^1\)http://www.openarchives.org/Register/BrowseSites
metadata (also called "harvesting") and its indexation which will improve the quality of the platform’s search function, so developing it will help improve the overall quality and pertinence of the platform.

Following other scientists and getting a private page of subscriptions sorted by query, with alerts, priority: low.
Public feed of publications, priority: high.

4.2 Ease of data versioning and diffusion control

While authoring documents is generally out of the platform’s scope, updating data is something the platform has to support. Versioning data is also an option we should provide, but above all we have to enable scientists to filter how their publication is diffused, and who can see it.

4.2.1 Use Case

As it stands, publishing corpora to dedicated platforms can be a demanding action that doesn’t align with the evolution of one’s work: for instance, ORTOLANG forces authors to publish their datasets with a specific license and makes authors pledge to deposit their publications long term in the open, a constraint that only makes sense for finished works and is unpractical for nascent projects.

Scientists need to be able to remove their work from public listing, make it private or accessible to a restricted set of collaborators if need be. They must be able to publish their data with the license(s) they want.

4.2.2 Functional Requirements

4.2.3 Description and Priority

A detailed implementation would take time, but states (public, unlisted, private, password-protected) could be adopted early-on by the platform, so the ease of implementation justifies a medium priority.

4.3 Communication around the publication

Discussion groups and more generally comments should be available in the interface and should be easily accessible to contribute to the discussion around the publication.

4.3.1 Use Case

At first we just aim to support basic textual comments using the Mastodon dialect. The content of the comment will be able to use \LaTeX mathematical syntax, which will be srendered when viewed in the OLKi platform web client. A specific reference syntax
will be available to make references to specific files inside a Corpus (see following Note where we assume we have a list of files being the source of a paper, and the paper itself).

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "@type": "Create",
  "actor": {
    "@type": "Person",
    "@id": "acct:rusnak@mastodon.example.org",
    "name": "Gene A. Rusnak"
  },
  "object": {
    "@type": "Note",
    "content": "In the recently published paper on 'Absolute Dating of Deep-Sea Cores by the Pa^231/Th^230 Method,' Rosholt, Emiliani, Geiss, Koczy, and Wangersky (1961) committed an error in their graphic representation of sediment accumulation rates base on gm/cm^2/1000 years. See file=6,7 or file=1&page=10&viewrect=50,50,640,480 and file=1&page=10&viewrect=150,150,640,480.",
    "inReplyTo": "http://olki.example.org/corpus/1",
  },
  "published": "1963-11-01T12:00:00Z"
}
```

Of course the redaction of full fragments referencing parts of a .pdf or image is difficult, and that is why the editor withing the web client will help their redaction with a visual selector. Even if they don’t use that extended syntax, The main goal is for Mastodon users to be able to answer easily. They can use the same syntax as above and it will be rendered as a link when displayed by the OLKi client, or they can directly put the fully qualified link:
Long term, supporting suggestions of modifications would be possible using Offer, Accept and Reject AS activity types².

4.3.2 Description

Communicating with third-party platforms is done via interoperability in its strictest sense: servers running the OLKi platform code (also named instances) should be able to exchange information seamlessly so as to make a comment from a user on instance A on a dataset published on instance B viewable by users from instance A and B (modulo diffusion restrictions set by the user with publishing rights over the dataset). But more than compatibility within the network of servers running the same software, being able to exchange information (even slightly degraded) with third-party platforms like Mastodon or Pleroma allows to port activities to where users are, and allows them to answer back. This is federation intra-codebase and inter-codebases, and this is allowed by social exchanges standards like ActivityPub³ (abbreviated AP later in this document), the latest and most adopted so far.

4.3.3 History of Web Federation Protocols

But before we dive into the functional requirements involved by ActivityPub and ActivityStreams⁴, let us recapitulate the past few years of federated social protocols in usage for the web (of which you can find more on lain’s blog):

Before AP, there were a few federation protocols in use already. The two largest ones are probably OStatus⁵, the system that GNU Social (a free, libre software social

⁴https://www.w3.org/TR/activitystreams-vocabulary/
platform created *circa* 2010) invented, and the Diaspora protocol\(^6\). These are actually rather similar and even partially interoperable (Friendica, another free and libre platform implemented both seamlessly with little adaptation). Both essentially revolve around exchanging activity data (a post, a like, a comment) between servers. Said simply, OStatus works well for Twitter-style open communication, Diaspora works best for group-based communications (friends, acquaintances, colleagues...).

These protocols worked as expected, overall. Still, some parts of them made people look for new solutions. Especially OStatus was (and is) severely underdocumented. There is no document that you can read to implement a compatible server. Even if you read the specifications of all the standards it is made up from, you will still not know how to write a compatible server. What you have to do is to “just do what GNU Social expects”, essentially copying whatever it does while checking the existing documentation. This is an uncomfortable process that is now similarly happening in the ActivityPub realm with Mastodon leading the implementation of the protocol with its dialect.

One might argue that there was no need for ActivityPub since OStatus and Diaspora were already covering most needs. But lack of documentation (as previously mentioned) or lack or flexibility in the types of objects that can be defined, or usage of a decades-old and difficult to parse XML got the better of them and put AP right on top. While theoretically there is no technical winner, the tide has since turned in favor of ActivityPub in terms of usages and prospective of implementations.

### 4.3.4 Functional Requirements

Federation-wise, the use case of OLKi is close to that of another platform: PeerTube. PeerTube federates Video objects for the eponym content that constitutes the core of its purpose, and *Note* objects (the name of text messages in ActivityStreams linguo) for comments that are compatible with the standard established by Mastodon dialect. In the case of OLKi, we would define a Corpus object (a collection of documents) and its associated discussions (comment threads in terms of *Note* objects). Much like with PeerTube, OLKi’s comments would be seen to their fullest in Mastodon but Corpus objects wouldn’t be understood. Instead, they would at best be transtyped to *degraded Note* objects, again as is currently the case for PeerTube’s Video objects when they reach a Mastodon instance.

### 4.3.5 Priority

Priority: high.

### 4.4 Annotation of corpora and data

While authoring documents is generally out of the platform’s scope, annotating data is something the platform has to support both for a corpus’s authors and its commentators.

---

\(^6\)https://diaspora.github.io/diaspora_federation/
Note: special attention should be given to making annotation referenceable from non-toolled platforms via links that redirect to a proper web view. For instance, a Note object sent to the fediverse should be openable for a detailed view of the annotation. A simple URI syntax should be given for the most simple annotation cases too.

4.4.1 Use Case

Decentralising the creation, publication, and annotation of hypertext documents provides authors with a technological guarantee for independence of any publication authority. While the platform doesn’t provide authoring (creation) of documents since it focuses on hosting various types of documents, individual authors still lack the ability to conveniently engage in social interactions with documents of others in a truly decentralised fashion. With open protocols and standards for annotation, annotations can be seen as social activities and shared as such by social platforms. Such an environment provides the right technological conditions for independent publication of scientific articles, news, and other works that benefit from diverse voices and open interactions.

What can the user scenario look like? See a .pdf in a corpus of documents, select a few lines of text and click annotate. Your selection is then referenced as a link put in your comment zone. You then just have to write your review in the comment section, and the comment will be sent as a textual social activity that can be seen in Mastodon and most textual federated platforms.

4.4.2 Functional Requirements

4.4.3 Description and Priority

4.5 Data diffusion

Data replication and persistence is not a goal we can achieve with full control over the data, but it is an option we should give long term so that we can provide a middle ground between having all data on the same server and offloading everything to a data warehouse with availability guarantees (CDN or long-term institutional storage). Interoperability with existing P2P protocols is envisioned, especially with the example given by Academic Torrents⁷.

4.5.1 Use Case

Big datasets might represent a heavy burden on some servers, especially non-institutional ones. They might be running the platform on a small server with limited disk space and/or bandwidth. In that case, storing the metadata and linking to a content stored elsewhere could be a proper answer.

Organising the diffusion by thinking of resources as links stored and eventual copies served by the server will p

⁷http://academictorrents.com
4.5.2 Functional Requirements
4.5.3 Description and Priority
5 Other Nonfunctional Requirements

5.1 Legal Requirements

In order to be compliant with the GDPR, the platform must respect a few points:

- keep users informed about how their personal data is used.
- make users confirm they are above 13 (or 16?) years old upon registration.
- make users confirm they agree to the terms of use.
- keep users informed about terms of use changes (via notifications).
- provide tools to hide specific content terms from listings.
- allow users to remove all the data associated to them (corpus, comments, etc.).
- allow users to export that data.

5.2 Performance Requirements

The runtime is expected to run below 200MB of RAM in average, and below 500MB at peak consumption. One virtual core should be enough to deal with computations, as no heavy computation in background processes is expected (no transcoding, post-upload file transformation or machine learning).

5.3 Safety Requirements

No specific safety requirements apply.

5.4 Security Requirements

The platform will not ensure TLS termination, as this is the web server’s job. We will however provide guideline to configure popular web servers correctly, and ensure we properly outline what is related to security in their configuration, as well as where to update these guidelines in case of new security recommendations.

The platform must properly store the users’ credentials: hashed with a password hashing function that supports salting and expensive key scheduling, such as Argon2i. The password hashing algorithm must be resistant to side-channel attacks as the server can be a shared machine.
5.5 Software Quality Attributes

The platform should favor interoperability over feature-completeness or fitness for the purpose. This is especially true when implementing annotations and federation, where different standards and implementations coexist. Maintainability both for the developer community (including scientists) and ease of deployment are two faces of the same problem, which should be prioritized over architecturing for performance when the gain is low, moreso than with other platforms.

Since the software should also be maintainable to some extent by scientists, and judging by the preeminence of Python-literate programmers in most computer science laboratories, Python has been selected as the implementation language of choice for the platform. Likewise, we chose the Django web framework as it provides an opinionated and very well documented set of API and countless libraries that will provide a solid basis for the project to grow.

5.6 Business Rules

No business operation-specific rule has to be enforced.

5.7 Deployment Requirements

The application must be deployable on the latest stable version of Debian, and if possible with a YunoHost package. The goal is to make the deployment as simple as possible, even if it means pre-configuring some parameters, so that as many users and institutions can test the software, if not adopt it.

It also means updating the applications should be as seamless as possible. This imposes the use of schema migrations on the application side to describe changes in the database schema from version to version.

A checklist of deployment will be given for system administrators to check their installation matches the expected quality that the platform purports to offer (including checks for the aforementioned other nonfunctional requirements).
6 Other Requirements

Internationalization of the platform web client, platform backend messages and landing page has to be supported by the end of the project, and as early as possible in the project to favor external input.

6.1 Appendix A: Glossary

A glossary dedicated to terms related to federated social networks and their protocols can be found at the Fediverse Glossary by Peter Gossner.

- **ActivityStreams (AS)**: a standardization effort to define common types of objects and actions (verbs) taken on various social media sites. Quick reference to AS types and verbs

- **harvesting**: gathering metadata from diverse sources in an archive so that services can be built using metadata from many archives. Initially referring to a protocol permitting e-print servers to expose metadata for the papers it held in a structured fashion so other repositories could identify and copy papers of interest with each other. This protocol is now named OAI-PMH1.

- **instance**: in software engineering and web development, an instanciation of a software, generally designating the server running it and its domain. Example: ”An OLKi instance” refers to a server, domain and the people running it. It is an important concept in federated networks where software is

- **user**: a person using the software. Unless specified, implies a registered user with an account on a platform (otherwise it is called a visitor).

---

1 [http://www.openarchives.org/OAI/openarchivesprotocol.html](http://www.openarchives.org/OAI/openarchivesprotocol.html)
## 6.2 Appendix B: Analysis Models

<table>
<thead>
<tr>
<th></th>
<th>Helpful (to achieve the objective)</th>
<th>Harmful (to achieve the objective)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal origin</strong> (product/software attributes)</td>
<td>own platform, basic annotation needs</td>
<td>scientific community not voicing needs, adoption</td>
</tr>
</tbody>
</table>

| **External origin** (environmental attributes) | existing communities strong need for review tools | federation dialects compatibility |

## 6.3 Appendix C: To Be Determined List
7 Project Backlog

- Users (2 months)
  - User registration via CAS/LDAP/local
  - User invites
  - User profile
  - User export (GDPR)
  - User delete
  - User upload of documents/Corpus creation
- Corpus (3 months)
  - Corpus page
  - Corpus listing
  - Corpus filtering (= basic search)
  - Corpus commenting
  - Corpus federated commenting
  - Corpus federated between instances
- Moderation (1 month)
  - User roles
  - User reports
  - User quarantine
  - User ban/mute
- Federation (2 months)
  - User federated profile
  - User messages and direct messages
  - User federated messages
  - User federated reports
- Feature: Watch (1,5 months)
  - Corpus search
  - Corpus alerts/daily/weekly
- User notifications (mail/AP/interface)
- Corpus harvesting (OAI-PMH client)
- Feature: Diffusion (2 months)
  - Serve feeds (RSS, OAI-PMH server)
  - BitTorrent storage/magnet (Dat ?)
  - Corpus versioning
  - Password-protected views