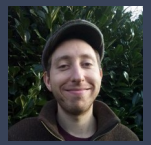
Outils et mécanismes d'indexation pour la recherche et la découverte de données dans un écosystème Solid



Maxime Lecoq-Gaillard

Agenda

01. Introduction and context
02. Indexes in the Solid ecosystem
03. Comparing indexing strategies
04. Implementations
05. Future and perspectives

Deliverable

01

Introduction and context

10/2022 → **10/2024**

@Wimmics team at INRIA

Maxime Lecoq-Gaillard
Research Engineer

Fabien Gandon
Research Director

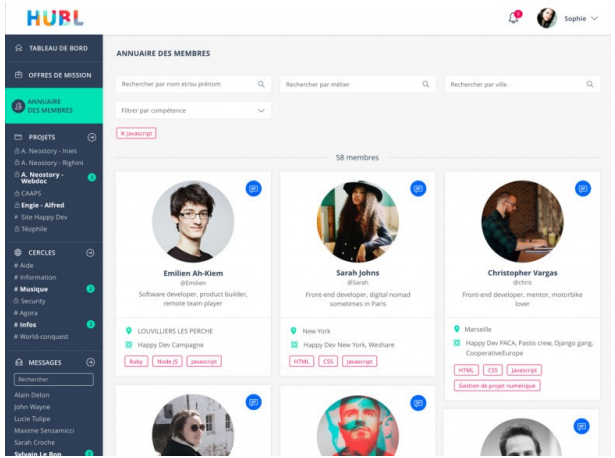
Pierre-Antoine Champin
Assistant Professor, Inria Delegation, W3C Fellow

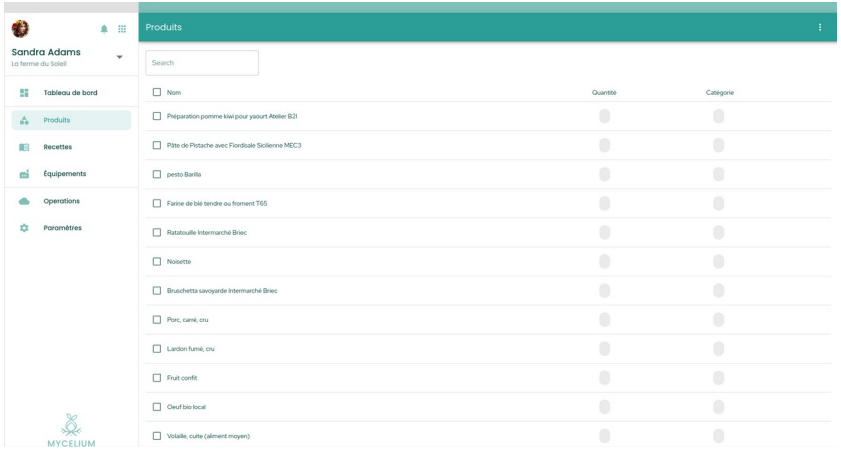
@Startin'Blox (startinblox.com)

- Framework to build semantic applications.
- Solid server in Python

Benoît Alessandroni
CTO

Sylvain Le Bon
CEO





HUBL
freelance communities

Find freelances given their skills, location or name...

MYCELIUM
agriculture activities

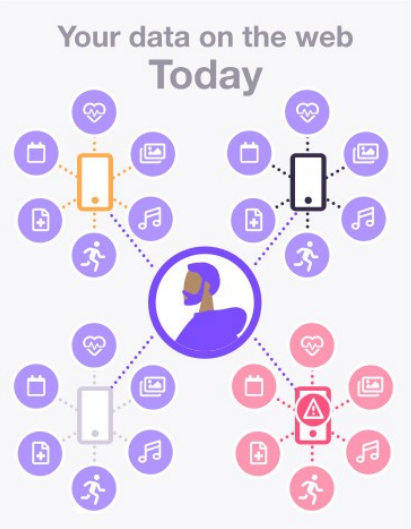
Find products, producers, search for orders...

Solid

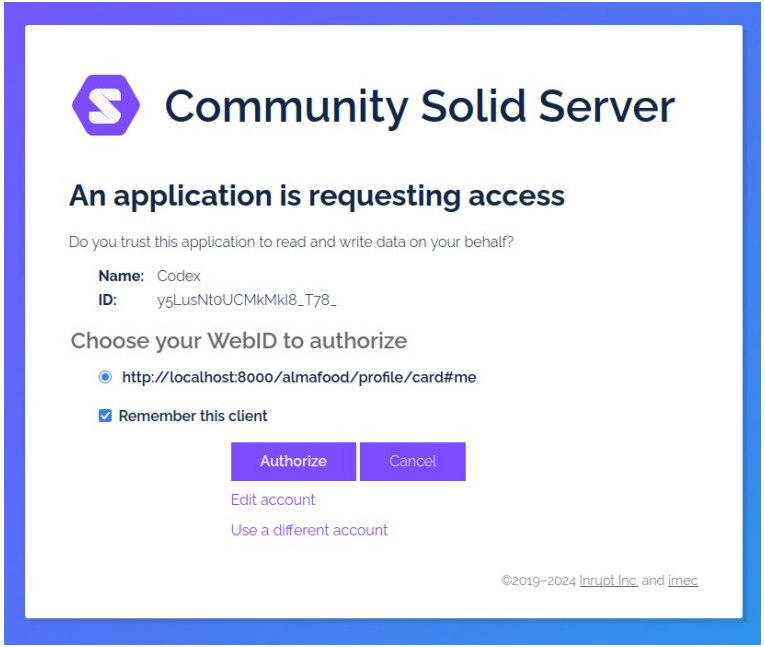
"When I invented the World Wide Web, I envisioned technology that would empower people and enable collaboration. Somewhere along the way, we lost that human-first approach; today's web often prioritises profits over people. Solid returns the web to its roots by giving everyone direct control over their own data."

– Sir Tim Berners-Lee

Solid


Your data on the web
Today


Your data on the web
with Solid


Community Solid Server
An application is requesting access
Do you trust this application to read and write data on your behalf?
Name: Codex
ID: y5LusNtoUCMkMkl8_T78_
Choose your WebID to authorize
http://localhost:8000/almafood/profile/card#me
Remember this client
Authorize
Cancel
Edit account
Use a different account
©2019–2024 Inrupt Inc. and imec

Solid protocols

client-client

Solid Protocol

Draft Community Group Report, 12 May 2024

▸ **More details about this document**

Abstract

This document connects a set of specifications that, together, provide applications with secure and permissioned access to externally stored data in an interoperable way.

Status of This Document

This report was published by the Solid Community Group. It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the W3C Community Contributor License Agreement (CLA) there is a limited opt-out and other conditions apply. Learn more about W3C Community and Business Groups.

1. Introduction

This section is non-normative.

The aims of the Solid project are in line with those of the Web itself: empowerment towards "an equitable, informed and interconnected society". Solid adds to existing Web standards to realise a space where individuals can maintain their autonomy, control their data and privacy, and choose applications and services to fulfil their needs.

The Solid ecosystem encapsulates a set of specifications that are guided by the principles we have adopted and also the priority of our values. We acknowledge that every technical decision has ethical implications both for the end user (short-term) as well as society (long-term). To contribute towards a net positive social benefit, we use the *Ethical Web Principles* [ETHICAL-WEB-PRINCIPLES] to orient ourselves. The consensus on the technical designs are informed by common use cases, implementation experience, and use.

An overarching design goal of the Solid ecosystem is to be evolvable and to provide fundamental affordances for

Type Indexes

Version 1.0.0, Editor's Draft, 2023-03-13

▸ **More details about this document**

Abstract

Type Indexes is a data discovery mechanism where coarse-grained library types are registered providing a way for applications to discover where an agent keeps data relevant for the application.

Status of This Document

This section describes the status of this document at the time of its publication.

This document was published by the Solid Community Group as an *Editor's Draft*. The information in this document is still subject to change. You are invited to contribute any feedback, comments, or questions you might have.

Publication as an *Editor's Draft* does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the W3C Community Contributor License Agreement (CLA). A human-readable summary is available.

1. Introduction

This section is non-normative.

In the Solid ecosystem, storages and clients are loosely coupled. This means clients are tight to data models. Furthermore, clients and thus storages should be able to be connected in a network of knowledge. The connection is only achievable if the data itself is easy to be interconnected. This specification is written for Solid application developers as a solution that shows how to interconnect clients and data on storages. This specification details the use of Type Indexes which were implemented already in different clients. Type Indexes offer a fine-grained approach to describing the location of specific types of resources on a storage. This

Solid Chat

Version 1.0.0, Editor's Draft, 2023-08-21

▸ **More details about this document**

Abstract

A Solid chat channel is a object in a Solid pod which allows a series of text (or multimedia) messages over time to be accumulated as an interactive discussion between one or more people (or other agents) Chat messages are written directly by the participants into a series of chat files organized by date. Participants watch the current chat file for updates in real time using Solid Live Update functionality. A chat channel may also have just one author, in which case it can function as a private "note to self" or as public or shared stream of thoughts. This format is designed to be able to interoperate with existing chat systems like Internet Relay Chat, Matrix, etc. The original version did not include threads or replies, but this one does. Also included is the use of social reactions, for which schema.org's Action class and subclasses are used, with literal emojis as content. This format is also designed to be suitable for archives of channels using other protocols.

Status of This Document

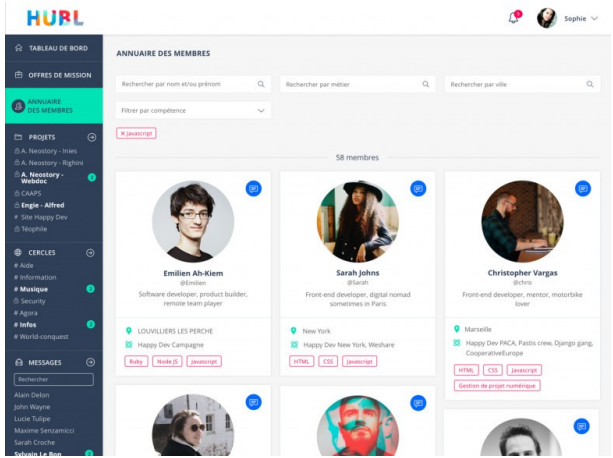
This section describes the status of this document at the time of its publication.

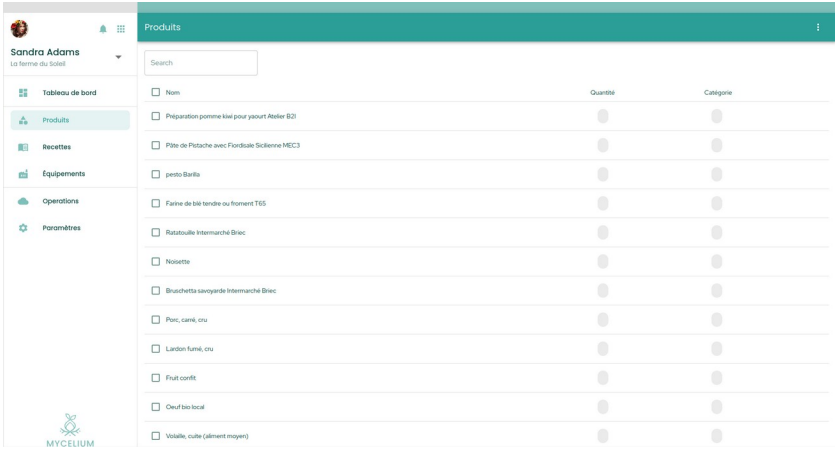
This document was published by the Solid Community Group as an *Editor's Draft*. The information in this document is still subject to change. You are invited to contribute any feedback, comments, or questions you might have.

Publication as an *Editor's Draft* does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the W3C Community Contributor License Agreement (CLA). A human-readable summary is available.

1. Introduction





HUBL

Find freelances given their skills, location or name...

MYCELIUM

Find products, producers, search for orders...

Don’t change the Solid protocol to stay generic

02

Find data in a Solid ecosystem using indexes

Indexes

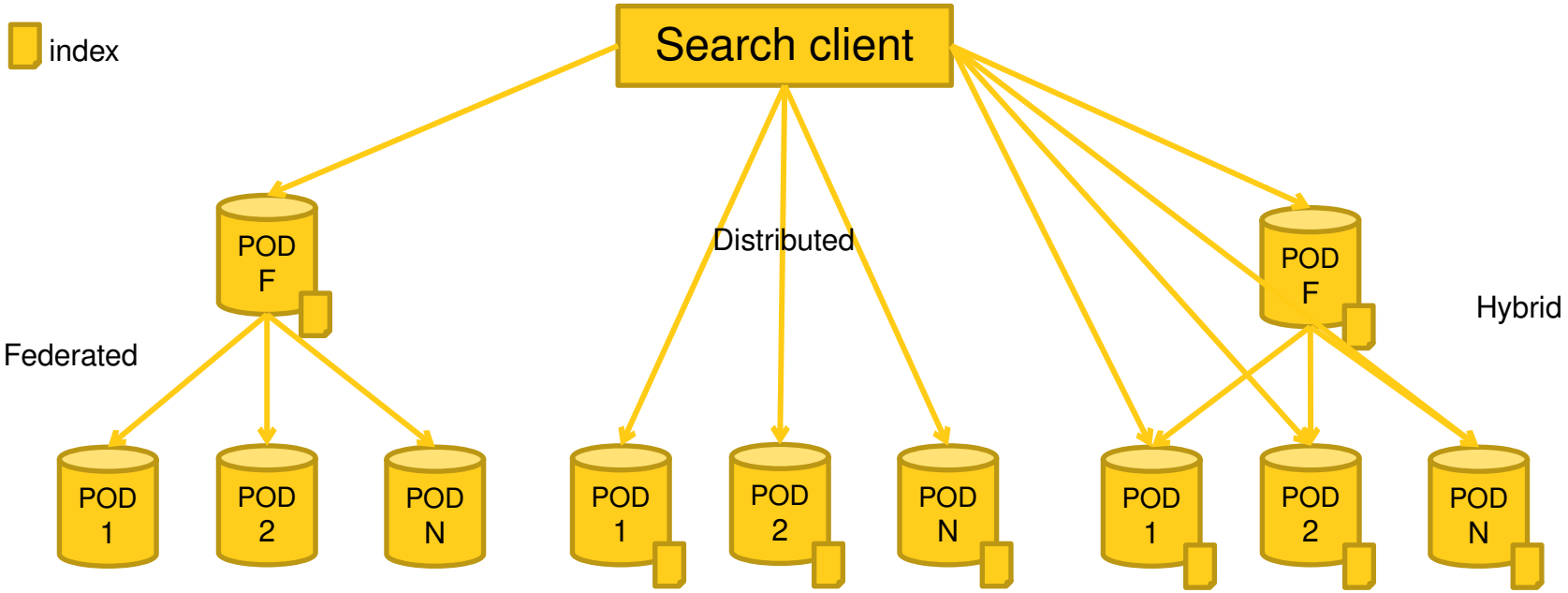
“lists of keywords with pointers to where further information about the keyword is found.”

- An index summarizes/characterizes the content and its location
- Used to :
  - > Find out what we have and where it is
  - > Plan and be faster to answer (distributed) queries
  - > Source selection against various criteria
  - > ...

Example of an index

```
<> a ex:SkillIndex;
    ex:forSkill ex:PHP;
    ex:entry <http://localhost:8001/user1/profile/card#me>,
    <http://localhost:8008/user1/profile/card#me>,
    <http://localhost:8010/user1/profile/card#me>,
    <http://localhost:8012/user1/profile/card#me>,
    <http://localhost:8014/user1/profile/card#me>,
    <http://localhost:8016/user1/profile/card#me>,
    <http://localhost:8023/user1/profile/card#me>.
```

Index distribution

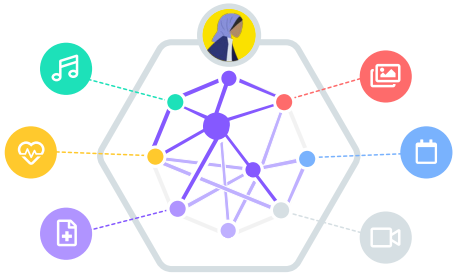


Indexing by Solid clients

In Solid, indexing could be done by applications on the client side either by browser applications or agents (bots).

- Each time a data is created/modified/deleted, applications should update the appropriate indexes.
- Applications could also perform indexing on demand.

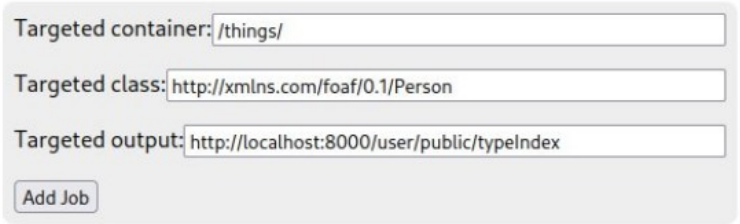
- Agents could be used to extend Solid servers capabilities.
- Indexing could be delegated to agents (ex: Solid indexer).
- Querying could be delegated to agents, using specific technologies.



Solid Indexer

Solid Indexer runs indexing jobs created by the user and writes back the results (index) in the user's storage.

- Interface for an indexing job is :
  - > A container to be crawled
  - > A class to be indexed
  - > A file to write the results back
- The jobs are stored on the user's POD.



Solid Indexer

1

Targeted container: /things/

Targeted class: http://xmlns.com/foaf/0.1/Person

Targeted output: http://localhost:8000/user/public/typeIndex

Add Job

2

List of jobs:

Job: http://localhost:8000/user/solid-indexer/jobs/a59040ed-d05b-4fc5-86aa-dccab8d063e8
Container: */things/*
Target: *http://xmlns.com/foaf/0.1/Person*
Output: *http://localhost:8000/user/public/typeIndex*

http://localhost:8000/solid-indexer/profile/card#me Start job

```
New Job received...

Found a job to do in the notification at: <http://localhost:8000/user/s
-dccab8d063e8>.

3

Starting Job(http://localhost:8000/user/things/, http://xmlns.com/foaf/
blic/typeIndex).

Resource <http://localhost:8000/user/things/thing1#1> has been indexed.
Resource <http://localhost:8000/user/things/thing2#1> has been indexed.
Resource <http://localhost:8000/user/things/thing2#2> has been indexed.
Job is complete.
The Job result has been successfully saved to the user's storage.
```

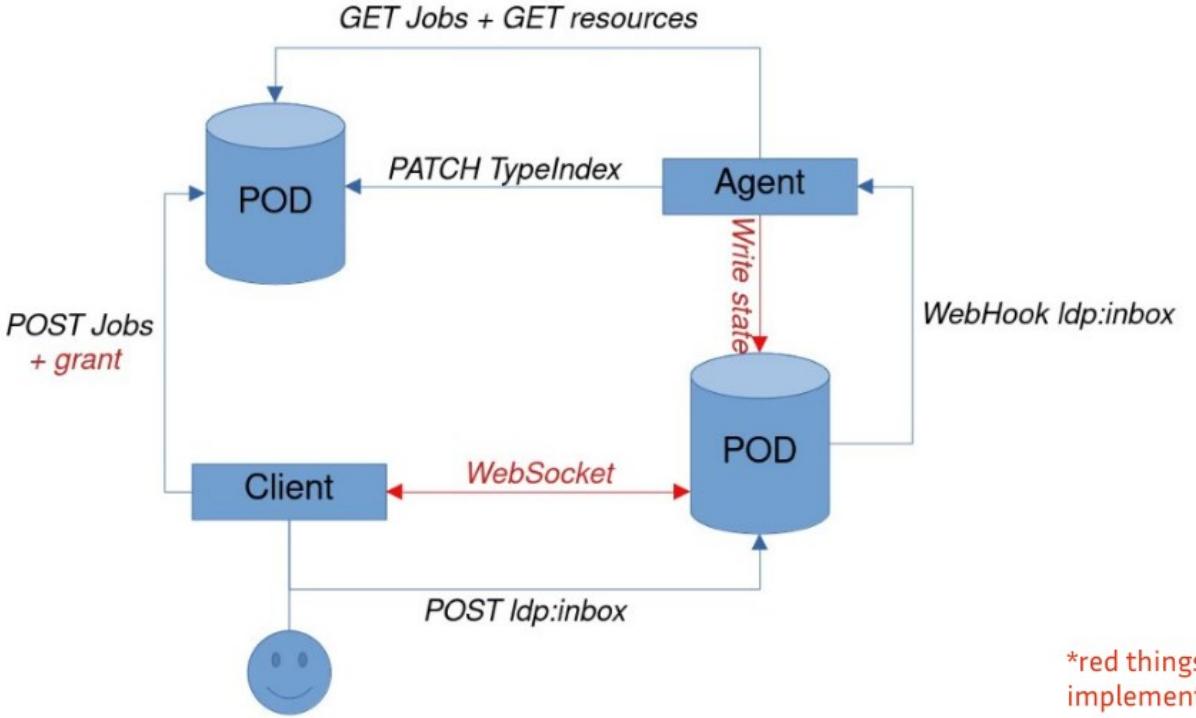
TypeIndex content

Read TypeIndex

4

```
@prefix solid: <http://www.w3.org/ns/solid/terms#>.

<http://localhost:8000/user/public/typeIndex> a solid:TypeIndex, solid:ListedDocument.
<#42ca174d-4ddb-4f3c-ad28-91c7e5de9181> a solid:TypeRegistration;
    solid:forClass <http://xmlns.com/foaf/0.1/Person>;
    solid:instanceContainer <http://localhost:8000/user/things/>.
<#845bc859-fbba-4d86-9f64-dbd3c4d70322> a solid:TypeRegistration;
    solid:forClass <http://xmlns.com/foaf/0.1/Person>;
    solid:instance <http://localhost:8000/user/things/thing1#1>.
<#ce75ecd6-d217-42eb-8647-49791cb754b1> a solid:TypeRegistration;
    solid:forClass <http://xmlns.com/foaf/0.1/Person>;
    solid:instance <http://localhost:8000/user/things/thing2#1>.
<#54c1da7c-e511-4379-80fc-6622a5c8bf66> a solid:TypeRegistration;
    solid:forClass <http://xmlns.com/foaf/0.1/Person>;
    solid:instance <http://localhost:8000/user/things/thing2#2>.
```

Solid Indexer



03

Comparing indexing strategies

Benchmark framework

A browser app to compare indexing strategies and a set of python scripts to deploy Solid servers and generated indexes from input data.

- Docker containers
- Use Comunity Solid Server
- Test federated and distributed indexes querying
- Use Comunica as the default query engine

https://comunica.dev/

Inria

2. Select indexing strategies

Select the strategies you want to compare for the query.

☐ Skill (federated)
Query the global indexes to find users with the given skills (cities are ignored).

See/hide SPARQL query

See/hide targeted sources (0)

☐ Skill (distributed)
Query the local indexes to find users with the given skills (cities are ignored).

See/hide SPARQL query

See/hide targeted sources (0)

☐ City (federated)
Query the global indexes to find users with the given city (skill are ignored).

See/hide SPARQL query

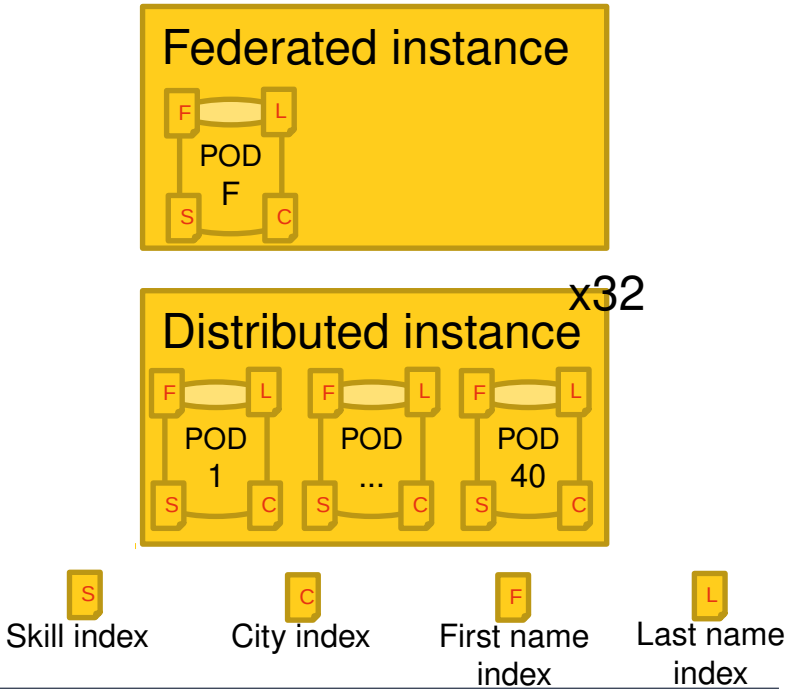
See/hide targeted sources (0)

- *Skill*
- *City*
- *Skill with traversal*
- *City with traversal*
- *Skill and city*
- *Skill and city with traversal*
- *Skill with traversal filtered by city*
- *City with traversal filtered by skill*
- *Skill with traversal and index discovey (federated) - hard coded skill*
- *Skill with named graph traversal and index discovey (federated) - hard coded skill*
- *Skill and city from distributed meta indexes with traversal (distributed)*

Test on Startin'Blox Hubl replica

1280 PODs, 600 skills, 10 cities

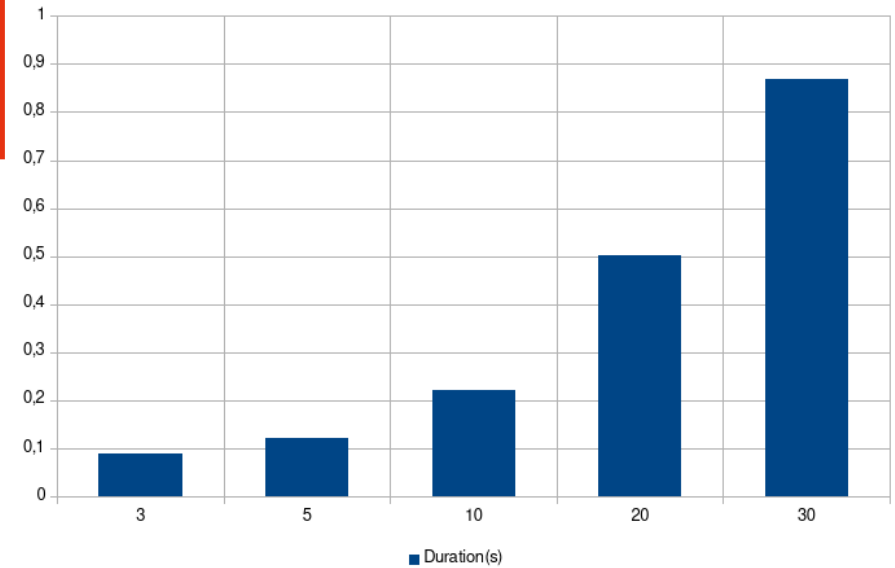
- 32 (+1) Solid server
- Each server hosts 40 PODs
- 1 user has one POD on one instance
- 1 user has [1 ; 10] skill(s)
- 1 user has one city

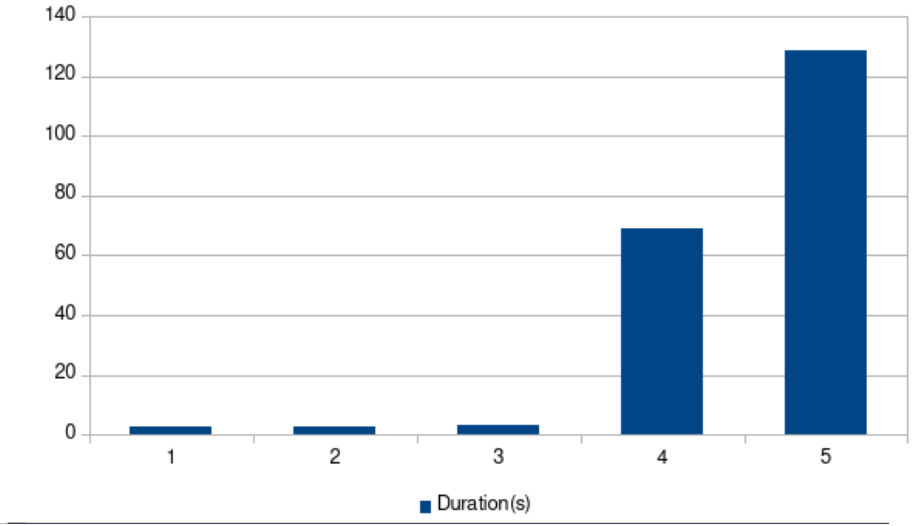


Experimental results

Federated/distributed skills search

```
SELECT DISTINCT ?user WHERE { ?skillIndex a ex:SkillIndex; ex:forSkill ex:someSkill; ex:entry ?user }
```

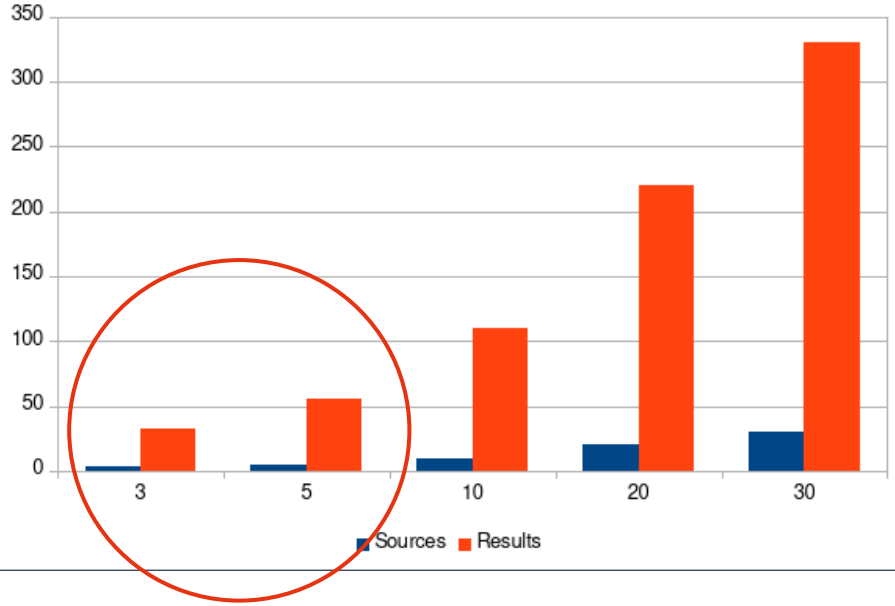


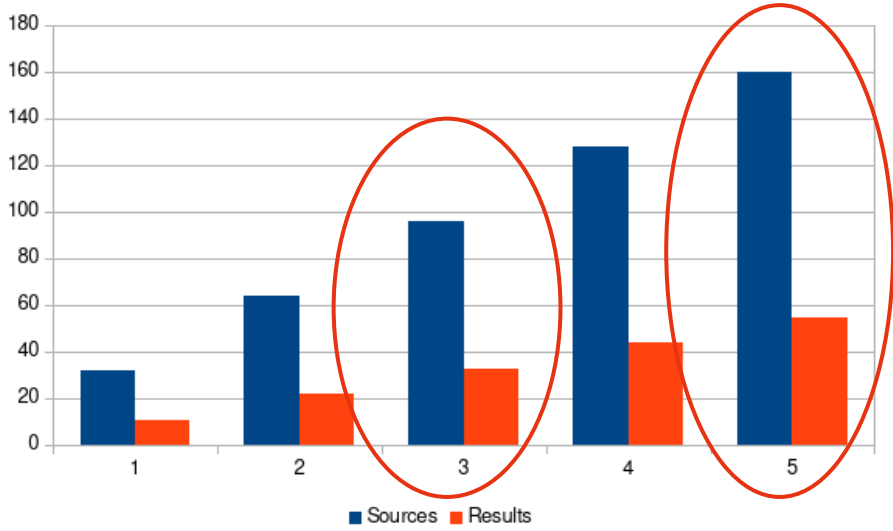


Experimental results

Federated/distributed skills search

```
SELECT DISTINCT ?user WHERE { ?skillIndex a ex:SkillIndex; ex:forSkill ex:someSkill; ex:entry ?user }
```

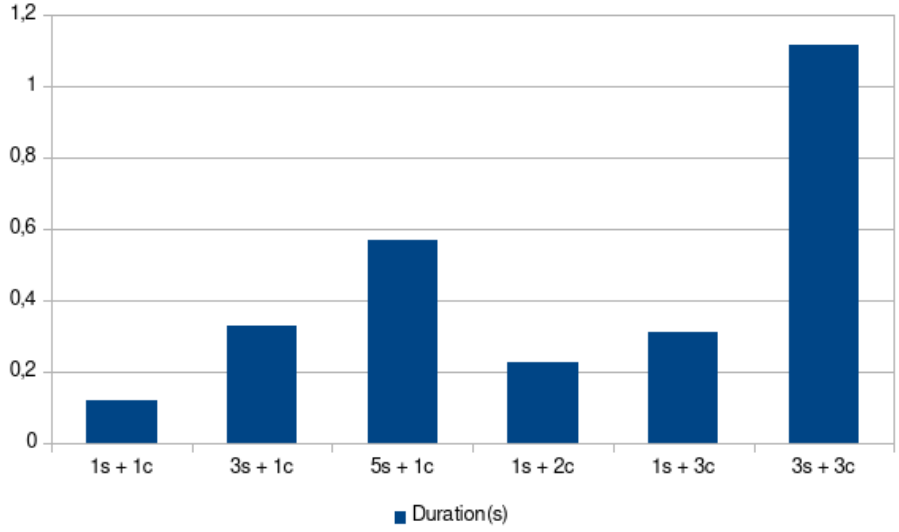


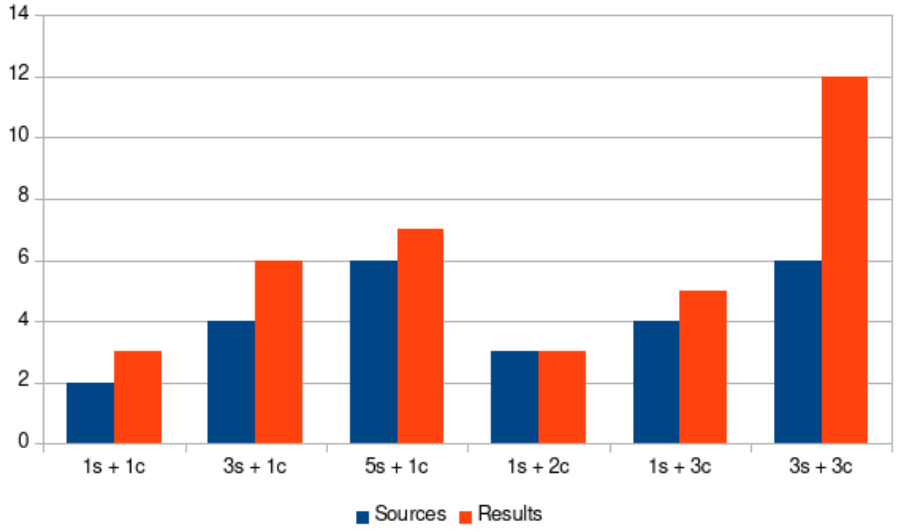


Experimental results

Federated skills and cities

```
SELECT DISTINCT ?user WHERE { ?skillIndex a ex:SkillIndex; ex:forSkill ex:someSkill; ex:entry ?user. ?
cityIndex a ex:CityIndex; ex:forCity ex:someCity; ex:entry ?user. }
```



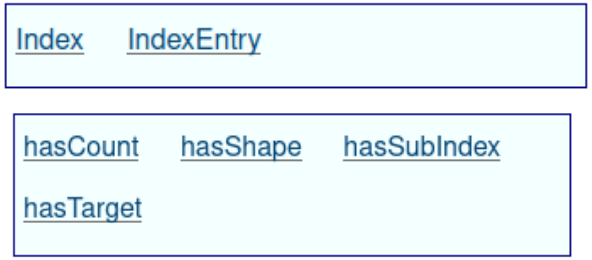


04

Implementations

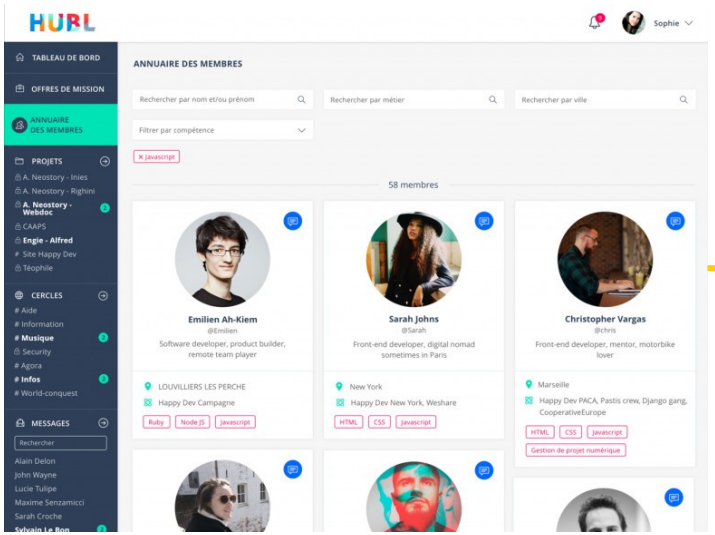
Shape-based indexing RDF vocabulary

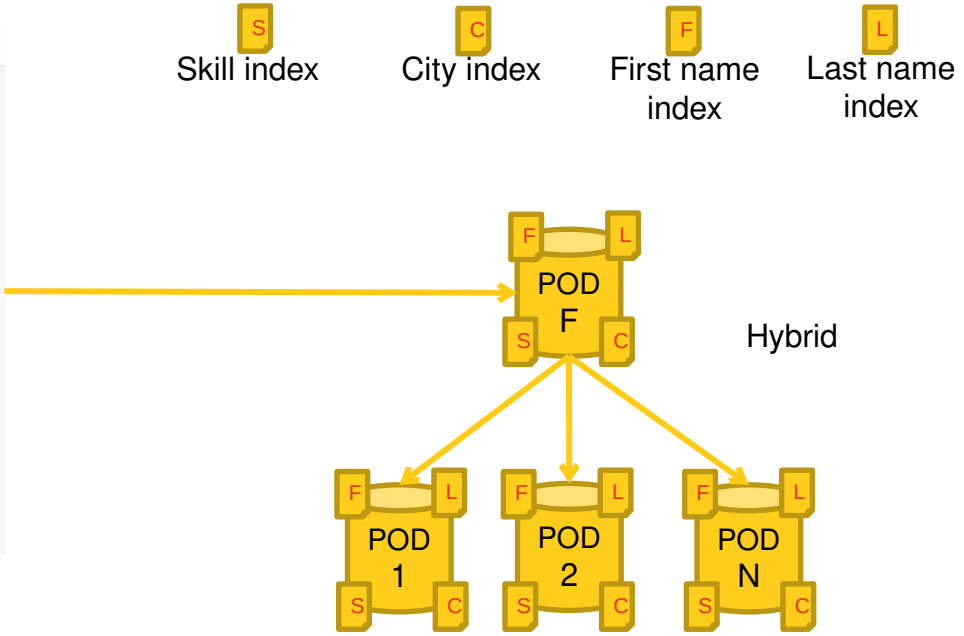
- Indexes and meta-indexes
- Recursive querying
- Use reduced shape vocabulary :
  - sh :path, sh :hasValue, sh :pattern




```
<> a idx:Index.

<#shape> a sh:NodeShape;
    sh:closed false;
    sh:property [
        sh:path rdf:type;
        sh:hasValue foaf:Person
    ];
    sh:property [
        sh:path schema:hasLocation;
        sh:hasValue "Paris"
    ].

<#entry1> a idx:IndexEntry;
    idx:hasShape <#shape>;
    idx:hasTarget <https://alice.solidcommunity.net/profile/card#me>.

<#entry2> a idx:IndexEntry;
    idx:hasShape <#shape>;
    idx:hasTarget <https://bob.solidcommunity.net/profile/card#me>.
```

Draft at https://wimmics.github.io/solid-start/specs/solid-indexing/

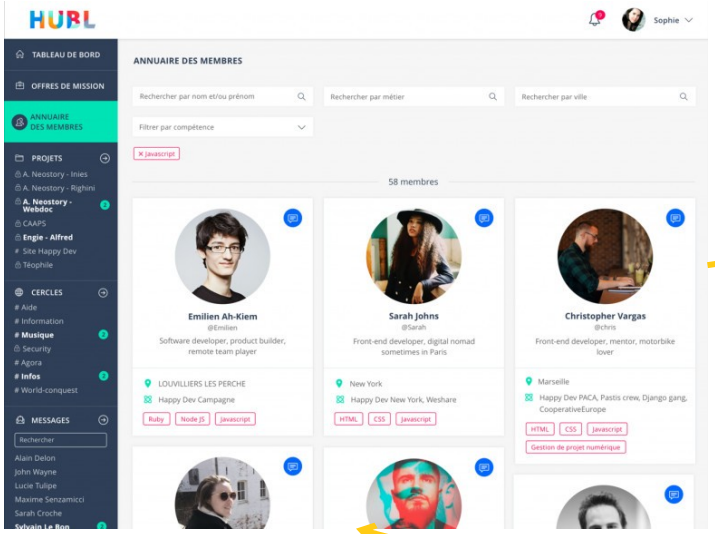


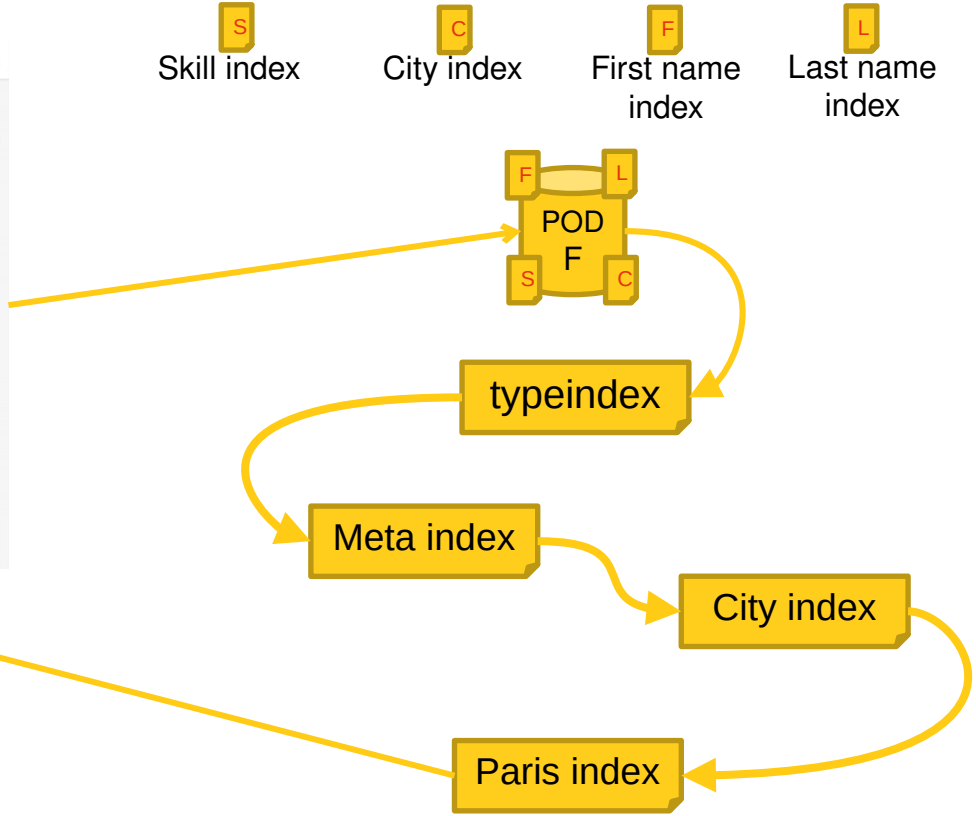


HUBL

Find freelances given their skills, location or name...

Federated indexes are constructed by a crawler





HUBL

Ex : Find freelances from the location “Paris”

Semantizer

- TypeScript library to manipulate RDF datasets
- Modular architecture, mixins, extendable
- Organize business logic code
- Use RDFJS by default
- MIT license

Mixin	Description
changelog	A mixin to record the changes made to a dataset.
dataset	The base mixin. Provides essential methods.
foaf-person	The foaf:Person mixin.
index	A mixin to query indexes.
shacl	A mixin to manipulate SHACL shapes.
solid-changelog-n3	A mixin to serialize a dataset changelog into a N3 patch.
solid-container	A mixin to manipulate Solid containers.
solid-webid	A mixin to manipulate Solid WebId profiles.
typeindex	A mixin to manipulate TypeIndexes.
webid	A mixin to manipulate WebId profiles.

Util
index-entry-stream-transformer
index-querying-strategy-shacl
index-querying-strategy-shacl-comunica
index-querying-strategy-shacl-conjunction
index-querying-strategy-shacl-final
loader-base
shacl-validator-default

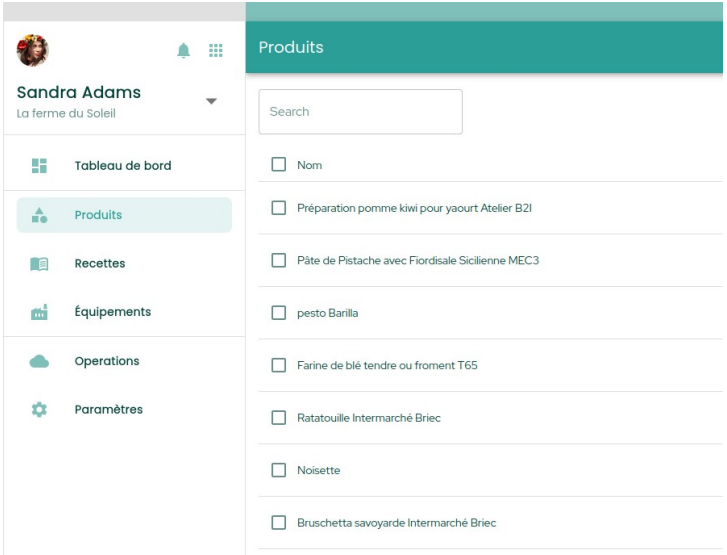
```
const index = await semantizer.load("indexUrl", IndexMixinFactory);
const readable = index.query(strategy); // results stream
```

github.com/semantizer/semantizer-typescript/tree/dev

ALMA FOOD

MYCELIUM Recipes

A Solid app to manage products, recipes and devices



Find products by name:
+2500 products
9160 words indexed
Constructed in ~2 minutes

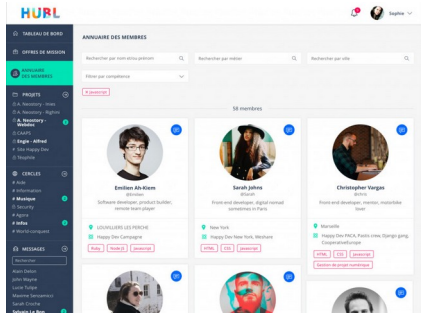
05

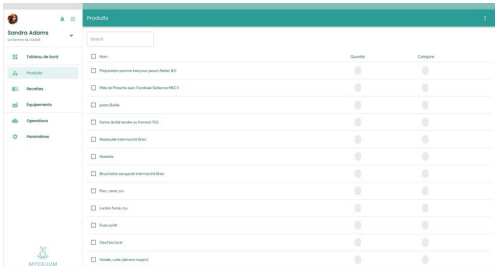
Future and perspectives

10/2022 ⟶ 10/2024

@Wimmics team at INRIA + @Startin'Blox (startinblox.com)

- Solid indexer
- Benchmark framework
- Shape-based indexing RDF vocabulary
- Semantizer





Next?

- Publish a final draft of the index vocabulary
- Handle pagination and query resuming
- Publish a Solid production ready indexing agent (crawler / use notifications)
- Publish our test framework as a Solid app

- Finish + deploy Mycelium recipes
- Finish + publish the DFC client-client protocol for data discovery and indexing
- Continue the indexing integration into the Startin'Blox framework...

Contact

github.com/Wimmics/solid-start
github.com/datafoodconsortium

almafood.fr
datafoodconsortium.org
solidproject.org

maxime@lecoqlibre.fr
@lecoqlibre on GitHub and Solid Gitter

Inria