eosc | FAIRCORE4EOSC
Core Components Supporting a FAIR EOSC

Archive your software to Software Heritage.

The example of an EOSC-Core service

Themis Zamani (GRNET)
Agelos Tsalapatis (GRNET)
Kostas Koumantaros (GRNET)

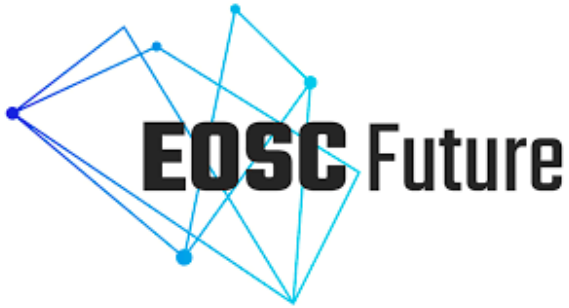
Funded by the European Union

WP6 Partners

grnet

Set up of **regular archival** in Software Heritage for **EOSC-core** code hosting platforms

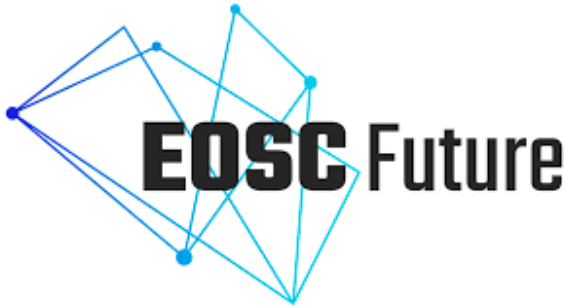
Objectives



Set up of **regular archival** in Software Heritage for **EOSC-core** code hosting platforms

Find repositories

Objectives



Set up of **regular archival** in Software Heritage for **EOSC-core** code hosting platforms

Software Heritage

Find repositories

How to archive

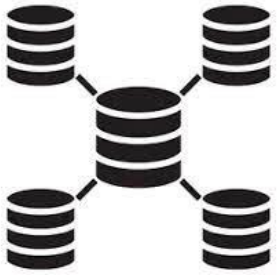
Objectives

EOSC Future

Set up of **regular archival** in Software Heritage for **EOSC-core** code hosting platforms

Software Heritage

repositories ← - - - - - - - - - - → archival ← - - - - - - - - - - - - - - - - - → **archive**



The Archival Script

The What

Archiving EOSC Core components repositories

The script reads a list of repositories sources alongside with their type (git,svn.etc) and attempts to archive each single one of them through the software heritage api.

Prerequisites : The List

Prerequisites : Understand the common functions

The Archival Script

Archiving EOSC Core components repositories

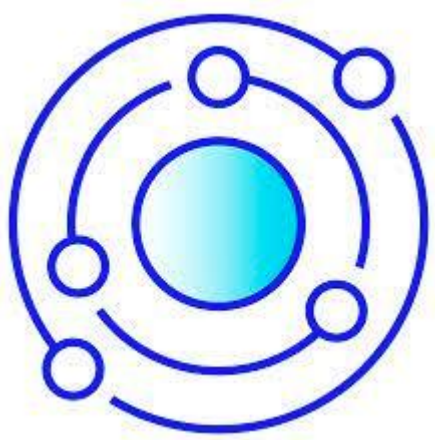
The Process

- Get a list of repositories to check
- Use the visit api call to determine if the repository is already archived.
- Use the save api call to either get the status or begin the process of archival for a missing one.
- Produce a report containing required information

Next Steps

- Archive the repositories with the related metadata. (deposit action)
- Get correct list of repos

Main functionalities



The repository archiving script is a bridging component that provides

The following functionality:

- Reads as file input a list of version control urls that correspond to various repositories and will attempt to que them for archival through the software heritage API
- The first action is to utilise the ***VISIT*** api call that returns information about the archival of a repository.
- If a repository can be visited, we collect some additional information through the ***STATUS*** api call that will used in the last action of the script .
- In the other hand , if the repository does not have any information present in the software heritage system , we use the ***ARCHIVE*** api call to que the repository for archival .Repositories from known sources like GitHub are picked up and queued for archival automatically.
- The final step of the script is the compilation of a results file report that contains information about the status of the archival of all the provided repositories.

The Archival Report

The Report

The report is a file where its line represents the archival data of one of the provided repositories

1: **4066**, **https://example.com**, **git**, **accepted**, **not yet scheduled**, **2023-06-24T13:37:17**

- **The id of the archival request assigned by the SWH**
- **The url of the repo to be archived**
- **The version control system**
- **The status of the request**
- **The status of the archival**
- **The timestamp of the archival request**

eosc

EOSC Future

Software Heritage

1st question

how can we search for the new repositories?

2nd question

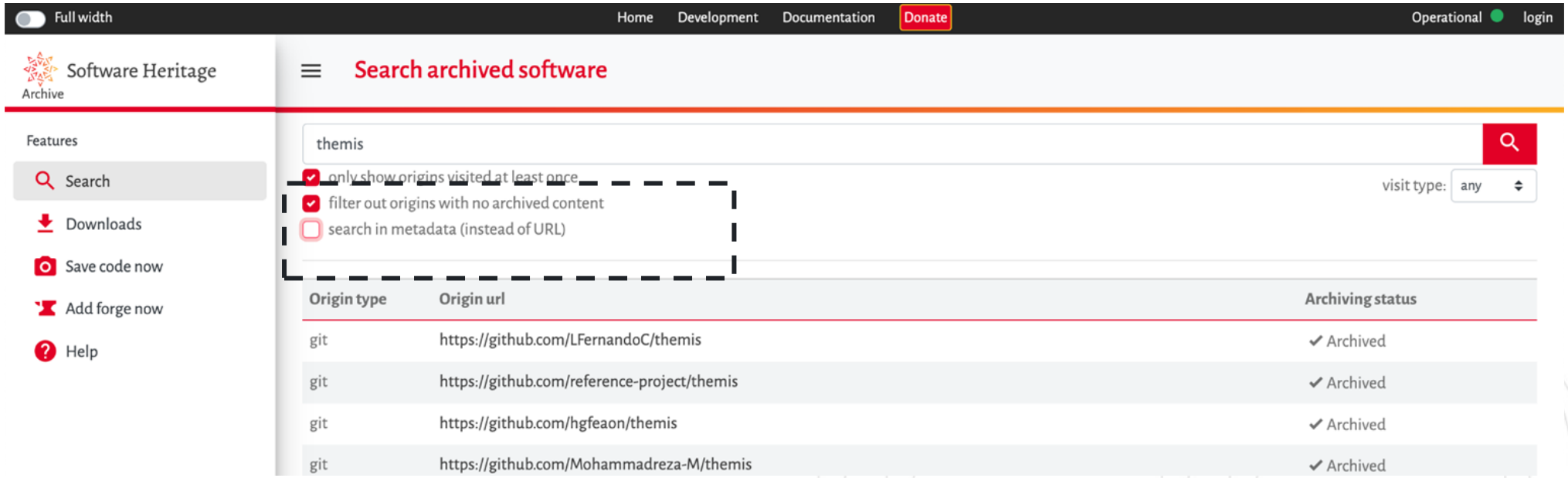
how can we add more information for the repositories so that everyone can easily find the code repository based on various metadata?

EOSC Future

Software Heritage

https://archive.softwareheritage.org/browse/search/

1st question how can we search for the new repositories?



eosc | FAIRCORE4EOSC
Core Components Supporting a FAIR EOSC

EOSC Future

Software Heritage

2nd question

how can i add more information for the repositories so that everyone can easily find the code repository based on these description ?

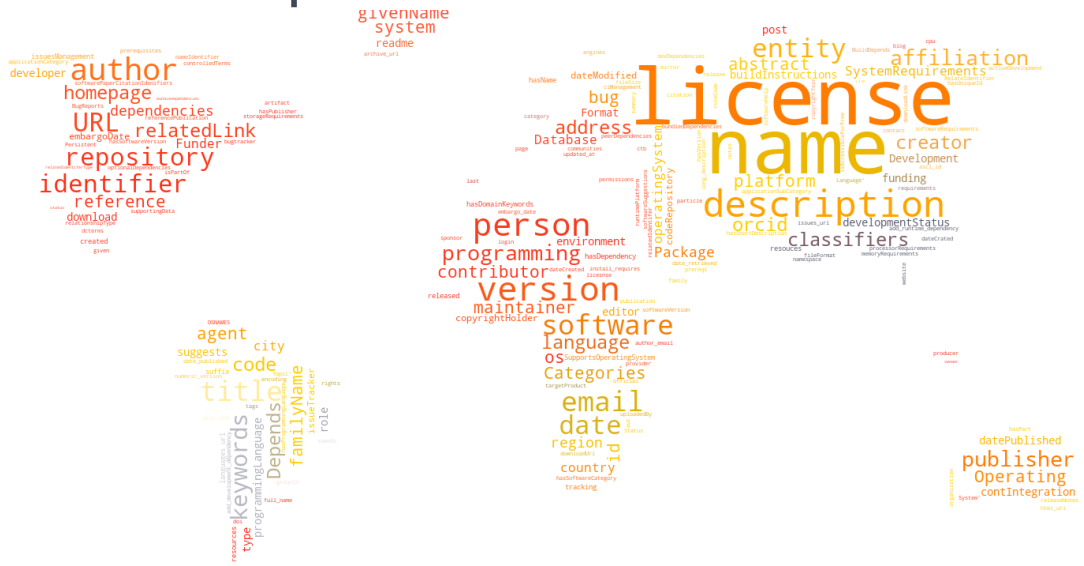
LICENSE	-rw-r--r--	25.8 KB
Makefile	-rw-r--r--	439 bytes
README.md	-rw-r--r--	7.1 KB
codemeta.json	-rw-r--r--	1.5 KB

The Solution: CodeMeta

Easy to discover the software projects you may be interested in

The goal of CodeMeta is to create a concept vocabulary that can be used to standardize the exchange of software metadata across repositories and organizations.

A CodeMeta instance file describes the metadata associated with a software object using JSON’s linked data (JSON-LD) notation.



What is CodeMeta in Software Heritage?

Easy to discover the software projects you may be interested in

At Software Heritage, they decided early on to use the CodeMeta vocabulary internally when indexing metadata.

The adoption of the CodeMeta representation, in the form of codemeta.json files, for describing landmark legacy software in the framework of the Software Heritage Acquisition Process

I am a repository owner

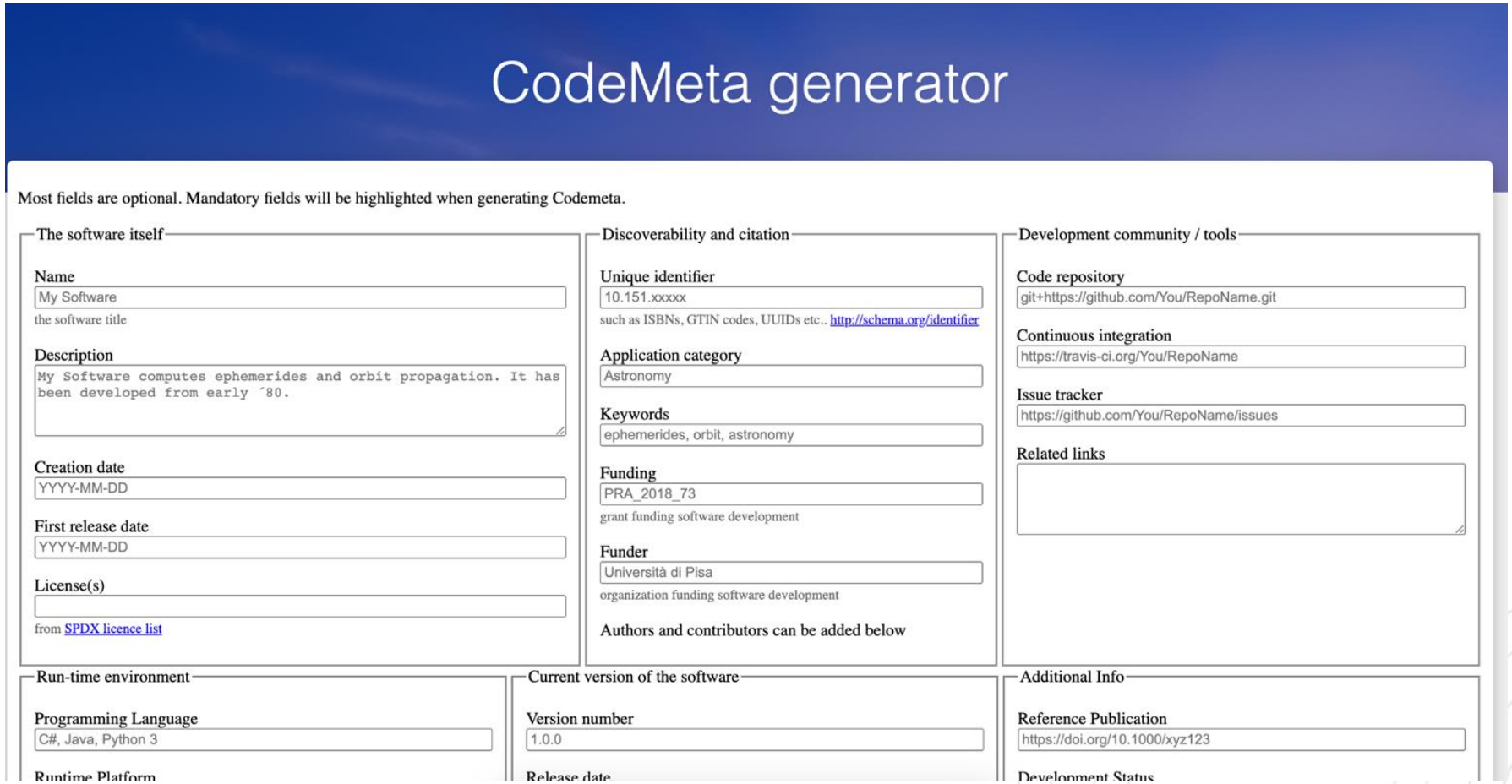
Is the script responsible for everything? What should i do?

The answer **Create the metadata.json file for your repository**

The Service Owners are the ones that know the information

It is a 10 minutes work to make your software famous :)

CodeMeta Generator https://codemeta.github.io/codemeta-generator/



CodeMeta metadata

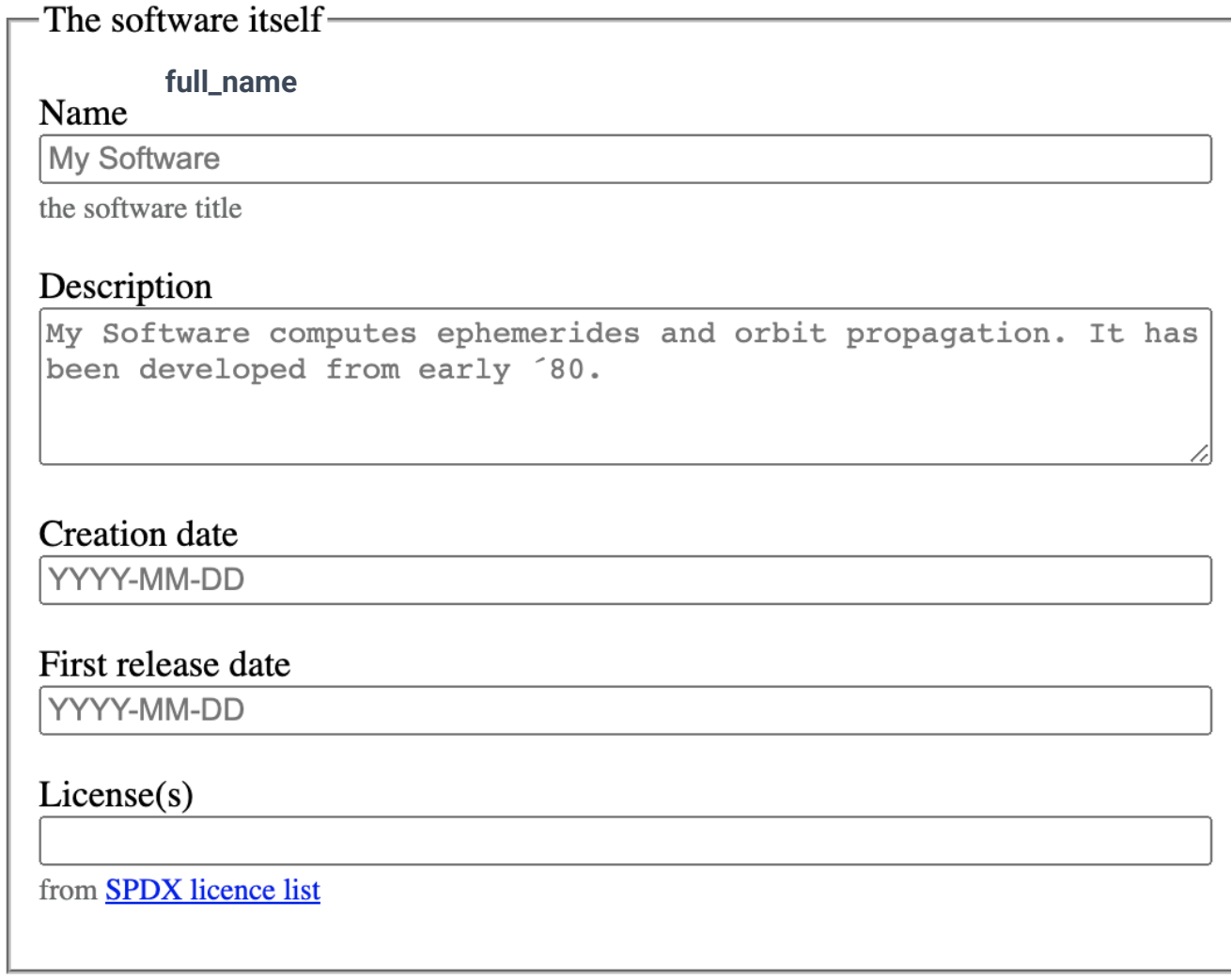
The Software itself

Syntax:
`https://api.github.com/repos/{:owner}/{:repository}`

example : https://api.github.com/repos/argoeu/argo-messaging

```
svn info filename
svn propget --revprop -r 0 svn:date
http://svn.apache.org/repos/asf
```

```
"html_url": "https://github.com/ARGOeu/argo-messaging",
"description": "The ARGO Messaging API is implemented as
a Publish/Subscribe Service. Instead of focusing on a
single Messaging API specification for handling the logic
of publishing/subscribing to the broker network the API
focuses on creating nodes of Publishers and Subscribers
as a Service.",
```



Use SPDX license identifier

CodeMeta metadata / GITHUB names

Property	GitHub
codeRepository	html_url
programmingLanguage	languages_url
downloadUrl	archive_url
author	login
dateCreated	created_at
dateModified	updated_at
license	license
description	description
identifier	id
name	full_name
issueTracker	issues_url

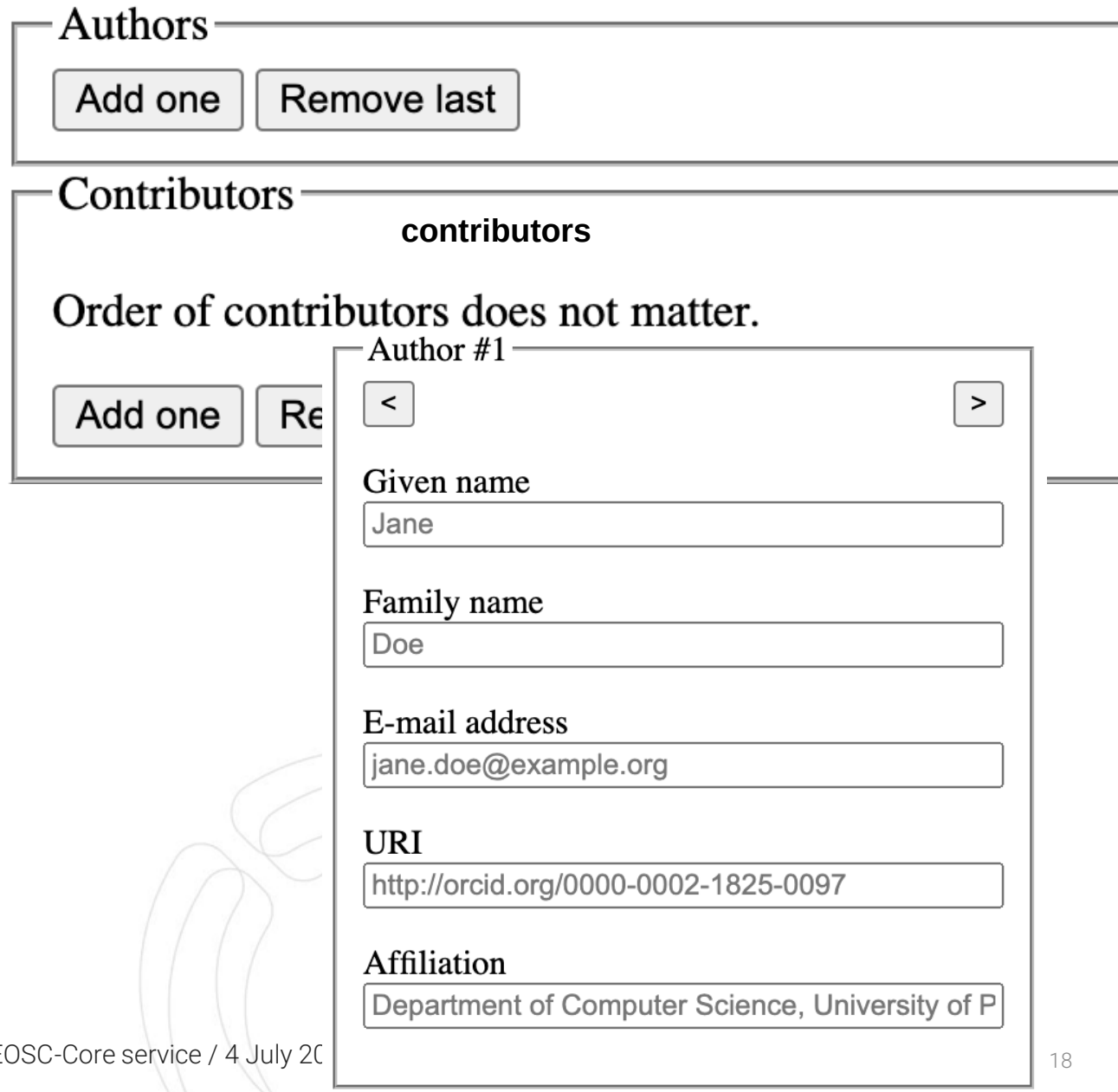
CodeMeta metadata

The Authors / Contributors

You can add as many

- authors
- contributors

as you want.

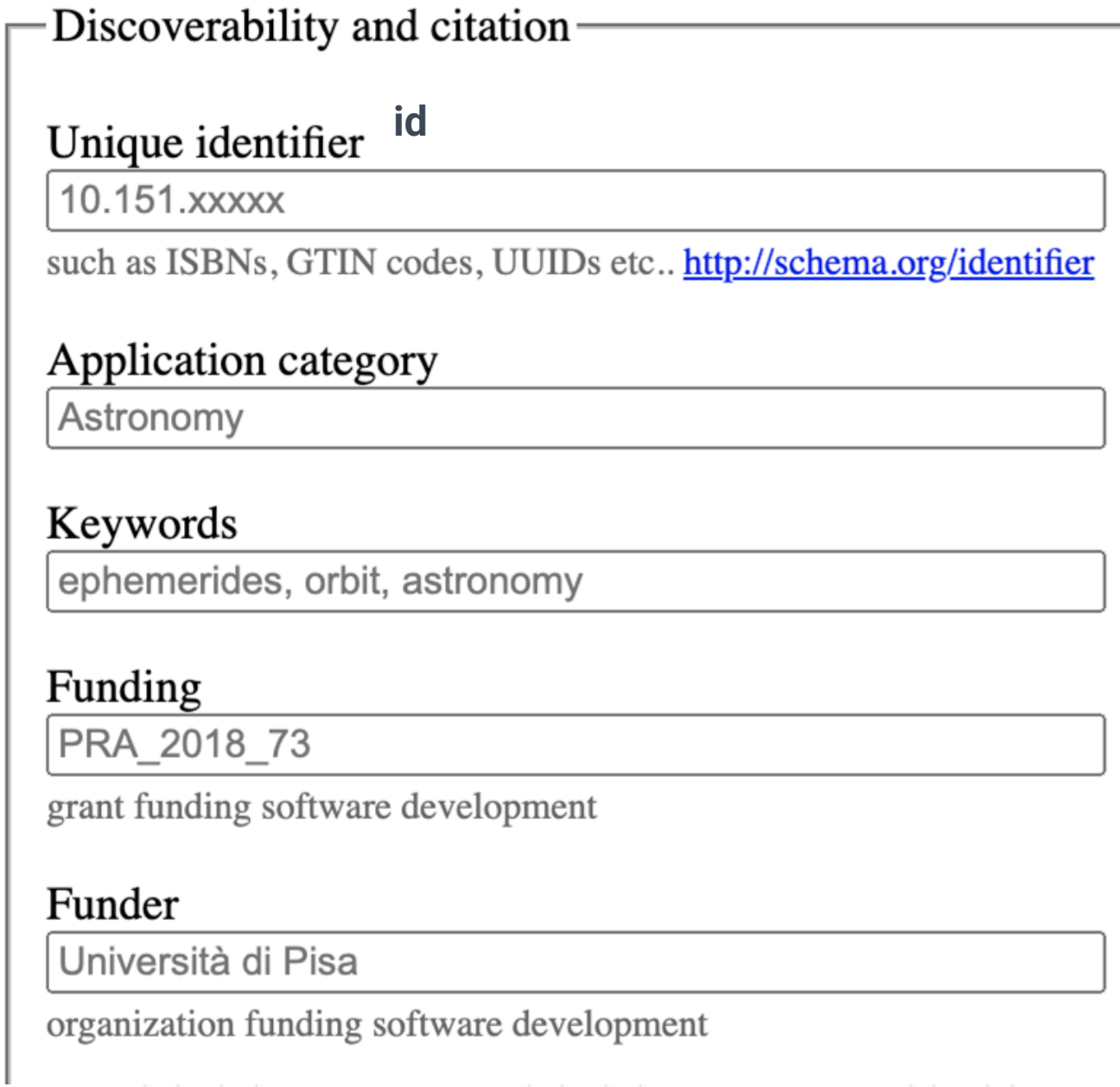


CodeMeta metadata

Discoverability /. Citation

This is information that the service owner should add so that potential users / developers can see

- application category
- keywords
- Funding
- Funder

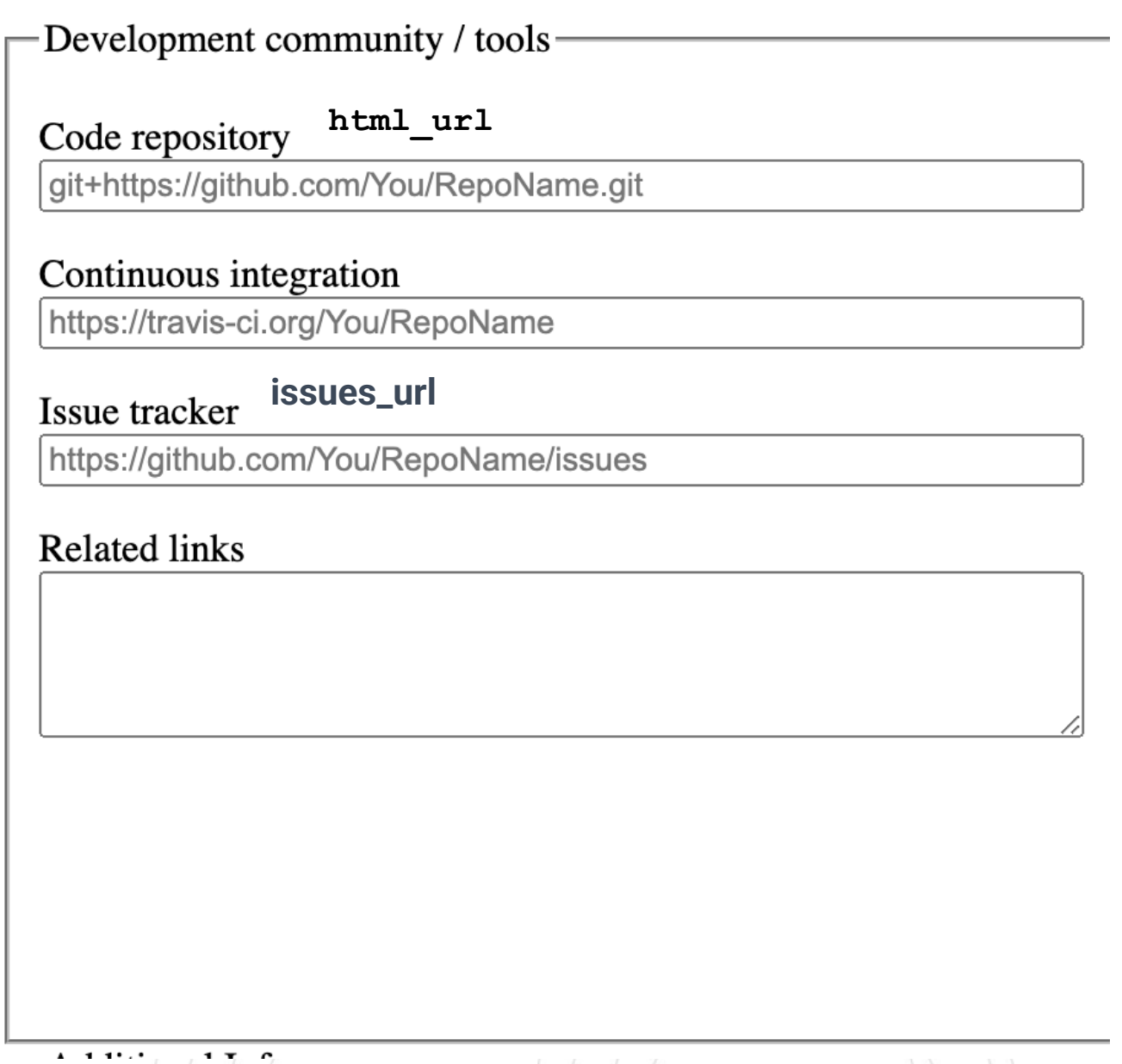


CodeMeta metadata

Development community / tools

This information can be added

- from the repo automatically
- manually by the service owner



CodeMeta metadata

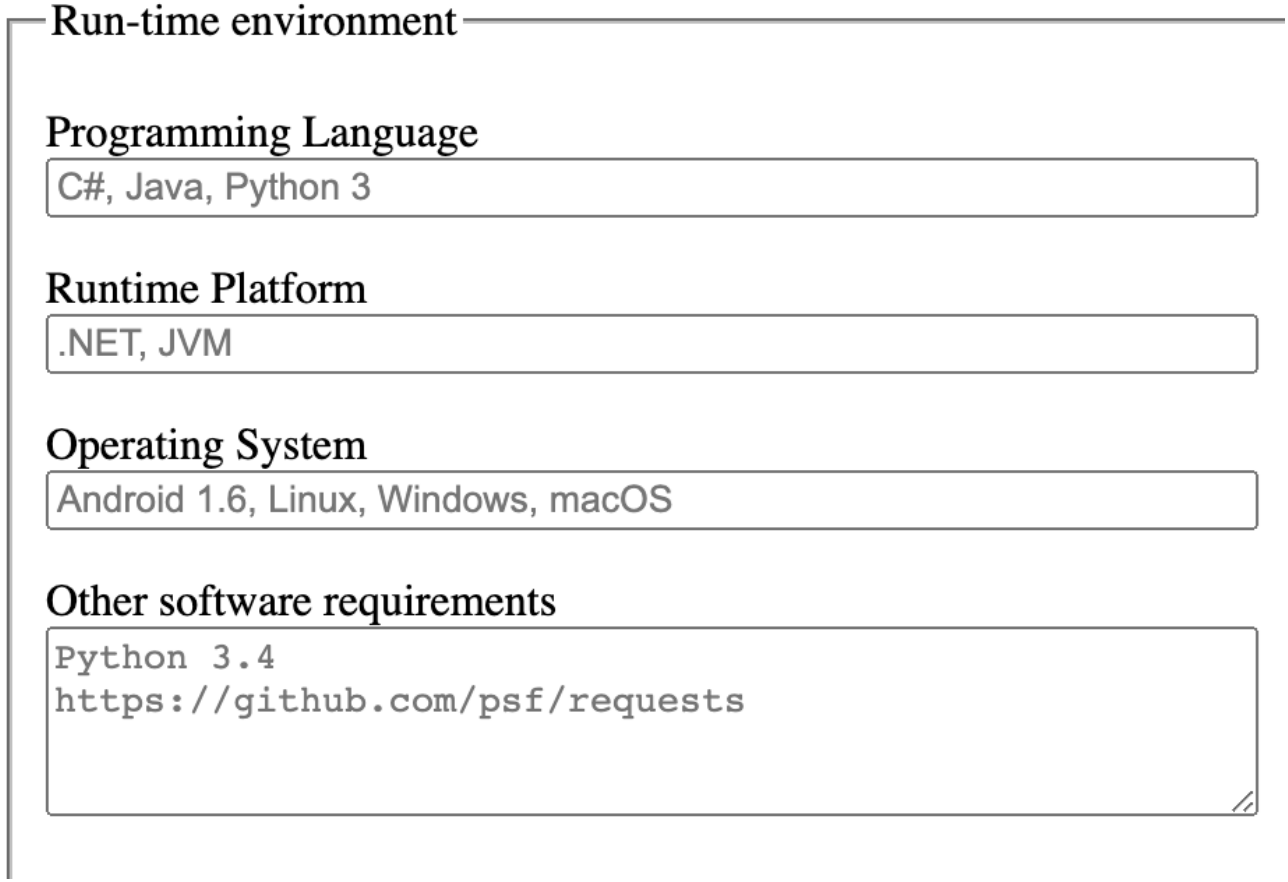
Run-time Environment

This information can be added

- from the repo automatically
- manually by the service owner

Syntax:
`https://api.github.com/repos/{:owner}/{:repository}/languages`

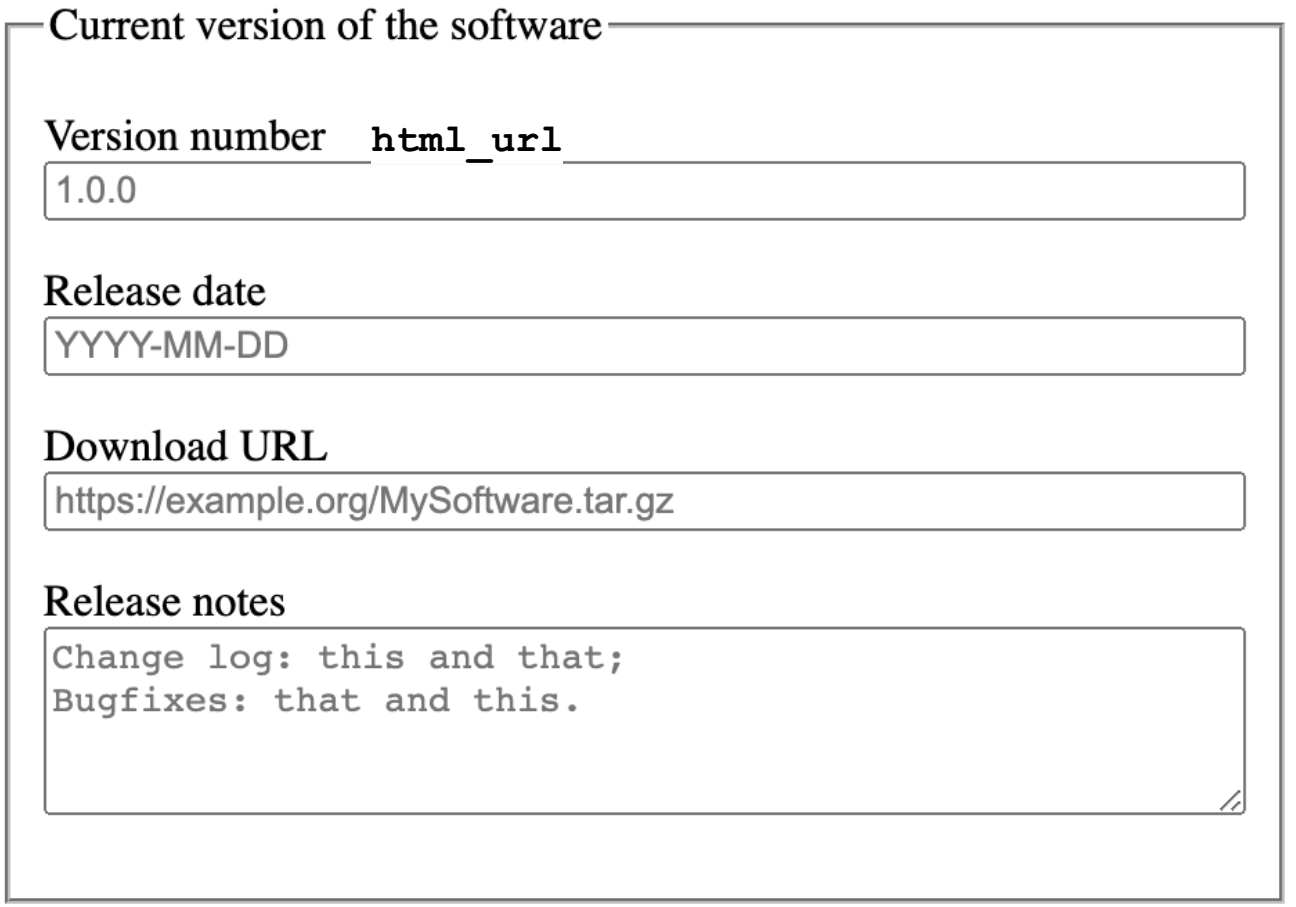

```
{
    "Go": 825574,
    "Python": 43705,
    "JavaScript": 7956,
    "HTML": 3458,
    "Makefile": 1634,
    "CSS": 1596
}
```

CodeMeta metadata

Current Version

This information is used to have a view of the current version of the software .



CodeMeta metadata

Additional Info

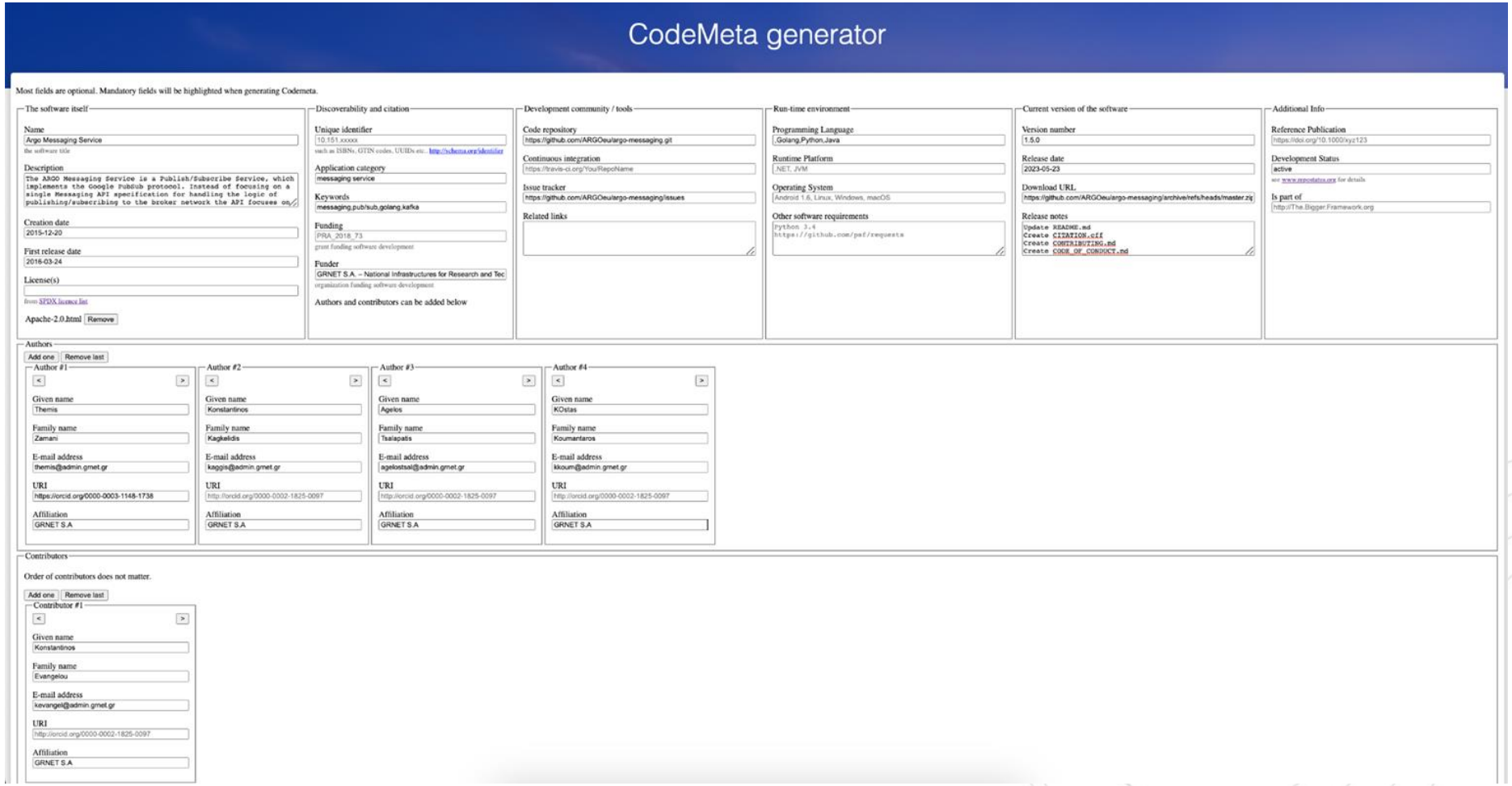
This information is used to connect sub-components of a specific component , or to connect to a bigger project

Status

- **WIP** – Initial development is in progress, but there has not yet been a stable, usable release suitable for the public.
- **Active** – The project has reached a stable, usable state and is being actively developed.
- **Inactive** – The project has reached a stable, usable state but is no longer being actively developed; support/maintenance will be provided as time allows.

CodeMeta JSON File

The EOSC Messaging example



CodeMeta JSON File The EOSC Messaging example


```
"@context": "https://doi.org/10.5063/schema/codemeta-2.0",
"@type": "SoftwareSourceCode",
"license": "https://spdx.org/licenses/Apache-2.0.html",
"codeRepository": "https://github.com/ARGOeu/argo-messaging.git",
"dateCreated": "2015-12-20",
"datePublished": "2016-03-24",
"dateModified": "2023-05-23",
"downloadUrl": "https://github.com/ARGOeu/argo-messaging/archive/refs/heads/master.zip",
"issueTracker": "https://github.com/ARGOeu/argo-messaging/issues",
"name": "Argo Messaging Service",
"version": "1.5.0",
"description": "The ARGO Messaging Service is a Publish/Subscribe Service, which implements the Google PubSub protocol. Instead of focusing on a single Messaging API specification for handling the logic of publishing/subscribing to the broker network the API focuses on creating nodes of Publishers and Subscribers as a Service. It provides an HTTP API that enables Users/Systems to implement message oriented services using the Publish/Subscribe Model over plain HTTP. In the Publish/Subscribe paradigm, Publishers are users/systems that can send messages to named-channels called Topics. Subscribers are users/systems that create Subscriptions to specific topics and receive messages.",
"applicationCategory": "messaging service",
"developmentStatus": "active",
"isPartOf": "https://eosc-portal.eu",
"funder": {
    "@type": "Organization",
    "name": "GRNET S.A. – National Infrastructures for Research and Technology"
}
```

CodeMeta JSON File

The EOSC Messaging example


```
  "keywords": [
     "messaging","pub/sub", "golang","kafka"
  ],
  "programmingLanguage": [
     "Golang","Python", "Java"
  ],
  "author": [
     {
        "@type": "Person",
        "@id": "https://orcid.org/0000-0003-1148-1738",
        "givenName": "Themis",
        "familyName": "Zamani",
        "email": "themis@admin.grnet.gr",
        "affiliation": {  "@type": "Organization",  "name":
"GRNET S.A"
        }},{
        "@type": "Person",
        "givenName": "Agelos",
        "familyName": "Tsalapatis",
        "email": "agelostsal@admin.grnet.gr",
        "affiliation": {   "@type": "Organization",  "name":
"GRNET S.A "} },
        {
           "@type": "Person",
           "givenName": "Konstantinos",
           "familyName": "Kagkelidis",
           "email": "kaggis@admin.grnet.gr",
           "affiliation": {
              "@type": "Organization",
              "name": "GRNET S.A "
           }
        },
        ],
     "contributor": [
        {
           "@type": "Person",
           "givenName": "Konstantinos",
           "familyName": "Evangelou",
           "email": "kevangel@admin.grnet.gr",
           "affiliation": {
              "@type": "Organization",
              "name": "GRNET S.A"
           }
        }
     ]
  }
```

What are the least data I should use

The codemeta file

All info is necessary

Create the metadata.json at least with:

The Software name

The Authors / Contributors

Discoverability / Citation

Additional Info

FAIRCORE4EOSC
Core Components Supporting a FAIR EOSC

faircore4eosc.eu

@FAIRCORE4EOSC

company/faircore4eosc

FAIRCORE4EOSC

Funded by
the European Union