



Universiteit
Leiden
The Netherlands

Opleiding Informatica

Setting up a FAIR Data Point
in LIACS

Hargurjit Singh (S1531727)

Supervisor:
Mirjam van Reisen
Special thanks to Kees Burger.

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

10/07/2020

Abstract

In this thesis we will see the difficulties of setting up a FAIR Data Point in LIACS within a relative short period of time. First of all, we will set up two softwares both published by the DTL FAIR Data team. The first one is called the FAIRifier, which is supposed to make raw data FAIR and to measure the FAIRness of data. The second software, the one on which the emphasis is in this article, is called the FAIR Data Point, abbreviated FDP. Secondly we will try to gain insight into the demand and acceptance of an FDP within the potential group of users. We will do so by conducting a survey among a select group of students.

Contents

1	Introduction	1
1.1	Involved organisations	2
1.2	Thesis overview	2
2	Background	3
2.1	FAIR Data Principles	4
2.2	What is a FDP exactly?	4
3	FAIR related software	6
3.1	FDP : First method	6
3.2	FDP : Second method	10
3.3	Editor	12
3.4	FAIRifier	13
4	Interest research	15
5	Conclusions and Further Research	18
5.1	Conclusion	18
5.2	Future work	18
	References	19
A	Appendix	21

1 Introduction

Academic research produces a lot of data. Especially in this digital era, we want to be sure that the optimal flow of (re)usable data is achieved. Roughly all data is digital nowadays. However, as simple as it seems, this data is often not findable, accessible, interoperable or reusable (FAIR) for other consumers of the same type of data. In this thesis we will call this unstructured or raw data and the main problem with this data is that it is not machine-readable.

Data can be of any type: geographical, scientific, financial, statistical, but also data about transport, nature or cultural data. This unsorted data can not be (re)used effectively by both machines and data consumers.

The currently implemented way of data stewardship in Europe and specifically in the Netherlands, often consists of unconnected data centres where the data is stored. We can see this implementation as a graph with unconnected nodes. The best case in this implementation is that every dataset is stored on the corresponding centre, since not every dataset gets published, and is retrievable from this centre, with its own GUI and access control/security layer. The user needs to get used to the respective interface since every data centre has their own. Logically this procedure is a lot of wasted time, inconvenient and costly.

This needed to be changed and so recently a data-movement arose to make this process more efficient [?]. This movement is named FAIR which is the abbreviation for Findable, Accessible, Interoperable and Reusable. These terms are further explained in the FAIR Data Principles. The FAIR Data principles act as an international guideline for high quality data stewardship [Mon18]. These principles were established in 2014, coincidentally also in Leiden, at a meeting about FAIR data [16]. We will elaborate on this in the next section of this thesis. FAIR Data - and the underlying linked data techniques are an addition to machine-readable harmonization and standardization of data. It makes information understandable for computing systems, and this is exactly the core and purpose of FAIR. Also for the users it is more efficient, since the users are not required to speak exactly the same language. The machine-readability is achieved through semantic modeling and the convergence between the two is achieved by making these semantic models interoperable.

This movement led to converting scientific data from raw to open and findable, a process called FAIRifying. There are tools that automatise the process to FAIRify, which have the obvious name of FAIRifier. We will see in this thesis that these tools are falsely named FAIRifiers, since they do not make data FAIR, but only function as a clean up/refine tool for the raw input data. However, the second piece of software called FDP solves this problem, which is the abbreviation of FAIR Data Point.

The name of this software gives it slightly away: the FDP is supposed to be a data entry point where the FAIRified metadata of the data can be uploaded. Information about the data is called metadata. We can see an FDP as a node on the data center-graph: the nodes are the FDPs, which are actually the data centres that are connected to an open distributed datacloud with one general Graphical User Interface [UH18], and the edges are generally speaking the open cloud itself because all nodes are connected with each other via this cloud.

1.1 Involved organisations

There are a few organizations that are worth mentioning in this thesis. These organizations are closely involved in the transition from the standard data processing method to a FAIR method. Some for example have created tools that stimulate a FAIR data environment and others are engaged in the formal part and legislation:

The EOSC (European Open Science Cloud) is an organisation funded by the European Commission to take care of this "FAIR" data movement in Europe. The EC announced the EOSC in April 2016. This commission is the executive of the European Union and promotes its general interest. The main aim of the EOSC is "to create a trusted environment for hosting and processing research data to support EU science in its global leading role" [MS16].

GO FAIR is a voluntary bottom-up initiative in Europe that was and still is closely involved in implementing the FAIR data principles (next section) in . A GO FAIR Implementation Network (INs) is "a consortium committed to defining and creating materials and tools as elements of the Internet of FAIR Data and Services (IFDS)" [GO-17], an organisation with similar goals as the European Open Science Cloud and facilitating its realisation. Implementation Networks are the core participating communities of the GO FAIR initiative.

DTLS (Dutch Techcentre for Life Sciences) actively promotes FAIR Data Stewardship of life science information in close collaboration with its international partners who also have a share in the FAIR data movement. DTLS has been (largely) responsible for the source code of the FDP and other FAIR related software.

1.2 Thesis overview

At the time of writing this chapter, there has been no prior research about setting up an FDP. This thesis will be the first publication about the process of putting on such a data point in Leiden Institute of Advanced Computer Science (LIACS). The aim of this study is therefore to set up such a FAIR Data Point for LIACS. The benefits of setting up a FDP are obvious: creating the possibility to make data FAIR. Researchers, students and teachers of LIACS will reap these benefits.

The aim of this study is therefore to investigate to what extent it is possible to set up a FAIR Data point in a given time period (March - June). Section 3 will contain the steps that need to be followed to successfully set up an FDP and some other related software. It goes without saying that setting up an FDP only makes sense if it will actually be put into use. Therefore we will also research the interest among the potential users by conducting a survey in section 4. In section 5 we will see the outcome of these research questions and any future work that remains unexamined.

2 Background

In this section, the necessary background information to better understand the FAIR movement and the principles of FAIR will be provided.

Good data management is essential in research. In this digital age search engines, online catalogs of books and references to online articles (like the one at the end of this thesis) optimize the findability of data. Formerly when academic data only was published on paper, these tools were not available. It is unimaginable to us how research was conducted at that time.

As is known, the current way of data storage in data warehouses is an outdated technique. FAIR defines working with a network of distributed nodes discoverable over the internet and operable through data visiting. Another point of attention is that data ownership is kept retained with the data producer and data subject. This conversion ensures an optimal and smooth flow of data. If data is stored correctly on the right location (node), the accessibility of this data of course increases. The researcher will comprehend where the desired data can be found, for exceptionally efficient data retrieving. Data retrieving becomes exceptionally efficient. The emphasis in the research can now be shifted from retrieving data to more beneficial tasks and this is an accomplishment by itself.

Furthermore, it can lead to the discovery of new elements in research. Not only finding desired data, data that you are initially looking for, but disclosure of new applicable data brings research to the next level.

Next to the benefits of making academic data more open, one should consider security as well. It plays an essential role in data stewardship. To date accountable people (called data stewards) were appointed to take the responsibility of properly managing data and everything about it. Even though we have made huge steps in terms of data management, there are still bottlenecks that we should investigate and resolve. In contrast to the manual way of working, we want to automate this process now and in the future: machines should be appointed to take care of this data stewardship. Thankfully we have invented tools that can find us all the information that we want, but these tools often are not optimized to retrieve FAIR and academic data.

Keeping this in mind, a large group of stakeholders including academia, funding agencies and researchers organised a workshop named Jointly Designing a Data Fairport [16]. This workshop was held in 2014 in Leiden and the stated parties agreed that the reuse and discovery of data should be examined.

This workshop was the first small step towards an immense change in the state of the art. Soon after this meeting the European Commission agreed to fund and set up the European Open Science Cloud (the EOSC). The name speaks for itself, but the so-called EOSC-vision makes things clearer : To give Europe a global lead in scientific data infrastructures and to ensure that European scientists reap the full benefits of data-driven science [MS16]. EOSC would later partner up with GO-FAIR, a Leiden based non-profit organisation, which is closely related to the Dutch Techcentre for Life Sciences (DTLS). The latter one is responsible for the available source code of the FAIR datapoint [FAI], which will be used for setting up such datapoint.

2.1 FAIR Data Principles

The FAIR guiding principles were established during this meeting. This list contains the conditions and properties of data to be called FAIR. These principles are listed below [JL18] [Mon18]:

To be Findable:

- F1. (meta)data are assigned a globally unique and persistent identifier
- F2. data are described with rich metadata (defined by R1 below)
- F3. metadata clearly and explicitly include the identifier of the data it describe
- F4. (meta)data are registered or indexed in a searchable resource

To be Accessible:

- A1. (meta)data are retrievable by their identifier using a standardized communications protocol
 - A1.1 the protocol is open, free, and universally implementable
 - A1.2 the protocol allows for an authentication and authorization procedure, where necessary
- A2. metadata are accessible, even when the data are no longer available

To be Interoperable:

- I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
- I2. (meta)data use vocabularies that follow FAIR principles
- I3. (meta)data include qualified references to other (meta)data

To be Reusable:

- R1. meta(data) are richly described with a plurality of accurate and relevant attributes
 - R1.1. (meta)data are released with a clear and accessible data usage license
 - R1.2. (meta)data are associated with detailed provenance
 - R1.3. (meta)data meet domain-relevant community standards

These principles define guidelines that the FAIR environment must adhere to. They relate to three components within the mentioned environment: the data, the metadata and the infrastructure. For those who have even a little superficial knowledge about data science, these principles speak for themselves. Nevertheless, they are all extensively discussed on the GO-FAIR website [GO-17].

2.2 What is a FDP exactly?

As the product scope of the FDP software states, FDP is a software that, from one side, allows data owners to expose datasets in a FAIR manner and, for another side, allows data users to discover properties about offered datasets (metadata) and, if license conditions allow, the actual data can be accessed. [FAI]. FAIR Data Points are hosted on a server, which can practically be anywhere, and serves as an entrypoint for FAIR data: FAIR data can be stored to the backing storage. To be precise, instead of data metadata will be stored on the FDPs. FDPs often do not contain the

actual data but instead they have references to where the data is stored.

The FDP naturally should meet the properties of being FAIR. One of the properties that particularly relates to the FDP, is that the metadata stored on an FDP should be interoperable: the language and syntax used should be generic and the characteristics of the stored metadata should correlate with those of previously published ones or, as I3 of the FAIR Guiding Principles states (meta)data include qualified references to other (meta)data [16].

As we will see in the next chapter, the FDP software is initially created to be hosted (deployed) on a web server called Apache Tomcat. Since we want to set up an FDP in LIACS, the FDP should be hosted on a server situated in LIACS. Once the FDP is up and running (i.e. accessible), users and data stewards should be able to upload metadata about their datasets and find the existing and available data.

FAIR has broad support in the world and also in the Netherlands. The benefits of findability, accessibility, interoperability and reusability are generally recognized.

Linking information for both research and operational situations can be simplified with this. FAIR Data can thus provide a major incentive for data interoperability and big data-like solutions. "Becoming a FAIR" is currently still a fairly technical operation in which the initial learning curve is fairly steep and user-friendly tooling is missing. The expectation is that such a set of instruments will become available. For example, Amazon will already offer linked data services in the short term. Those who go through the learning curve and have access to data that is already available in structured form can quickly make the switch to FAIR.

3 FAIR related software

In this chapter we will see the technical programming details of setting up an FDP and a FAIRifier. This documentation includes instructions to locally deploy the FDP on your own personal machine for example for demonstration or test purposes. However this same guide can be used to host a FDP on a server for a production deployment.

There are two ways of setting up a local FDP. Both are based on deploying the executable file, which gets generated by compiling the source code of the FDP. Until June 2020, the initial method was the only way to set up an FDP. This is also the method we will discuss first. Because frameworks have now been created to facilitate the deployment, several of the listed steps are conjugated into one. This is the second method we will see.

3.1 FDP : First method

The open source FDP software of DTL is online available on the github repository [FAI]. We will use a Ubuntu 16.04 system. Firstly we will set up a web-server called Tomcat, which will require Java to be installed. After that we will build and install the FDP software. This will generate a .war file which can be deployed on the running web-server. The FDP will be set up and hosted from the Ubuntu system that we installed the web-server on. Since we want to set up the FDP in LIACS, the same steps will be executed on a system linked to a server both located in LIACS. Note that it is possible that some links/software may be outdated by the time the readers get to read this thesis. Replacing these software with the updated ones will simply solve the problem.

Apache Tomcat (commonly called Tomcat), is an open source web-server developed by Apache Software Foundation. It is written primarily in Java and it is released on the Apache license 2.0. It is a cross platform application. The initial release of Apache Tomcat was done in 1999, and the latest version is Tomcat 9.0.20 released on 13-05-2019 [Fou].

Tomcat has it's dependencies on the Java Repository, so the first step to install Tomcat would be installing Java. Firstly, updating the 'apt' package in the terminal is necessary. Open the terminal window and type `sudo apt update`. It will probably take a few minutes. Once the package index is updated, the installation of the default Java OpenJDK package can begin: `sudo apt install default-jdk`. Also this step will take a few minutes. Verification of the installation can be done by checking the installed version, typing `java -version`. This command should return a statement like:

```
openJDK version "1.8.0_212"  
openJDK Runtime Environment (build 1.8.0_212-8u212-b03-0ubuntu1.16.04.1-b03)  
OpenJDK 64-Bit Server VM (build 25.212-b03, mixed mode)
```

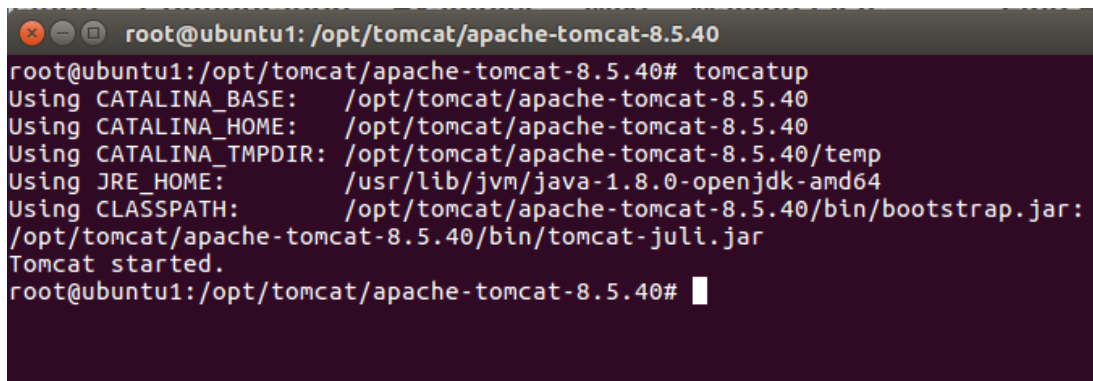
The second step is to install Tomcat itself. First the zip file from the website <https://tomcat.apache.org/download90.cgi> should be downloaded. The latest version is, as said above, 9.0.20. Navigating to the Downloads folder will show the zip file. The following step is to make a 'tomcat' directory in the /opt

folder, such that the zip file can be extracted. This should be done by typing `mkdir /opt/tomcat`. Now the downloaded zip file should be extracted and moved into the made directory, by entering the command `mv WidgetList.zip apache-tomcat-9.0.20.zip /opt/tomcat/`. The zip should be extracted in the directory. Since the directory is saved in a restricted part of the system, it requires to be logged in as "super user" to make changes in the directory. To become a super user, the command `sudo -i`, followed by the password of the user should be executed. The terminal allows now to navigate through and to make changes in the directories.

The next step is to unzip the zip file. This is done by executing `unzip apache-tomcat-9.0.20.zip`. Now, to 700 `/opt/tomcat/apache-tomcat-9.0.20/bin/*sh`. The next steps are to create symbolic links for the startup and shutdown scripts:

```
ln -s /opt/tomcat/apache-tomcat-9.0.20/bin/startup.sh /usr/bin/tomcatup
ln -s /opt/tomcat/apache-tomcat-9.0.20/bin/shutdown.sh /usr/bin/tomcatdown
```

After running these commands it became possible to startup and shutdown tomcat from anywhere. So the user does not have to go to the tomcat directory in the `/opt` folder anymore. To make sure the link is made correctly, the `tomcatup` command can be executed. To end the hosting, we simply execute the `tomcatdown` command. The response of the first mentioned command should be something like Figure 1.

A terminal window titled 'root@ubuntu1: /opt/tomcat/apache-tomcat-8.5.40' showing the execution of the 'tomcatup' command. The output lists the configuration variables used: CATALINA_BASE, CATALINA_HOME, CATALINA_TMPDIR, JRE_HOME, and CLASSPATH. It then states 'Tomcat started.' and returns to the prompt.

```
root@ubuntu1: /opt/tomcat/apache-tomcat-8.5.40# tomcatup
Using CATALINA_BASE:   /opt/tomcat/apache-tomcat-8.5.40
Using CATALINA_HOME:   /opt/tomcat/apache-tomcat-8.5.40
Using CATALINA_TMPDIR: /opt/tomcat/apache-tomcat-8.5.40/temp
Using JRE_HOME:        /usr/lib/jvm/java-1.8.0-openjdk-amd64
Using CLASSPATH:       /opt/tomcat/apache-tomcat-8.5.40/bin/bootstrap.jar:
/opt/tomcat/apache-tomcat-8.5.40/bin/tomcat-juli.jar
Tomcat started.
root@ubuntu1: /opt/tomcat/apache-tomcat-8.5.40#
```

Figure 1: Response after executing the 'up' command

The User Interface of the Tomcat Server can be viewed by opening the local host in a web browser. The site <http://127.0.0.1:8080/> will open the GUI of Tomcat, that should look like Figure 2

The Tomcat Server is installed. Now the FDP source code needs to be compiled. First the directory needs to be downloaded/cloned from the github repository and entered in the terminal:

```
\verb|git clone https://github.com/FAIRDataTeam/FAIRDataPoint
\verb|cd FAIRDataPoint
```

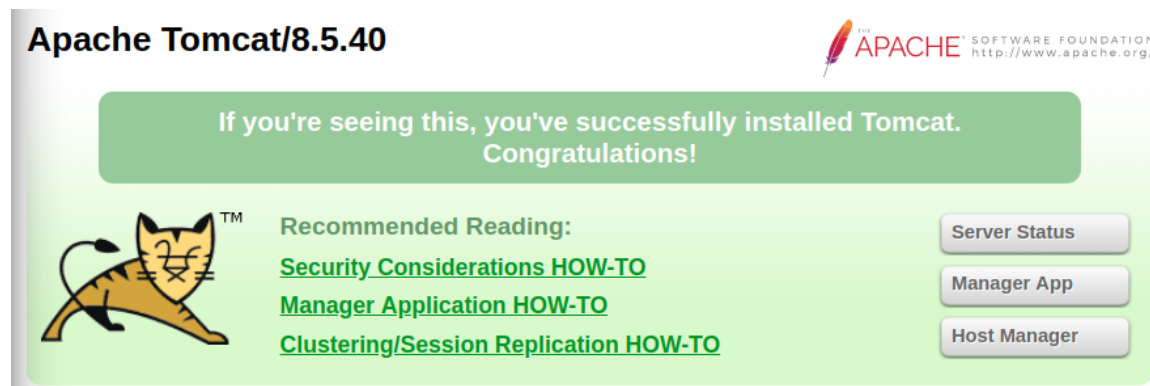


Figure 2: GUI of the Tomcat server

Now the directory is downloaded, the next step is to install the FAIRDataPoint. This will give a .war file which can be deployed in the Manager App of the Tomcat Webserver. A WAR (Web Application Resource) is a file type in which resources for a web application can be stored. In this case, the resources to create an FDP will be stored in fdp.war

The same software company (Apache Software Foundation) that released the webserver (Tomcat) where the FDP software is based on, also has brought out a software project management and comprehension tool named Maven. The FDP software needs to be installed by using this tool as follows:

```
mvn install
```

This will take a few minutes to install. The installation process now has created a **target** map. One of the files in this map is the file named fdp.war—.

Now the next step is to deploy this file in the Manager app of the Tomcat Server. First we need to create a user for the GUI of Tomcat, such that the Manager app is accessible. We have to add a username and password to the tomcat-users.xml file located in the /conf folder:
`gedit conf/tomcat-users.xml`

we now see a block of commented text at the bottom of the file where users are created. We need to add / uncomment the following lines (like Figure 3):

```
<role rolename = "manager-gui" />
<user username = "tomcat" password = "tomcat" roles = "manager-gui" />
```

By clicking on the Manager App-button as showed in figure... , the GUI will ask to fill in the username (tomcat) and password (also tomcat), which we just created. This will lead to the Application Manager. fdp.war file can be deployed at the Deploy section as seen in Figure 4.

It is obvious that we now have to click Choose File and upload and deploy it. After deploying the file, the /fdp path should be visible in the application section of the Application Manager. The

```

<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<role rolename="manager-gui"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
</tomcat-users>

```

Figure 3: Creating a user for the FDP

/manager	none specified	Tomcat Manager Application	true	1	Expire sessions with idle ≥ 30 minutes
----------	----------------	----------------------------	------	---	--

Deploy

Deploy directory or WAR file located on server

Context Path (required):
XML Configuration file URL:
WAR or Directory URL:

WAR file to deploy

Select WAR file to upload No file chosen

Configuration

Figure 4: Deploying the fdp.war file

FDP is now up and running. To view the GUI of the FDP, we go to the link <http://127.0.0.1:8080/fdp/swagger-ui.html#/>. The result should look like Figure 5. We see a few buttons with the text post. We will need this to post our metadata. The other options help us retrieving data from the FDP. The FDP requires us to upload the metadata in RTF format. To do so, we will need the editor tool which we will discuss later in this chapter 3.3. The result should look like Figure 5.

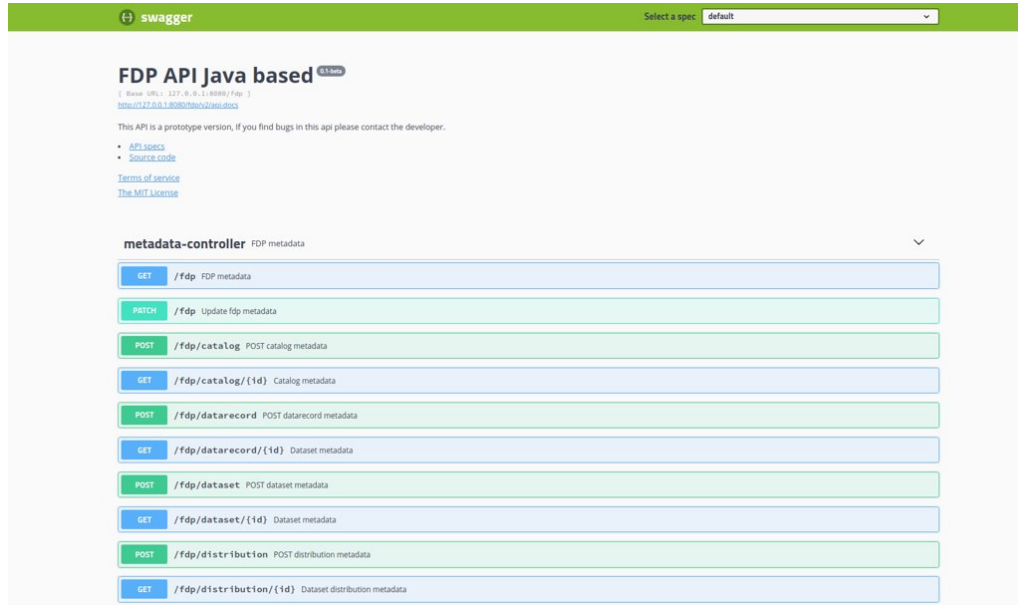


Figure 5: GUI of the FDP

3.2 FDP : Second method

Taking users from other operating systems into account, we will perform these steps on a Windows machine. With this second (less complex) method we do not use Tomcat. To facilitate the deployment of our reference implementation of the FDP we distribute it as a Docker image [Doc20]. This requires us to install the Docker runtime environment on our machine first. We can download and install this from <https://download.docker.com/win/stable/Docker%20Desktop%20Installer.exe> [Doc20]. To check whether Docker has been installed, we open a terminal window and execute the command `docker`. We should see the user manual of Docker like Figure 6:

An example of a docker compose configuration is given on the deployment manual website of the FDP [Doc20]. This configuration contains the commands to deploy the FDP server, the FDP client and also a command relating to MongoDB, a document database that we use as underlying storage [12]. These three parts of the commands will ensure that the correct files are downloaded and built for setting up the FDP. The commands are listed in Figure 7.

```

C:\Users\Admin>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\\Users\\Admin\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify

```

Figure 6: Status of Docker

```

# docker-compose.yml

version: '3'
services:
  fdp:
    image: fairdata/fairdatapoint:1.6.0
  fdp-client:
    image: fairdata/fairdatapoint-client:1.6.0
    ports:
      - 80:80
    environment:
      - FDP_HOST=fdp
  mongo:
    image: mongo:4.0.12

```

Figure 7: Docker Compose Configuration. This code needs to be saved as an 'yaml' file into the folder we want install the FDP in.

We copy and paste the code into a file and save it as `docker-compose.yml` on our machine. The location of the file must be determined by the user himself. We navigate to that chosen folder using the command interpreter and execute `docker-compose up -d`. The given aforementioned three images will be downloaded from the Docker Hub repository. Once the script is finished downloading and deploying the FDP, we can check whether the installation is successful and the images are indeed executing. We do so by starting the Docker Desktop Dashboard, and check if the FDP images are running.

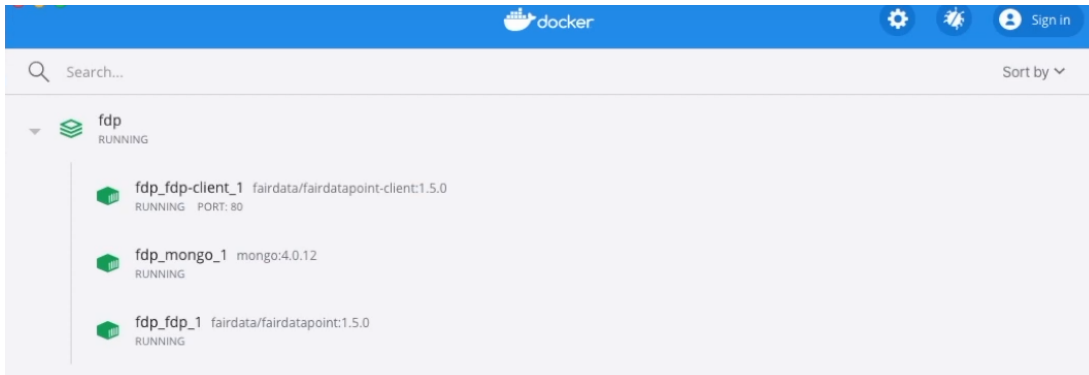


Figure 8: Docker Desktop, the GUI of Docker. Check whether the FDP source code is downloaded and installed correctly from the repository.

Once we have checked this, we want to know if our webclient is working. This is easily done by navigating to the url `localhost` in any web browser. If everything is working fine, we will see a webpage with the default metadata of the FDP, just like in the previous section of this chapter. As given on the GitHub repository, the combination of username `albert.einstein@example.com` and password `password` will grant you access to the FDP.

3.3 Editor

To upload metadata to our newly deployed FDP we can either use the built-in editor of the FDP-API deployed with the second method (called the Metadata Provider) or use the online editor if we set up the FDP using Tomcat. The editor can be found on [`https://editor.fair-dtls.surf-hosted.nl/#!/`](https://editor.fair-dtls.surf-hosted.nl/#!/) [LOBdSS20]. The editor has an documentation [DTL16] that we will use to generate the code to insert metadata in the FDP.

There are four possible layers of metadata that can be added to the FDP : Catalogs, datasets, distributions and repositories. All of these layers contain pointers to the level beneath. This means that every FDP has at least one entry of a catalog, a catalog has at least one pointer to a dataset and so on. The first thing that comes to mind of an attentive reader who has some knowledge of graphs is that this will form a tree structure. This is actually not true as it is possible that the same repository may exist in multiple datasets.

The repositories contain the actual resources of the data, which is why resources sometimes are referred to as the fifth layer of metadata. Generally we do not want to upload the actual data to the FDP. The location of the data must be known, which is exactly the purpose of the fifth layer. The editor offers the possibility to add an extensive number of types of metadata in all four layers, including title, description, language and much more. The figure below shows us how the editor looks like Figure 9.

The screenshot shows the FDP Metadata editor interface. At the top, there are tabs for 'Repository', 'Catalog', 'Dataset', and 'Distribution'. The 'Catalog' tab is selected. To the right of these tabs are 'Build' and 'Share' buttons. Below the tabs, there is a list of metadata fields with corresponding input boxes: Title, Has version, Publisher, Publisher Name, Description, Language, License, Rights, Homepage, and Theme taxonomy. Some fields have a small blue icon next to them. On the right side, there is an 'RDF preview' section showing the generated DCAT format code.

Figure 9: FDP Metadata editor

The syntax in which the editor outputs the generated code is of DCAT format, which is an RDF (Resource Description Framework) data model defined by W3C [Bro20] [Gra14]. This data model is broadly similar to other modeling approaches such as class diagrams which follow a given set of rules like the UML syntax. As we can see in figure the RDF preview on the right is in DCAT format. Fortunately we can use the built-in editor of the FDP to upload the metadata. In that case we do not even get to see the RTF code.

3.4 FAIRifier

The last software that we will look into in this thesis is the so-called FAIRifier. Just like the FDP, this software has also been developed by the FAIRDataTeam. The name gives the functionality slightly away. As the description on the Github repository of the FAIRifier states: The FAIRifier is a tool to make messy data FAIR. [FAI]. It was built as an extension of the free open source data tool called OpenRefine, formerly known as Google Refine [Dav].

What this extension initially added to the refine tool is that it offered the possibility to download the refined (FAIRified) (meta)data in RDF format so that it could be directly uploaded to an FDP. This reminds us of the online editor that we saw previously. Later the FAIRifier was updated such that the FAIR data could be uploaded directly from the FAIRifier to an FDP, without the RDF

intermediate step [FAI].

The screenshot shows the FAIRifier web application. At the top, it says 'gonl SV r5 truncated plus vcf' and 'Permalink'. Below this is a 'Facet / Filter' sidebar with 'Undo / Redo 17' and a section titled 'Using facets and filters' with instructions. The main area displays a table with 64 rows. The table has columns: #CHROM, POS, ID, and REF. The first few rows are visible, showing chromosome information and genomic coordinates. On the right side, there is an 'Export project' dropdown menu with options like 'Tab-separated value', 'Comma-separated value', 'HTML table', 'Excel (.xls)', 'Excel 2007+ (.xlsx)', 'ODF spreadsheet', 'Triple loader', 'MQLWrite', 'Custom tabular exporter...', 'Templating...', 'RDF as RDF/XML', 'RDF as Turtle', and 'POST to FAIR Data Point'.

Figure 10: Every thesis should have figures. Source: www.marxbrothers.org.

As we can see in Figure 10, the FAIRifier is a tool that is primarily intended as a manual cleaning tool, to FAIRify data of different types. But because FAIR's main purpose is to make data machine readable, the name of this software is not entirely correct. The FAIRifier does not bring us any steps closer to automating the extraction, cleaning or machine-readability of data.

To install the FAIRifier, the OpenRefine tool must therefore be installed first. Instructions for this can be found on its Github repository, which is simply cloning the repository to the computer and then executing the command `./refine` [FAI]. Now to install the FAIRifier, or the OpenRefine-metadata-extension as it is now called, we download the zip file from the repository of the extension [FAI]. Now we need to unzip this in the extensions folder of the refine directory using the following command :

```
unzip metadata-X.Y.Z-OpenRefine-3.3.zip path/to/openrefine-3.3/webapp/extensions
```

The use of the tool is almost self-explanatory. We upload data, and the tool gives us suggestions for making adjustments.

4 Interest research

In the previous chapter we have seen how to set up an FDP. Setting up the FDP is of course only useful if it is of added value; if noone is going to make use of the FDP, than it has no added value to the faculty. We would like to map this demand. After all, setting up an FDP only really makes sense if there is sufficient acceptance, and the degree of potential participation is also high. We will do this by trying to measure the acceptance of the FDP by the potential users. This gives us insight into the demand for an FDP.

The conducted survey will both follow characteristics of a qualitative and quantitative research. Since we want to examine the will and ability of the potential users to adapt the new concept of FAIR, it is therefore a emergent design type of a qualitative research. On the other hand, we want to by applying statistics This corresponds to descriptive research design type; we do not have any assumption in mind but we are curious about the outcome.

We let a select group of university students fill in a survey about the FDP. We consider this group as representatives and sample of the potential future users of the FDP in LIACS. We chose a well educated group of readers; thirty-eight bachelor or master students, all of them studying in the Netherlands. This was exactly the inclusion and exclusion criteria. The surveyed students are between 19 and 30 years old and were contacted through email. By opting for this specific and small group, we try to reduce the chance of the miscommunication mentioned below. We did not hold a pilot for this research.

A short brief summary of what FAIR and FDPs are and do is given at the beginning of the survey [Sin] A. Based on this introduction the surveyed students will form an opinion about FAIR and setting up an FDP. We will capture this first glance through the four questions below :

1. **Would an FDP be useful e.g. in LIACS? Yes/no, Please explain why.**
2. **Which characteristic of FAIR appeals the most to you? Please explain why.**
3. **Would you share your (meta)data on an FDP?**
4. **Would you look for and extract data from an FDP?**

On the one hand, this gives the opportunity to get to know different opinions and interests from the potential user side. And, as previously stated, this is exactly the purpose of this survey. On the other hand however it is possible that this short summary for some readers may provide some distorted views of what FAIR and FDPs actually are, we should keep in mind that answers to the survey are based on the image that the questioned readers get from the summary.

The survey was held online using Google Forms [Sin], and was available for a period of approximately two weeks. The four questions that are asked ensure that an impression of the acceptance and participation level of the students is exposed. In terms of statistics we assume that this sample will represent the students of LIACS. This is ultimately the target group we want to reach, and we want to estimate their usage in the future. As earlier stated, the majority of the students is from the Netherlands. However, most of them do not study at LIACS. In order to get useful opinions from this large group, the survey is based on the situation that an FDP will be set up in their

faculty. As we have seen in the previous chapter, the only difference would be the location of the server where the FDP is hosted.

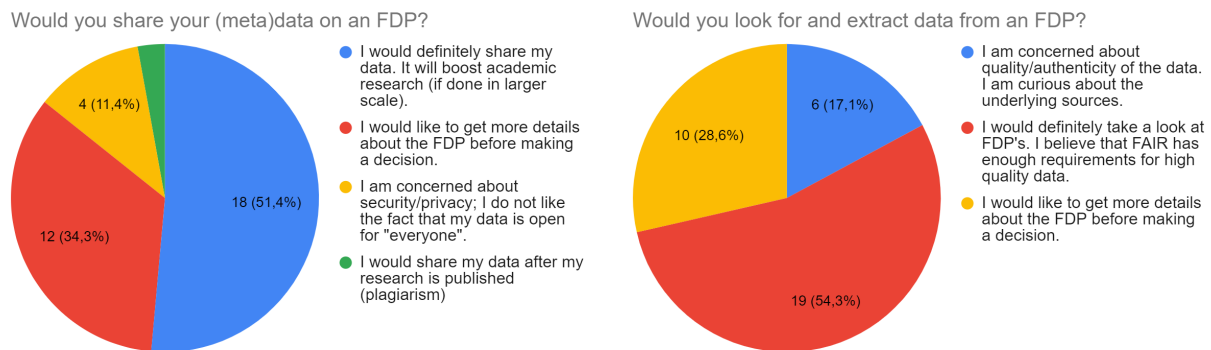


Figure 11:

Pie charts of the last three questions are shown in Figures 11 and 12 and we will discuss the first question later. About half of the respondents state that they would share their data on a FDP. About a third of the students said that they need more information about the FDP to form a statement. The rest, about one-sixth, is concerned with security and privacy, and wants to get more information about FDPs. None of the respondents said that they definitely would not use the FDP. We also see roughly the same result when asked if the students would retrieve data from the FDP.

Based on these results, we can cautiously conclude that chances are high that an FDP will indeed be accepted by the user audience, despite the fact that a large part indicates that it does not want to make a decision before getting more information. Also a significant number of students is concerned about the security and the related quality of the data. This indicates that users will be aware of the fact that there should be a sense of responsibility with regard to data.

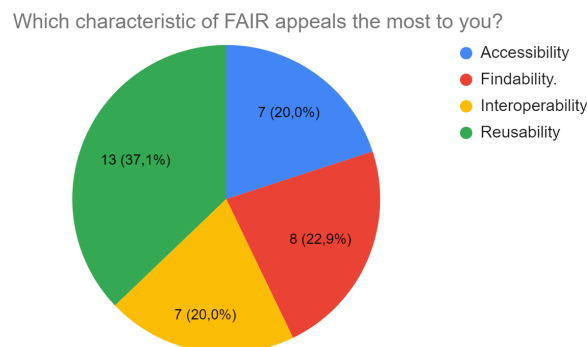


Figure 12:

The last question, unlike the rest of the questions, was answered very differently compared to the first two. This question was only intended to investigate interest in the various aspects of FAIR. The majority of the surveyed students indicate that the reusability of data appeals to them the most. So the fact that the users will be able to use already generated data and also will be able to upload their own data such that it can be reused, is appealing to them.

The first question was answered positively by almost everyone. except for an exception who answered: No, I believe working with raw data and adjusting it for your own goals is the optimal way of understanding a dataset thoroughly.. The rest of the students indicate that they would use an FDP.

These results imply that an FDP would be well received by the users. Based on these answers, we may conclude that an FDP will be accepted.

5 Conclusions and Further Research

5.1 Conclusion

The main goal of this thesis, to set up an FDP in a relatively short period, has been achieved. The last step that needs to be taken is performing these exact steps on a computer in LIACS computer connected to a server. Unfortunately we did not get to this due to the lockdown. We have even seen several other software that matter too. We have seen two ways of hosting an FDP. The first one requires some programming experience and knowledge, and retains the ability to keep different components of the FDP separated; the FAIRifier, the editor and the FDP itself to be precisely. However, the second method is more plain and delivers a far more compact product. This method keeps the frontend and the backend completely separated. As we have seen, it has a built-in editor, such that we do not need to work with any RTF format. It also requires less prior knowledge about programming. Our second research question also has a positive outcome. We have seen that the potential future users are positive about the ideas and pursuit of FAIR. This implies that the FAIR Data Principles will be respected and we conclude that the use of an FDP would be accepted.

5.2 Future work

Although this thesis covers and solves the stated research questions and problems, there still are many possibilities that remain unexplored. Due to time constraints it was impossible to inspect, test or implement the following thoughts and additions :

- As several times mentioned in this thesis, FAIR strives to make data machine-readable. In this thesis the vast majority of the research was done manually. As a matter of fact it is understandable that the FDP has to be set up by hand. But it would be practical and logical if the FDP could automatically upload metadata, instead of having to specify what type of data we are dealing with. This holds for both the methods that we used.
- The same goes for the FAIRifier. As the same says, we expect that we input raw data, and it gives us the FAIRified data back. But instead it now still is an manual tool to clean up data by hand. This could be improved to a software based on Artificial Intelligence that exactly does what the name says, FAIRify data automatically.
- Just like improvement in the second method, which implemented the editor in the FDP itself, it would be convenient if the FAIRifier could be combined as well.
- Last but not least, we were not able to host the FDP phisically in LIACS. This point can be neglected, since this thesis is the exact recipe to fulfill this addition.

References

- [12] What is MongoDB? , <https://www.mongodb.com/what-is-mongodb>.
- [Bro20] David Browning. DCAT 2 vocabulary. 2020. , <https://www.w3.org/ns/dcat>.
- [Dav] David Huynh . OpenRefine Documentation. , <https://openrefine.org/documentation.html>.
- [Doc20] Docker. Desktop engine overview. 2020. , <https://www.docker.com/products/docker-desktop>.
- [DTL16] DTLS. FDP / metadata editor / search engine walkthrough. 2016. , <https://docs.google.com/document/d/1eBJKSg1u5gep6-2cbjGpWNulMvC7F9BSmnKYq7FFSGA/edit#heading=h.ah8yn51oup91>.
- [16] Wilkinson *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3, 2016. , <https://doi.org/10.1038/sdata.2016.18>.
- [FAI] FAIRDataTeam. FAIR Data Point design specification. *GitHub Repository*. , <https://github.com/FAIRDataTeam/>.
- [Fou] The Apache Tomcat Foundation. Tomcat 8 software downloads. , <https://tomcat.apache.org/download-80.cgi>.
- [GO-17] GO-FAIR. FAIR Principles (detailed). 2017. , <https://www.go-fair.org/fair-principles/>.
- [Gra14] Graham Klyne, Jeremy J. Carroll, Brian McBride. RDF 1.1 Concepts and Abstract Syntax. 2014. , <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
- [JL18] Ying Zhang Wenjuan Cui Zhihui Du Jianhui Li, Xiafeng Meng. Big scientific data management (FAIR Data Principles). pages 296–303, 2018. ISBN 978-3-030-28060-4.
- [LOBdSS20] FAIR Data Team Luiz Olavo Bonino da Silva Santos, Kees Burger. FAIR Metadata Editor. 2020. , <https://editor.fair-dtls.surf-hosted.nl/#!/>.
- [Mon18] Barend Mons. Data stewardship for open science. 2018. ISBN 978-1-4897-5317-3.
- [MS16] Jean-Claude Burgelman Michel Schouppe. Relevance of the eossc initiative and fair principles in the realm of open science and implementation phases of the eossc. *Scientific Data*, 2016. , https://ec.europa.eu/research/openscience/pdf/eossc-fair_paper_schouppe-burgelman_2018.pdf.
- [Sin] Hargurjit Singh. FAIR Data survey. , https://docs.google.com/forms/d/e/1FAIpQLSc0sXtZOG-DFY5B5ZpTNpoQTVWdx4-0i51jtuT_mgl78piHsg/viewform.
- [UH18] *et al.* Ursula Hubner, Antonia Zapf. German medical data sciences: A learning health-care system). pages 209–213, 2018. , ISBN 978-1-61499-895-2.

A Appendix

7/23/2020

FAIR Data

FAIR Data

Academic research produces a lot of data. However, there is no generally accepted technique of storing and retrieving these data. The so-called "FAIR data movement" arose in 2014, which ensured that certain guidelines related to saving and retrieving data from databases/server/data lakes were established.

FAIR is the abbreviation of Findable, Accessible, Interoperable and Reusable. These terms (obviously) relate to type of data and the way of storing and retrieving it.

Findable: To make (meta)data findable through an interface, for people as well as AI's.

Accessible: To make (meta)data accessible, i.e. the possibility to look into the data.

Interoperable: To make databases interoperable such that a network of data-storing points is generated.

Reusable: To ensure that data is reusable, which will boost academic research and save time and money.

The currently-implemented way of storing data is often raw, unsorted data stored in databases, without a user interface. A FAIR Data Point (FDP) is an "entry point" of FAIR Data: the improved substitute of the old databases, which will contain FAIR metadata (information about the data, for example the column headers/names of a spreadsheet). Furthermore the FDP's will be interoperable with each other: if every faculty or university decides to set up an FDP, it will generate a network/graph where the nodes represent the FDP's. All FDP's are interoperable by definition.

The situation that we get is entry points (initially at the universities all over Europe) where FAIR data can be uploaded to and retrieved from. We naturally have to take into account aspects like security, granting access, etc. This will not only increase the scale of reusability, but since this process will generate one single network, it will be easy for AI's to discover different data patterns.

The aim in this thesis is to set up such an FDP (in the Leiden Institute of Advanced Computer Science (LIACS), my faculty). This can only be successful when potential users have trust and affection for such a new concept. That is exactly the reason for this survey.

Please answer these questions while keeping in mind that you (already have or also) will have to work with data that you have to search and clean!

***Vereist**

1. Would an FDP be useful e.g. in LIACS? Yes/no, Please explain why. *

2. Which characteristic of FAIR appeals the most to you? Please explain why. *

3. Would you share your (meta)data on an FDP? *

Markeer slechts één ovaal.

- ☐ I would definitely share my data. It will boost academic research (if done in larger scale).
- ☐ I would like to get more details about the FDP before making a decision.
- ☐ I am concerned about security/privacy; I do not like the fact that my data is open for "everyone".
- ☐ I would not share my data. My data is private.
- ☐ Anders: _____

4. Would you look for and extract data from an FDP? *

Markeer slechts één ovaal.

- ☐ I would definitely take a look at FDP's. I believe that FAIR has enough requirements for high quality data.
- ☐ I would like to get more details about the FDP before making a decision.
- ☐ I am concerned about quality/authenticity of the data. I am curious about the underlying sources.
- ☐ I would definitely not trust an FDP. I want to generate or find my own data.
- ☐ Anders: _____

Deze content is niet gemaakt of goedgekeurd door Google.

Google Formulieren