Implementing support for managing DHIS2 large scale deployments

Institution: Barcelona School of Informatics (FIB)

Studies: Bachelor’s degree in Informatics Engineering

Speciality: Software Engineering

Year: 2018-19 Q2

Student: Alexis Rico Carreto (alexisrico@essi.upc.edu)

Director: Petar Jovanovic (petar@essi.upc.edu)

Ponent: Alberto Abelló Gamazo (aabello@essi.upc.edu)

Abstract

This is the report for the thesis “Implementing support for managing DHIS2 large scale deployments”. The thesis consists on the creation of three applications that will be used by the IT team that provides support to the department “Neglected Tropical Diseases” of the “World Health Organization” (WHO).

This project is backed up by the “Database Technologies and Information Management Group” and the “Department of Service and Information System Engineering” of the “Polytechnic University of Catalonia”. We have also been in collaboration with the “World Health Organization”, the “WHO Information System to Control and Eliminate Neglected tropical diseases” and the “WHO Integrated Data Platform”.

Esta es la memoria para el trabajo “Implementing support for managing DHIS2 large scale deployments”. El trabajo final de grado tiene como base la creación de tres aplicaciones que serán usadas por el equipo de IT que proporciona soporte al departamento de “Enfermedades Tropicales Minoritarias” de la “Organización Mundial de la Salud” (OMS).

Este proyecto está respaldado por el “Grupo de Tecnologías de Bases de Datos y Gestión de la Información” y el “Departamento de Ingeniería de Servicios y Sistemas de Información” de la “Universidad Politécnica de Cataluña”. Además hemos estado colaborando con la “Organización Mundial de la Salud”, el “Sistema de Información de la OMS para controlar y eliminar enfermedades tropicales minoritaria” y la “Plataforma integrada de datos de la OMS”.

Aquesta és la memòria pel treball "Implementing support for managing DHIS2 large scale deployments". El treball de fi de grau té com a base la creació de tres aplicacions que seran usades per l'equip d'IT que proporciona suport al departament de "Malalties Tropicals Minoritàries" de l'Organització Mundial de la Salut (OMS).

Aquest projecte està recolzat pel "Grup de Tecnologies de Bases de Dades i Gestió de la Informació" i el "Departament d'Enginyeria de Serveis i Sistemes d'Informació" de la "Universitat Politècnica de Catalunya". A més hem estat col·laborant amb l'Organització Mundial de la Salut, el "Sistema d'Informació de l'OMS per controlar i eliminar malalties tropicals minoritàries" i la "Plataforma integrada de dades de l'OMS".

Glossary

WHO: Acronym for World Health Organization, a specialized agency of the United Nations that is concerned with international public health.

NTD: Acronym for Neglected Tropical Diseases, a department of WHO that studies 21 different diseases with diverse, multidimensional natures, that affect low income and rural areas around the world.

UPC: Acronym for Polytechnic University of Catalonia, the largest engineering university in Catalonia, Spain.

ESSI: Acronym for Department of Service and Information System Engineering, an interdisciplinary unit of UPC that focuses on the common ground between service science, management, engineering, information systems and technology.

WISCENTD: Acronym for WHO Information System to Control/Eliminate Neglected Tropical Diseases, a project of the NTD department at WHO focused in innovating the research of data management.

WIDP: Acronym for WHO Integrated Data Platform, a project at WHO focused in creating a global data platform.

DHIS2: Acronym for District Health Information System 2, an open source software platform for reporting, analysis and dissemination of data for health programs.

HISP: Acronym for Health Information Systems Programme, a global network of people, entities and organisations that design, implement and sustain Health Information Systems.

UiO: Acronym for University of Oslo, the oldest university in Norway, located in the Norwegian capital of Oslo.

Metadata: A set of data that describes and gives information about other data. In DHIS2 we understand it as the internal schema for the data stored in the database.

Contents

1. Introduction **8**

2. Context **9**

2.1. Areas of interest 9

2.2. Stakeholders 10

2.2.1. The World Health Organization 10

2.2.2. Polytechnic University of Catalonia 10

2.2.3. Affected population 10

2.2.4. External DHIS2 users and developers 11

2.2.5. DHIS2 core developers (University of Oslo) 11

3. Problem formulation **12**

3.1. Packaging of bundled metadata 13

3.2. Bulk importing of data 14

3.3. Version tracking of metadata changes 15

4. Specific objectives of the thesis **16**

5. State of the art **17**

5.1. Existing applications and research 17

5.2. Available technologies 18

5.3. Helper libraries 18

5.4. DHIS2 community 19

5.5. DHIS2 core developers 19

6. Initial project planning **20**

6.1. Scope of the project 20

6.2. Potential obstacles 20

6.2.1. Fixed timing 20

6.2.2. New requirements 21

6.2.3. Technical issues 21

6.2.4. Research and innovation 21

6.3. Methodology and rigor 22

6.3.1. Overview 22

6.3.2. Development tools 22

6.3.3. Development cycles 23

6.3.4. Integration tests 23

6.4. Monitoring tools and validation 24

6.5. Development phases 25

6.5.1. Initial Planning and Scope 25
6.5.2. Viability analysis 25
6.5.3. Application development 25
6.5.4. Thesis documentation 25
6.6. Temporal planning 26
6.6.1. Task planning 27
6.6.2. Gantt diagram 28
6.6.2. Action Plan 29
6.7. Resources and budget 30
6.7.1. Hardware resources 30
6.7.2. Software resources 31
6.7.3. Human resources 32
6.7.4. Budget monitoring 33
6.7.5. Unexpected and contingency costs 33
6.7.6. Cost distribution across development tasks 34
6.7.7. Total estimated budget 35

7. Sustainability report 36
7.1. Sustainability matrix 36
7.2. Social sustainability 36
7.3. Environmental sustainability 37
7.4. Economic sustainability 37

8. Design and implementation 38
8.1. Advanced Export App 38
8.1.1. Main objectives 38
8.1.2. Technical overview 38
8.1.3. Application evolution 40
8.1.4. Conclusions 44
8.2. Bulk Load App 45
8.2.1. Original application 45
8.2.2. Main objectives 46
8.2.3. Technical overview 46
8.2.4. Application features 49
8.2.5. Conclusions 51
8.3. Metadata Repository Script 52
8.3.1. Original scripts 52
8.3.2. Main objectives 52
8.3.3. Technical overview 53
8.3.4. Conclusions 54

8.4. Local development environment 55
8.5. Application testing and requirement acceptance 56
8.5.1. Development team 56
8.5.2. Field IT personnel 56
8.5.3. WHO Program managers 56
8.5.4. External DHIS2 users 56

9. Conclusions 57
9.1. Overview 57
9.2. Objective fulfillment 58
9.3. Planning deviations 59
9.4. Future work 60
9.4.1. Synchronization wizard in the Advanced Export App 60
9.4.2. Dependency blacklisting in the Advanced Export App 60
9.4.3. Tracker programs in the Bulk Load App 61
9.4.4. Advanced validation in the Bulk Load App 61
9.4.5. Slave Synchronization in the Metadata Repository Script 61

References 62

Table of figures

Figure 1. DHIS2 Web interface 9
Figure 2. Different instances used by the WHO for the global data platform 12
Figure 3. DHIS2 Official web application to import metadata packages 13
Figure 4. DHIS2 Official web application to fill data forms 14
Figure 5. Initial objectives of the thesis 16
Figure 6. DHIS2 core development decisions made in May 2019 19
Figure 7. Redmine dashboard used by WHO and UPC 24
Figure 8. Estimated time dedication table 26
Figure 9. Initial task planning 27
Figure 10. Gantt diagram 28
Figure 11. Action Plan Stages 29
Figure 12. Hardware resources budget 30
Figure 13. Software resources budget 31
Figure 14. Human resources budget 32
Figure 15. Cost distribution table with task description 34
Figure 16. Total estimated budget 35
Figure 17. Sustainability matrix 36
Figure 18. Advanced Export App's Extractor public methods 39
Figure 19. Advanced Export App's Extractor library dependencies 39
Figure 20. Advanced Export App v0.1.0 40
Figure 21. Advanced Export App v0.2.0 41
Figure 22. Advanced Export App v0.2.1 41
Figure 23. Advanced Export App v0.2.2 42
Figure 24. Advanced Export App showcasing the "Options menu" 43
Figure 25. Advanced Export App showcasing the "Admin menu" 43
Figure 26. Original Bulk Load App developed in 2016 45
Figure 27. Bulk Load App methods for template creation 47
Figure 28. Bulk Load App methods for spreadsheet parsing 48
Figure 29. Bulk Load App methods for back-end connection 48
Figure 30. Bulk Load Application web front-end 49
Figure 31. Example data entry sheet for a dataset with disaggregation 50
Figure 32. Example legend sheet for a program with field validation 50
Figure 33. Example metadata sheet for a dataset 50
Figure 34. Usage help of the Metadata Repository Script 53
Figure 35. Configuration options of the Metadata Repository Script 53
Figure 36. Remote GitHub repository pushed by the Metadata Repository Script 54
Figure 37. DHIS2 System Settings to whitelist CORS domains 55
Figure 38. Final objectives of the thesis 58
Figure 39. Final task planning 59
Figure 40. Advanced Export App's new Wizard mock-up 60

This page is intentionally empty

1. Introduction

Neglected Tropical Diseases (NTD) is a department at the World Health Organization (WHO) that studies 21 different diseases with diverse, multidimensional natures, that affect low income and rural areas around the world.

These tropical diseases affect millions of people around the world but it is especially important in countries of Africa, South America and Central America.

As an example, one of these neglected diseases is Chagas disease, caused by Trypanosoma cruzi parasite, endemic of rural countries, but present worldwide and currently estimated to affect more than 7 million people.

Detection of new cases, transmission routes and follow-up treatment rely on collaborations with official sources, remote institutions and isolated health care providers. That is why one of the biggest handicaps for the researchers at WHO is data collection.

To address that issue, a collaboration effort was established between WHO and UPC in the form of two different, yet related, projects named “WHO Information System to Control/Eliminate Neglected Tropical Diseases” (WISCENTD) and “WHO Integrated Data Platform” (WIDP).

Since that initial agreement, several researchers from UPC and a handful of students have received grants to build a common aggregated data platform that helps data collection of new and follow-up cases for the NTD department at WHO.

In February 2018, a collaboration grant was offered by the “Department of Service and Information System Engineering” (ESSI) at UPC. The main objective of the grant was designing and developing three new applications to solve different problems the joint project was facing.

In September 2018, the grant was renewed by the same department at UPC with the possibility to develop a final degree thesis. The thesis was established under a curricular practices agreement with the main purpose of documenting the software required by the joint project.

Hence, this is not a regular final degree project. The software presented here has a strong societal impact as it will likely help in the research for the diseases studied by the NTD department.

2. Context

2.1. Areas of interest

The official tool WHO uses to study diseases in the field is called “District Health Information System 2” (DHIS2). It is part of “WHO Integrated Data Platform” (WIDP) and is used by many programs such as Hepatitis, Health Workforce or Emergency Care.

DHIS2 is an open source software platform for reporting, analysis and dissemination of data for health programs, used in more than 60 countries and helping healthcare facilities around the globe. It was originally built by the “Health Information Systems Programme” (HISP) and nowadays the core development is coordinated by the “University of Oslo” (UiO).

Re-using such a big platform allows us to enjoy a lot of functionalities backed-up by rock solid software. But having all those functionalities for free has a hidden cost: generic software does not always fit specific requirements.

To solve and fill the requirements the project faces, we need to build internal DHIS2 forms and front-end applications. Most of our requirements are still not available in the core platform but we are eager to collaborate with the University of Oslo to bring the functionalities to upcoming releases of DHIS2 as an open source project.

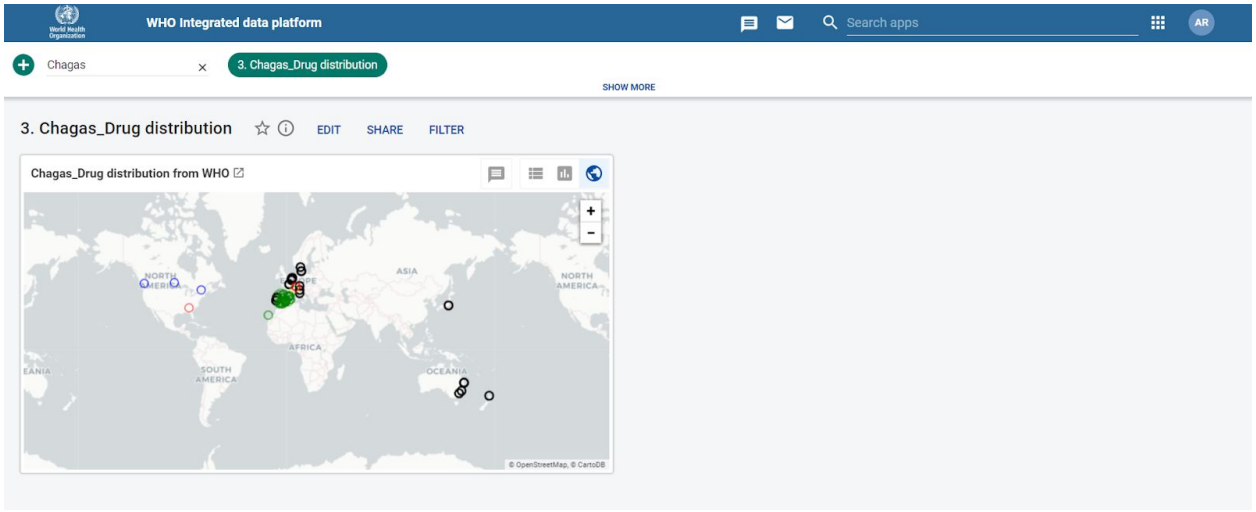


Figure 1. DHIS2 Web interface

2.2. Stakeholders

2.2.1. The World Health Organization

The main stakeholder we can observe is the World Health Organization. Inside the organization we include the IT personnel that manages their DHIS2 instances, the program managers and the researchers assigned to the Neglected Tropical Diseases department.

They directly benefit from the result of this project as end-users and provide the main software and system requirements. We could refer to them as the target audience of the project.

The IT personnel behind World Health Organization also participate in our testing pipeline and provide feedback of the applications developed in this thesis. They test the internal releases of the applications and update or review the global requirements.

2.2.2. Polytechnic University of Catalonia

Another important stakeholder is the Polytechnic University of Catalonia who is assigned to develop and maintain the tools that the World Health Organization needs for the whole project.

In a sense, the Polytechnic University of Catalonia acts as technical and development consultant to those working in the project. They sponsor this final degree thesis and are involved in the correct development of the tasks assigned to it.

Among others working in the project, we can include Petar Jovanovic as the director of the thesis, Alberto Abelló as the faculty representative and Alexis Rico as a software engineer.

2.2.3. Affected population

A sometimes underestimated and indirect stakeholder group are the people that are suffering from one of the 21 Neglected Tropical Diseases we are working for in the NTD department.

Even though they do not always participate on the project directly, they will largely benefit from the research of the NTD department. They are also the source of data that our platform consumes and it provides the required motivation for this project to succeed.

2.2.4. External DHIS2 users and developers

Since DHIS2 is a global open source platform and it is not exclusive of our research group, we build software that could be used by others in many health institutions.

We always have external DHIS2 users and developers in mind as we try to make our applications generic enough that can be abstracted to other projects at WHO or even in other organizations.

On the other hand, we also benefit from the development done by external DHIS2 developers by reading their documentation or using libraries that are useful to us.

The main communication tool with external developers that use DHIS2 is the official community forum[1]. We actively participate in that forum to investigate what others are accomplishing in DHIS2 and to share our latest results of application development.

2.2.5. DHIS2 core developers (University of Oslo)

The University of Oslo (UiO) handles the core development of the DHIS2 platform. As they are building an open platform, they usually show interest in third-party development of new applications that use their tools and technologies.

There are high chances that upcoming releases of DHIS2 include the functionality introduced by the applications we have developed in this thesis.

To ensure UiO benefits from this thesis, we report back any issue we discover in their infrastructure and we have constant communication with the official channels they provide, such as Jira[2] or GitHub[3].

We also regularly review the internal discussions of the web development team[4]. That includes reading the agenda and memorandums of their weekly meetings and trying to follow any recommendations they have for external developers.

[1] https://community.dhis2.org
[2] https://jira.dhis2.org
[3] https://github.com/dhis2/dhis2-docs
[4] https://github.com/dhis2/notes

3. Problem formulation

The normal usage for DHIS2 are local deployments that handle national or regional data.[5]

The main problem with DHIS2 is that it heavily relies on internal metadata and WHO is facing important issues when trying to scale DHIS2 to their complex infrastructure.

Instead of working with regional data, WHO provides support for a global platform. This involves sharing information across seven different DHIS2 instances and every instance needs to keep the internal data and metadata in sync with the rest.

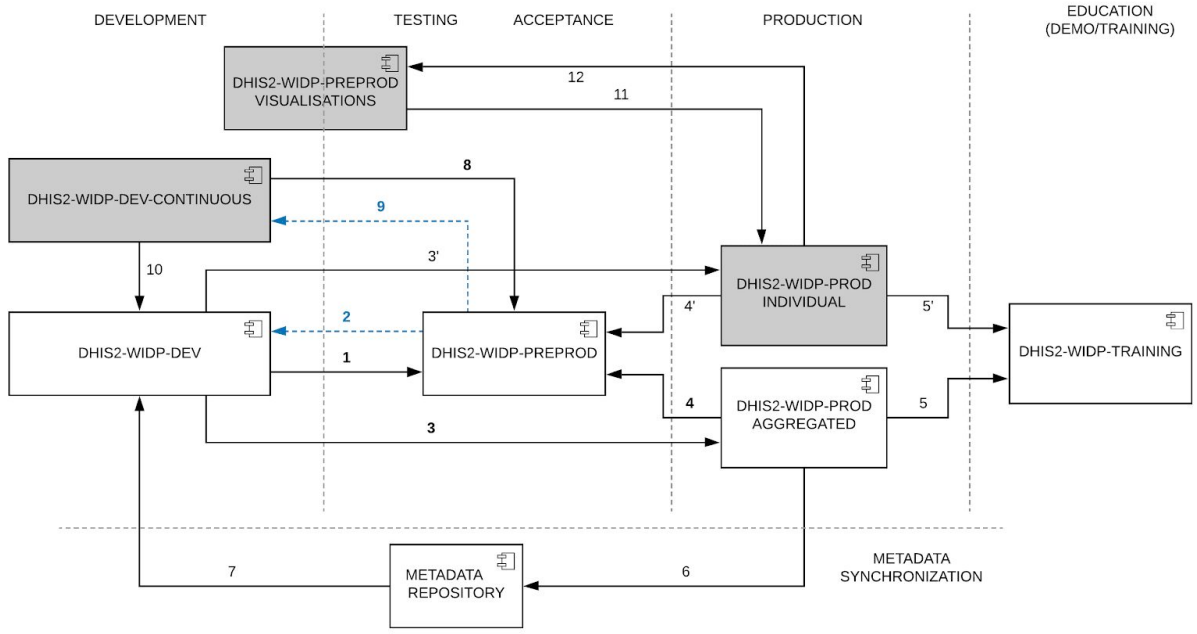


Figure 2. Different instances used by the WHO for the global data platform

Since this problem is too big for a single final degree thesis, we have selected three smaller problems that the student should be able to solve and that will contribute to reduce the complexity of the background problem.

[5] As an example, South Africa adopted DHIS2 as their main health information system

3.1. Packaging of bundled metadata

The first problem that this thesis should solve is the packaging of bundled metadata. By design, DHIS2 was built to be deployed in remote areas where Internet connection is unreliable. So instead of providing a centralized solution stored in data centers in Europe or America, they used a decentralized “build your own server” approach.

This enables anyone to deploy his own instance of DHIS2 and start collecting data without even having a reliable Internet connection. But the problem comes when sharing the collected data as you are forced to use the same internal schema[6] if you are willing to share data to another DHIS2 server.

Since the web interface creates internal unique identifiers during runtime, having exact copies of the metadata is not trivial and the internal references between metadata cannot be maintained manually.

Even though there’s an official application for exporting metadata, it does not respect the dependencies. The only solution to share exact copies of forms between DHIS2 instances is to query the web API and manually compile the shareable files that should be imported into the destination instances.

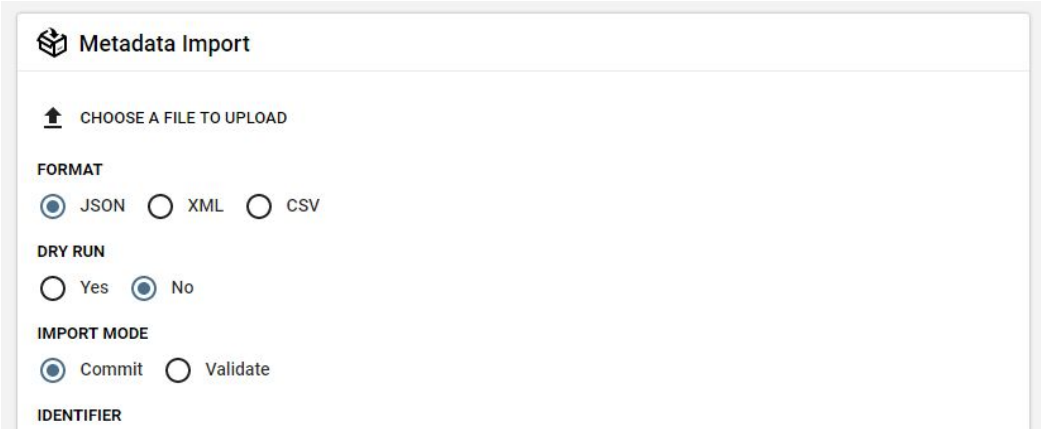


Figure 3. DHIS2 Official web application to import metadata packages

[6] The DHIS2 schema is called “metadata” and we will extensively use this term.
We could understand metadata as the internal abstraction of any form presented to the user

3.2. Bulk importing of data

Another problem the stakeholders face is to bulk import data collected in the field into DHIS2.

Even though DHIS2 is the best option when it comes to analyze and study health-related data, it is not designed to be a reporting platform. At least not when it comes down to a global infrastructure such as the one WHO works with.

Instead data collection has been traditionally done through surveys and spreadsheet files. This is a much easier way to share aggregated data in remote areas and underdeveloped countries.

Even though DHIS2 allows importing data from comma separated value files. Those files are not user friendly and it is impossible to import full featured excel files.

Also with complex forms, such as the ones that support disaggregation, the import procedure fails when the data is malformed.

Most countries end up using human staff to fill the data manually through the web interface.

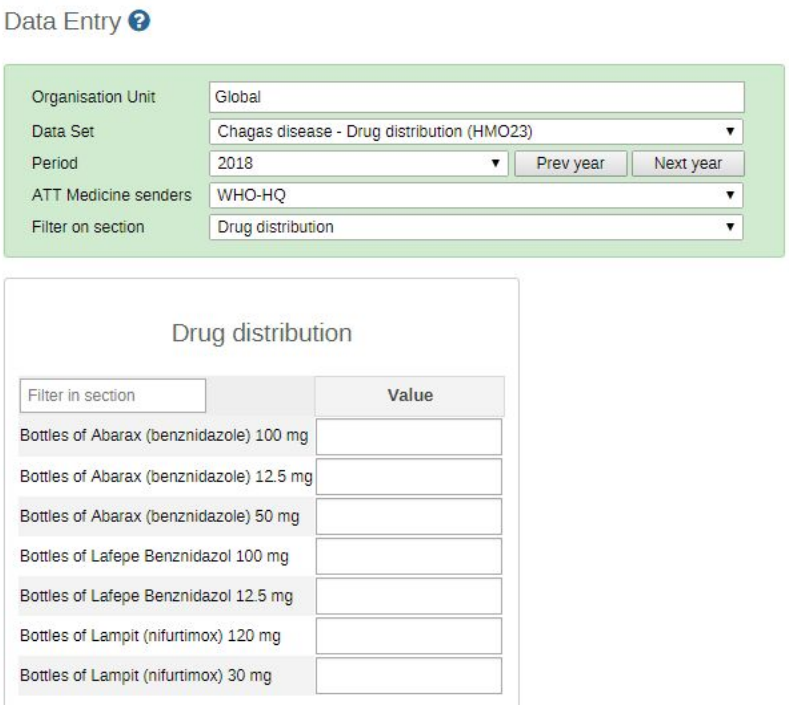


Figure 4. DHIS2 Official web application to fill data forms

3.3. Version tracking of metadata changes

The last problem this thesis aims to solve is the tracking of metadata changes in the forms available in our DHIS2 instances.

To ensure we have the highest uptime possible, we rely on a complex infrastructure that runs in powerful machines operated by WHO IT department. We are currently using seven different instances that replicate and expose the same metadata.

The problem comes when trying to maintain the same metadata synchronized across instances in a platform that was not designed for parallel execution. If we recall “Figure 2”, this issue is represented in the arrows 6 and 7.

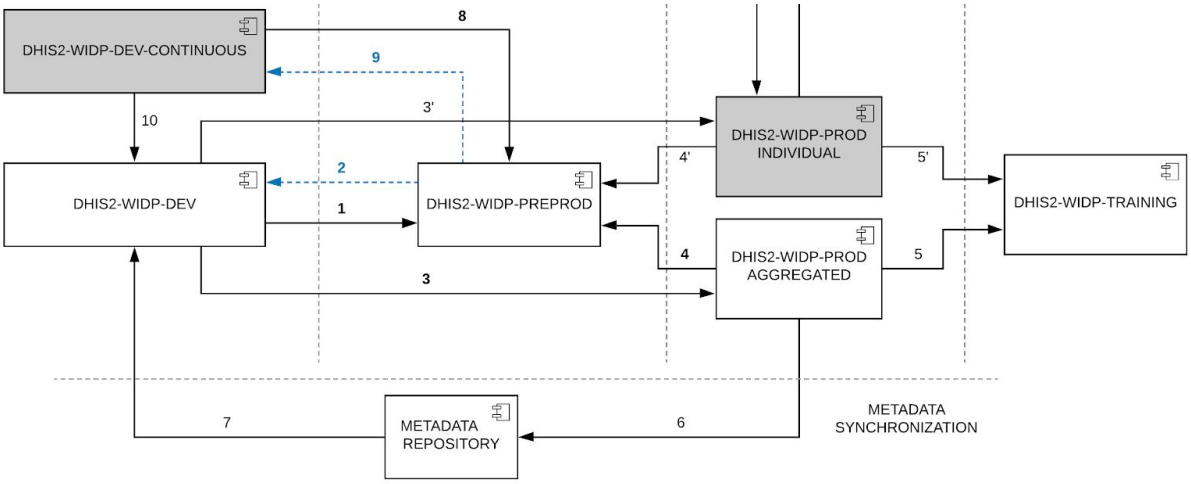


Figure 2. Different instances used by the WHO for the global data platform

Since the metadata can be introduced in different instances[7] and DHIS2 does not support a version control system that tracks metadata changes, we can only detect anomalies by manually examining the internal “last updated” date properties of the database.

We end up doing direct database clones of the SQL environment to secure metadata integrity and be able to roll-back the systems if any issues arise.

This approach does not allow traceability of the issues found and sometimes leads to data loss.

[7] Most team members, including program managers, work in “Development” and when the form is ready they push to “Pre-production”. If there are not found any blocking issue, the form appears in “Production”.

4. Specific objectives of the thesis

As we have seen in the previous section, we face a huge problem that can not be contained in a single final degree thesis. So, instead of targeting an unfeasible objective we have selected three smaller problems that will help to reduce the impact of the main problem.

After reviewing the viability of the project, we have determined that each of the three problems of this thesis requires the creation of a software application. The goals behind the development of the applications is to automate the tasks that are currently done manually by IT personnel.

We expect the student to fully integrate with the team at UPC and coordinate the implementation of the new applications with other members from the NTD department at WHO.

Before introducing the state of the art and initial project planning, we would like to introduce the specific objectives that this thesis is based on. In the final chapter of this thesis, we will also review the fulfilment of these specific objectives.

- **O1. Create a new application to Export Metadata**
    - O1.1. Allow the user to select any metadata of the running DHIS2 instance
    - O1.2. Recursively fetch metadata dependencies
    - O1.3. Create importable metadata packages in JSON format
- **O2. Create a new application to Bulk Import Data**
    - O2.1. Mirror the capabilities of the old “Bulk Data Upload” application
    - O2.2. Allow the user to create an excel template for datasets and programs
    - O2.3. Allow the user to import a previously created template
    - O2.4. Build user friendly excel files that include a legend with usage instructions
- **O3. Create new scripts to handle Metadata Synchronization**
    - O3.1. Traverse the entire metadata collection of a master DHIS2 instance
    - O3.2. Track changes to the metadata in a remote git repository
    - O3.3. Allow configuration for multiple synchronizations through a JSON file
    - O3.4. Update slave DHIS2 instances with the changes detected to master

Figure 5. Initial objectives of the thesis

5. State of the art

5.1. Existing applications and research

The problems we are facing are well-known issues in DHIS2 and there are currently no alternative solutions to look at yet.

We are building tools that might either become core functionalities of future versions of DHIS2 or supported as official apps in the app store[8].

Since we are not the only ones facing metadata management issues, we are making our research open source and we are willing to collaborate.

It is worth mentioning that the HISP South Africa institution is currently developing an application for metadata synchronization. We have investigated their approach to extract metadata from DHIS2 and it is very similar to the one we have developed in this thesis for the Advanced Export App.

The biggest difference between our applications is that we want to allow end users, that are not technical, to bundle and package their metadata in JSON files for public distribution.

Since the application of HISP South Africa is still in the works, we are constantly monitoring the improvements they achieve in their public GitHub[9]. In case they create something we can benefit we will introduce it to our own research.

As the collaboration with WHO has lasted several years, we have some older applications developed at UPC that are useful to the topics we research in this thesis. We reviewed the solutions presented in those previous applications and extracted some useful insight.

Finally, we have reviewed two different final degree projects[10] that are relevant for our project and we tried to learn as much as we could from their research, the titles of the TFGs are: “Parametrization of the chagas disease surveillance system for the WHO” and “Performing data lineage for an ingestion system to a data lake”

[8] https://play.dhis2.org/appstore
[9] https://github.com/EyeSeeTea/metadata-synchronization-blessed
[10] The reports are available in the Bibliography section of this thesis.

5.2. Available technologies

The development of applications for DHIS2 has a very well defined scaffolding. Their back-end offers a powerful Web API[11] to access internal data and metadata.

In order to maximize the results, the University of Oslo recommends to use the same technologies and libraries they are using in their core applications. That includes JavaScript and TypeScript as main development languages, React as front-end web framework or Cypress and Jest as testing frameworks.

In case of doubt about the methodologies to follow, they recommend to visit their GitHub and read the publicly available code of their apps such as the “Maintenance App” or the “Maps App”.

5.3. Helper libraries

To speed-up the development of third party apps we have available two main libraries built by the University of Oslo that help to streamline the development of new applications. This makes possible to build third party applications that act and feel the same as the official applications and that with little to none effort can be later included in upcoming releases of DHIS2.

The first library is called d2[12] and provides a JavaScript connector for the Web Api, this allows to access easily in an asynchronous way the back-end without needing to build custom methods and communications. It is really helpful to center the development only in the application without worrying of how it works beneath.

The second library has a similar name and is called d2-ui[13] and provides a toolbelt of React components that mimic the look and feel of the official applications. In fact, all the new core applications use this exact library and we can build streamlined applications with ease in the same direction the University of Oslo does. They offer an always up-to-date documentation webpage[14] for these two libraries that serves as the entry point for any developer willing to work with third party applications for DHIS2.

[11] https://docs.dhis2.org/master/en/developer/html/dhis2_developer_manual_full.html
[12] https://github.com/dhis2/d2
[13] https://github.com/dhis2/d2-ui
[14] https://d2-ci.github.io/d2

5.4. DHIS2 community

One of the strong points of DHIS2 as a platform is the big community that is backing the project. Hundreds of individuals are building third party applications for DHIS2, and we greatly benefit by tools provided by developers around the globe who are working hand to hand.

There are different non-gubernamental health institutions around the globe, such as HISP Tanzania[15] and HISP India[16], that provide loads of open source utilities around DHIS2.

And even some private companies share their products as open source software. For example, one of the main contributors to the community is EyeSeeTea[17], a company that maintains part of the WHO infrastructure and builds applications for WHO, non gubernamental organizations and other end-users. They host a large repository of projects that can be reused and freely reviewed.

5.5. DHIS2 core developers

The core team behind DHIS2 takes a strong stance in what open source means. They do not only create free to use, modify or redistribute software, instead they publicly share all the internal discussions and decisions they make.

They have created a big knowledge base where anyone is able to view, review and participate in any ongoing discussion topics for current and future development.

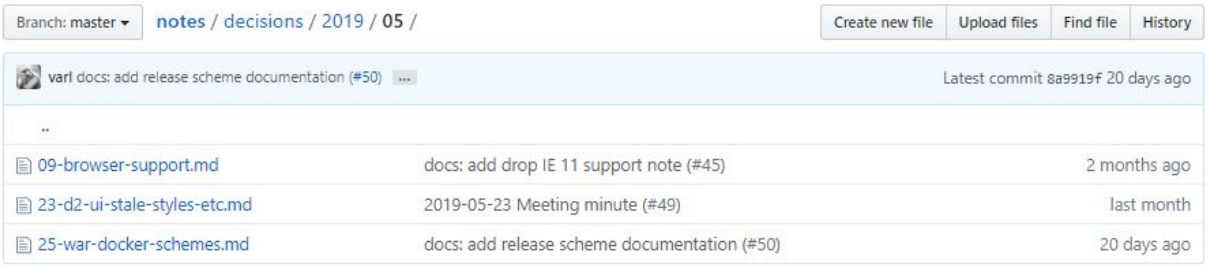


Figure 6. DHIS2 core development decisions made in May 2019

[15] https://github.com/hisptz
[16] https://github.com/hispindia
[17] https://github.com/EyeSeeTea

6. Initial project planning

6.1. Scope of the project

Since my personal collaboration started in February 2018 we could say the scope is bigger than a normal final degree thesis.

The main objective of the project is building up to three different Web Applications that attack the problems mentioned before.

In any case, we do not limit the project to the design and software implementation, we also contemplate deployment and end-user feedback tasks in the thesis.

6.2. Potential obstacles

6.2.1. Fixed timing

The main concern we have is that the development is fixed to finish on June 2019. Since our objectives are very extensive we are worried that any change to the requirements could potentially lead to unexpected delays.

Every application development starts sequentially after the previous one but multiple applications coexists during the integration testing with end-users and the documentation phases. If a blocking issue appears then, the next iteration could be delayed.

If we end up with multiple delays that could lead to missed objects and rendering a complete failure to the project.

Commonly final degree thesis attack one major problem but during this thesis we will try to solve three different problems and create three different applications that will be tested by end-users.

It is more than probable that due to the agile nature of the project we face deviations from the initial project planning.

6.2.2. New requirements

We work on-demand with WHO requirements and the planning is always subject to change. The thesis was approved with the current planning back in September but if a rising development is more urgent, the whole project could suffer from a complete redesign.

We do not expect this to happen because the problems that this thesis attacks are high priority at the World Health Organization. But since the applications to be developed will be used in ongoing operations by IT personnel, we expect new requirements to arise.

We will always try to take into consideration those new requirements and build the solution that better fits into the current organization. Still, if any of these requirement changes are faced in the final stages of the thesis we might be unable to include them into the project.

6.2.3. Technical issues

The thesis has as a main objective to develop brand new applications that offer a functionality that has never done before.

That means that the development could be restricted by limitations imposed by third party technologies such as DHIS2 or React.

But to make sure this thesis was viable, we did an initial proof of concept during the first half of 2018. The proof of concept assessment was done to ensure the viability of the thesis and there are no signs of any blocking technical issues.

6.2.4. Research and innovation

There are no existing solutions for the problems presented in this thesis, we expect the student to perform an in-depth research task.

This is a potential obstacle as we can not guarantee or account the time dedicated for innovation. We are confident that DHIS2 allows us to fulfil all our expectations and produce software that establishes the future of the platform.

6.3. Methodology and rigor

6.3.1. Overview

The working methodology is an agile based methodology to quickly adapt to changes and we have regular meetings with the director of the project and monthly meetings with WHO representatives.

All the feedback is gathered through a management platform called Redmine, that acts as a project management and issue tracker. Inside Redmine we use a per-application Kanban board that handles the task status, feature development, functional requirements and detected bugs or issues.

Since we collaborate with dozens of developers and researchers across Europe we heavily rely on this project management utilities in a day-to-day basis.

6.3.2. Development tools

The development of the different applications is based on JavaScript based technologies such as NodeJS and ReactJS. Each application is intended to be a Single Page Application that is packaged into a release zip that can be installed on a DHIS2 instance.

We plan to test the applications locally through Webpack, a JavaScript tool that builds a development http server that hosts and emulates a real scenario. We have already built a set of functions that allow to work localhost with remote DHIS2 instances.

Even though we could develop the applications directly on a text editor we have chosen to use IntelliJ suite, specifically WebStorm as an IDE (Integrated development environment) to improve the performance of the software engineer.

We will also use Docker to create isolated environments that host local DHIS2 instances. To prevent incompatibility with local machine requirements we will build a virtualized container that replicates the same resources the production servers have.

Finally we will use Insomnia, an open source REST API client to manually communicate with the web API that DHIS2 exposes.

6.3.3. Development cycles

To reduce the probability of failure we have planned short development cycles with continuous feedback from the stakeholders. Weekly meetings ensure that the requirements are being met and that progress is happening.

Every iteration also allows us to do a brief retrospective and keep improving the quality as a whole. We focus on functionality but code quality is a must in this project since we want to make it as maintainable as possible.

This thesis will provide new applications that might be used by researchers during the next five years, or even more.

6.3.4. Integration tests

Every time a new functionality is implemented we tag a new version release and notify the end-users they can test the new features. We also build a production ready package to be offered publicly on GitHub.

We also keep updated the three different development DHIS2 instances with the latest version of the applications so that we can receive all the feedback possible.

The end-users do not only test new features, they also provide create issues through Redmine and assign them directly to the software engineer.

Those issues on Redmine include detected bugs and feature requests to be implemented during the next iteration of the development.

Also during the regular face-to-face meetings we collect and discuss the overall user experience and review the evolution of the applications over time.

The development team is always eager to introduce new features that could make the applications developed more useful to the required needs.

6.4. Monitoring tools and validation

To monitor the changes to the code base we use Git and GitHub as main tools, we have an organisation[18] prepared to host the different projects for each application.

On GitHub we will find the main source branch, multiple development branches for new features and release tagging when the milestones are reached.

To communicate between the different members of the project we use institutional mail addresses and Cisco WebEx for online video calling.

To monitor the functional requirements we use Redmine with tickets that allow feature discussion between team members and end-users. It allows task definition, time estimation, automatic gantt chart building and kanban board management. We regularly log hours in the Redmine and monthly review the progress achieved.

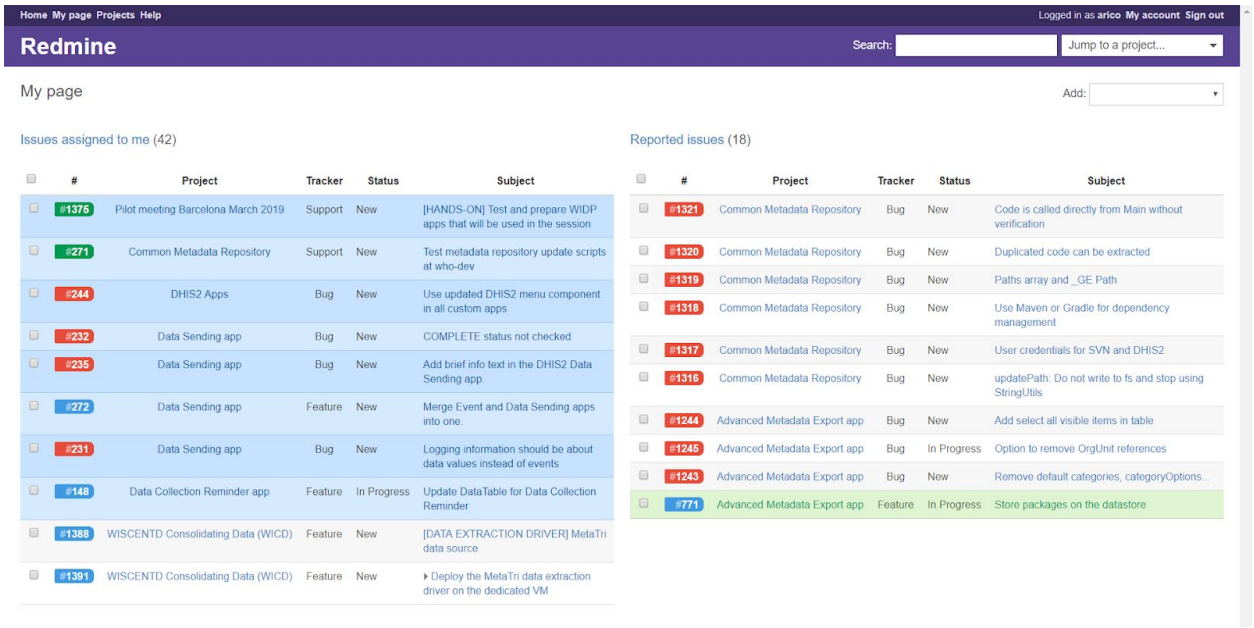


Figure 7. Redmine dashboard used by WHO and UPC

[18] https://github.com/WISCENTD-UPC

6.5. Development phases

6.5.1. Initial Planning and Scope

In September, with the director of the thesis we discussed an initial planning of the tasks that could potentially fit inside the thesis and defined a global scope that would allow us to start the development before building the refined planning of the GEP course.

Note that the GEP course happens in the middle of the thesis with the development started months before. The documents and presentations of GEP include a revised thesis planning.

6.5.2. Viability analysis

After we decided the project scope and which applications we had to develop before June 2019 as per the requirements of WHO, we began a viability study of the existing solutions and how they could be improved.

6.5.3. Application development

The development of the three applications will happen at the same time but since they address WHO deadlines the priority of them is decided in each Milestone.

Each application requires a separate design phase that includes both a brainstorming with the stakeholders the features of the application and the decision of which technologies are going to be used.

As soon as we have usable versions of the application, we build beta testing releases and distribute them internally to the rest of the team members. We always try to fix the issues introduced as soon as possible.

6.5.4. Thesis documentation

When all the features have been implemented, a final documentation phase happens so that everything can be later reviewed. That final documentation will also include a future work planning and road-map with a list of new features worth implementing by the thesis student or future project members.

6.6. Temporal planning

The collaboration between the student and the UPC for the WISCENTD project started the 19th of February 2018. Still, the real start date for the thesis is the 1st of September 2018 and finishes the 30th of June 2019 for a total of 10 months dedicated to the research and study of the thesis.

The contract associated with this thesis determines that the amount of dedication for the collaboration is of a total of 735 hours with a 20 hour per week dedication. The estimation does not account additional time spent writing the thesis memory and other management tasks.

Stage	**Estimated dedication (in hours)**
Initial Planning and Scope	30
Viability analysis	60
Advanced Metadata Export Application	245
Bulk Load Application	220
Metadata Repository Application	180
Total time	**735**
Thesis documentation	275
GEP Course	75

Figure 8. Estimated time dedication table

6.6.1. Task planning

To begin with the temporal planning estimation we are going to resume all the different tasks we have identified for this thesis.

We have grouped the development specific tasks with their respective application. We have also given a two month estimation for probable delays and thesis documentation.

TASK	HOURS	START	END
Initial Planning and Scope	30.00	3-Sep	20-Sep
Viability analysis	60.00	14-Sep	15-Feb
Advanced Metadata Export	245.00	20-Sep	19-Apr
Initial design and set-up	20.00	20-Sep	28-Sep
Feature development	150.00	1-Oct	28-Jan
Integration test with users	60.00	5-Nov	19-Apr
Final documentation	15.00	15-Apr	19-Apr
Bulk Load	220.00	22-Nov	26-Apr
Initial design and set-up	15.00	22-Nov	28-Nov
Feature development	130.00	28-Nov	15-Mar
Integration test with users	60.00	27-Feb	26-Apr
Final documentation	15.00	22-Apr	26-Apr
Metadata Synchronization	180.00	2-Jan	3-May
Initial design and set-up	10.00	2-Jan	4-Jan
Feature development	120.00	7-Jan	1-Mar
Integration test with users	40.00	1-Apr	3-May
Final documentation	10.00	29-Apr	3-May
Total development	**735.00**	**3-Sep**	**3-May**
Product Management	**120.00**	**3-Sep**	**28-Jun**

Figure 9. Initial task planning

6.6.2. Gantt diagram

With the task planning in mind, we have also elaborated an initial Gantt diagram that shows the overlapping of the different tasks over the thesis lifespan.

In the gantt chart we can observe both the two month estimation for delays and the overlapping of development tasks to match the project's internal deadlines for the applications.

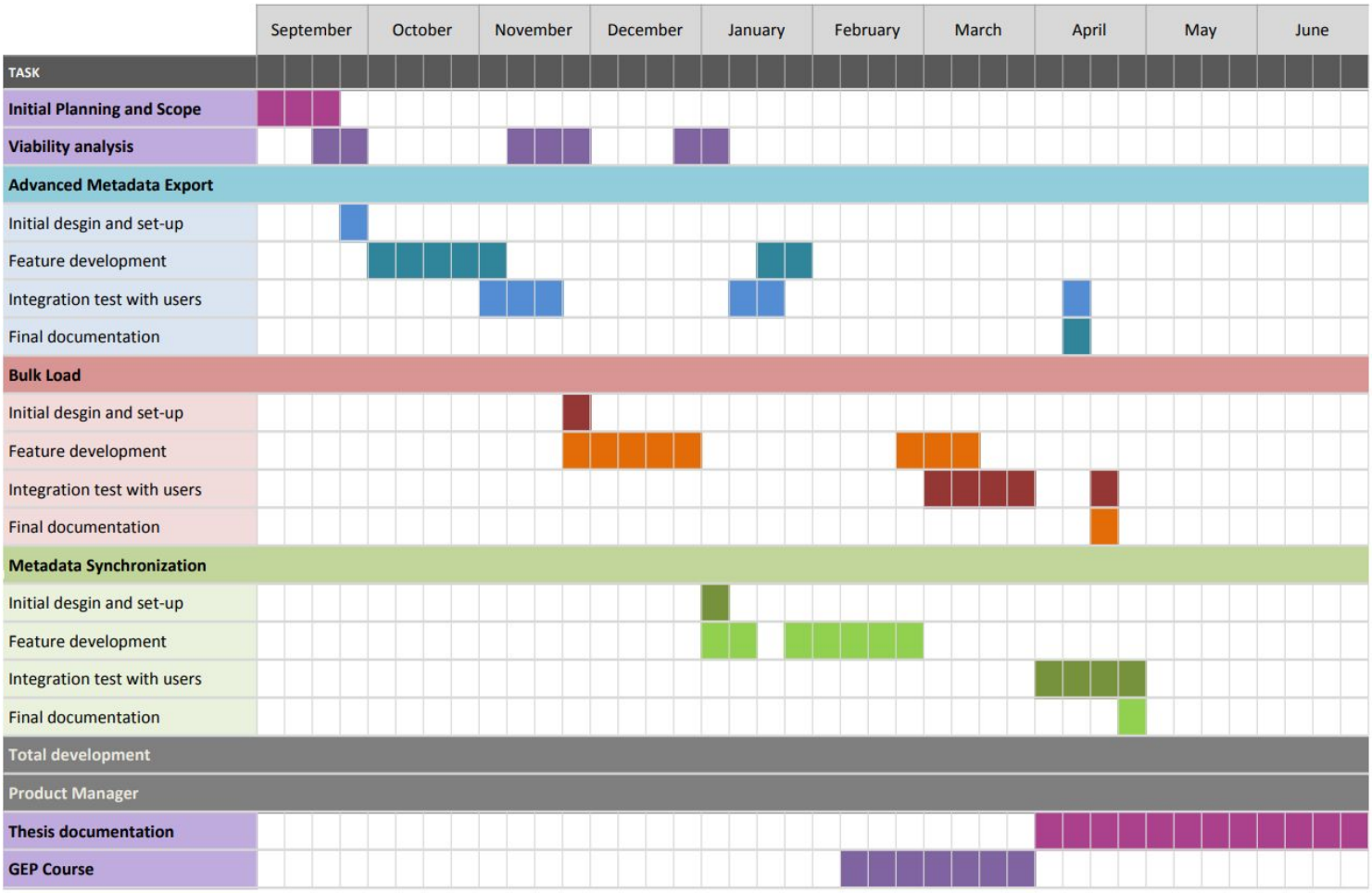


Figure 10. Gantt diagram

6.6.2. Action Plan

The agile methodologies we use in this thesis are always subject to change and the schedule is always changing. Some of the tasks do not happen sequentially and the next task is sometimes decided by external priorities.

If the tasks are concluded before the expected date, the next programmed task is set to start before planned to maximize the performance of the thesis.

At least once a week a meeting with the director is arranged to review the work done since the previous meeting and to program the pending tasks. Any other relevant topics to discuss are handled in thess weekly meetings.

Additional meetings happen when more follow-up is needed or when the whole team behind WISCENTD has the Sprint Review meeting, usually every one or two months.

Although we have not prepared a detailed alternative plan the whole project is based on an agile approach and we continuously update and improve the way to proceed.

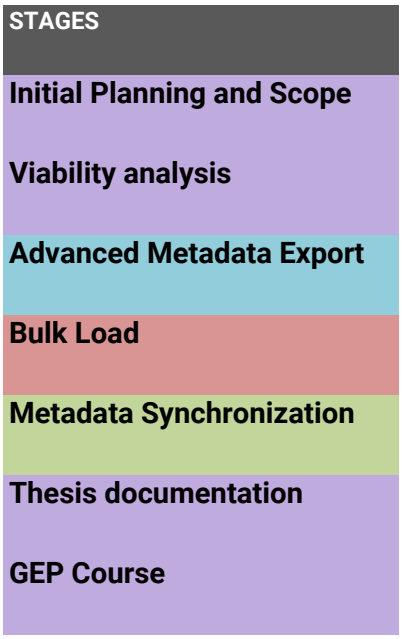


Figure 11. Action Plan Stages

6.7. Resources and budget

The project is developed using the resources of an European fund to the research of neglected diseases and a research fund of Generalitat de Catalunya assigned to the UPC as a research institution.

The main costs are derived from human resources, hardware equipment and software licenses but other unexpected costs can happen at any time.

6.7.1. Hardware resources

First of all, a development laptop will be required with enough power to handle the development environment tools and multiple DHIS2 instances at the same time. We plan to use for this specific part a MSI PS63 with 8th generation i7, 16GB of RAM and 512GB SSD.

Secondly, we require a development server hosted at UPC to deploy the applications for alpha testing before releasing the milestones required by WHO. The server is rented by UPCnet and hosted at http://who-dev.essi.upc.edu:8081.

Finally, we will use an integration testing server hosted at WHO to release the milestones and coordinate the beta testing with end users. The server is rented by WHO IT and hosted at https://extranet.who.int with different instances: Production (https://extranet.who.int/dhis2), Develop (https://extranet.who.int/dhis2-dev) and Training (https://extranet.who.int/dhis2-demo).

Product	Price	Useful life	Amortization
Development Laptop	1.500,00 €	5 years	150,00 €
UPC Server	1.850,00 €	5 years	185,00 €
WHO Server	2.350,00 €	5 years	235,00 €
Total	**5.700,00 €**		**570 €**

Figure 12. Hardware resources budget

6.7.2. Software resources

The main software used for the development of the project is an IDE (Integrated development environment) built by JetBrains Inc. The software suite is called IntelliJ IDEA and the main utility used in the thesis is called WebStorm. The applications are built using NodeJS, ReactJS, Javascript, TypeScript and other web technologies.

The operating system for the servers is a Linux based Ubuntu distribution that allows the execution of the PostgreSQL database and the java runtime for DHIS2.

To automate the execution of the web services we use Docker community and other utilities such as Git and GitHub.

Product	Price	Useful life	Amortization
JetBrains Suite (WebStorm)	Free for Open Source Projects	N/A	N/A
NodeJS and ReactJS	Free	N/A	N/A
JavaScript and TypeScript	Free	N/A	N/A
Ubuntu Server	Free	N/A	N/A
PostgreSQL	Free	N/A	N/A
Docker Community	Free with Registration	N/A	N/A
Git and GitHub	Free for Open Source Projects	N/A	N/A
Total	**0,00 €**		**0,00 €**

Figure 13. Software resources budget

6.7.3. Human resources

The main development of the project is done by a software engineer and a product manager.

The software engineer will take care of the design and implementation of the required applications while the Product Manager will organize the tasks of the Engineer while following and adapting the project management.

The software engineer should be competent enough to fulfill all the tasks alone without any major guidance other than researching the available knowledge base or asking the Product Manager.

The product manager should have expertise guiding software development teams, and natural ability to coordinate and improve the performance of the software engineer.

Role	Estimated hours	Price per hour	Total cost
Project Manager	120	45 €/hour[19]	5.400,00 €
Software Engineer	735	30 €/hour[20]	22.050,00 €
Total	**855**		**27.450,00 €**

Figure 14. Human resources budget

The price per hour is estimated using actual data from LinkedIn Salaries in Barcelona for each role, these salaries do not account the real state tax nor the social security fee.

These hidden costs are handled either by the employee, in the case of the real state tax as a retention, or the company, in the case of the social security fee, but never by the project.

[19] https://www.linkedin.com/salary/explorer?countryCode=es&regionCode=5064&titleId=4
[20] https://www.linkedin.com/salary/explorer?countryCode=es&regionCode=5064&titleId=9

6.7.4. Budget monitoring

The approved budget is monitored every month to avoid unexpected costs and update the usage of the remaining budget. The development strategy could be easily modified and the budget assigned recalculated.

Hardware costs are not expected to change because the equipment is enough for the project and it still has a valid warranty agreement.

In the unlikely case that the hardware malfunctions we would only need to mitigate the downtime with rented hardware until a replacement from the manufacturer happens.

Software costs should stay at zero and are not subject to change as everything we use is free and redistributable. Anyway the costs of any potential license that we could need is covered under the unexpected costs budget.

The only part we need to extensively monitor is human resources as any change in the task estimation directly produces unexpected costs that affect the budget.

The following formula can be used to express the deviations:

$$\text{Cost deviation} = (\text{Estimated cost} - \text{Real cost}) \cdot \text{Real hours}$$

$$\text{Consumption deviation} = (\text{Estimated hours} - \text{Real hours}) \cdot \text{Estimated cost}$$

6.7.5. Unexpected and contingency costs

In all projects there are always hidden costs that appear during the development and cannot be predicted beforehand.

Those hidden costs are normally because of extraordinary hours, hardware malfunctioning and repair, additional software licenses or external services.

To allocate enough budget for these unpredicted costs we establish of the total required budget a 15% margin for unexpected costs and a 5% margin for contingency costs.

6.7.6. Cost distribution across development tasks

With the estimation of hours and the price per hour we can estimate the cost for each task.

TASK	HOURS	START	END	COST
Initial Planning and Scope	30.00	3-Sep	20-Sep	900.00 €
Viability analysis	60.00	14-Sep	15-Feb	1,800.00 €
Advanced Metadata Export	245.00	20-Sep	19-Apr	7,350.00 €
Initial design and set-up	20.00	20-Sep	28-Sep	600.00 €
Feature development	150.00	1-Oct	28-Jan	4,500.00 €
Integration test with users	60.00	5-Nov	19-Apr	1,800.00 €
Final documentation	15.00	15-Apr	19-Apr	450.00 €
Bulk Load	220.00	22-Nov	26-Apr	6,600.00 €
Initial design and set-up	15.00	22-Nov	28-Nov	450.00 €
Feature development	130.00	28-Nov	15-Mar	3,900.00 €
Integration test with users	60.00	27-Feb	26-Apr	1,800.00 €
Final documentation	15.00	22-Apr	26-Apr	450.00 €
Metadata Synchronization	180.00	2-Jan	3-May	5,400.00 €
Initial design and set-up	10.00	2-Jan	4-Jan	300.00 €
Feature development	120.00	7-Jan	1-Mar	3,600.00 €
Integration test with users	40.00	1-Apr	3-May	1,200.00 €
Final documentation	10.00	29-Apr	3-May	300.00 €
Total development	**735.00**	**3-Sep**	**3-May**	**22,050.00 €**
Product Manager	**120.00**	**3-Sep**	**28-Jun**	**5,400.00 €**
Thesis documentation	275.00	1-Apr	28-Jun	- €
GEP Course	75.00	11-Feb	29-Mar	- €

Figure 15. Cost distribution table with task description

6.7.7. Total estimated budget

After reviewing the human, hardware, software and unpredicted costs we can estimate the total budget required for the project.

Concept	Estimated cost
Human Resources	27.450,00 €
Hardware	5.700,00 €
Software	0,00 €
Unexpected and contingency costs (15% + 5%)	6.630,00 €
Total	**39.780,00 €**

Figure 16. Total estimated budget

7. Sustainability report

7.1. Sustainability matrix

Overall the student is more than happy with the results of this report. In collaboration with the stakeholders we have taken into consideration several aspects that ensure the sustainability of the project.

	Project development	Exploitation	Risks
Social	10	20	0
Environmental	7	16	0
Economic	9	16	0
Sustainability score	26 (over 30)	52 (over 60)	0 (between -60 and 0)

Figure 17. Sustainability matrix

The explanation of the results can be found in the following sections.

7.2. Social sustainability

This thesis will largely aid the research for neglected tropical diseases and will improve the distribution of medicines to patients worldwide. We are indirectly helping thousands of people that deserve a cure for deadly diseases.

Also the software built under this project is open source and will be used and reviewed by third parties in the future, enabling others to benefit even more people with our applications.

And finally the project is part of the education of the student and the lessons learnt during the development of his thesis will contribute to his personal development and work experience.

7.3. Environmental sustainability

The main improvements to the environmental dimension of the project are the hardware investments.

We have an estimated product life of 5 years for each equipment taking as reference studies on hardware amortization.

Since we already accounted a part of the budget for these hardware equipments and we might not completely use the equipment.

To improve the environmental sustainability, we have reused all the hardware from different entities.

- Development laptop: The student will use his own laptop for development since his machine is powerful enough and he will maximize the usage of the equipment.
- UPC Server: Instead of investing on our own server for development purposes we asked “Laboratori de Calcul” for a virtual machine in one of their servers called “Leon”. The server provides resources for dozens of other projects for the ESSI department at UPC.
- WHO Server: The dedicated server we have acquired will be reused for other projects after the finalization of this thesis.

7.4. Economic sustainability

We have presented a detailed planning and an estimated cost for the project, where it has been included human and material resources.

The project already consists of the minimum required human resources, such as one product manager and one software engineer with an average rate per hour.

The project also tries to use the cheapest hardware available that offers the performance we need and all the software licenses used in the project are free for open source projects as ours.

8. Design and implementation

8.1. Advanced Export App

8.1.1. Main objectives

The main objective of the application is to browse the metadata of the running DHIS2 instance, select the desired objects to be exported, review the dependencies bundled with the selected objects and download a shareable package.

The resulting package is compliant with DHIS2 sharing standard so that it can be later imported in another instance whilst keeping the dependencies and relations of the database.

The application is also configurable so that advanced users can choose which dependencies should be included or ignored. These configurations are different for every export.

8.1.2. Technical overview

The advanced export app is built in modern JavaScript using the React framework to design declarative UI web components. To build the application template we used the official command line utility “Create React App” provided by Facebook[21] and that allowed us to quickstart the initial development set-up.

Since the application runs on-top of an existing DHIS2 instance we use the official “d2” to communicate with the core back-end and maintain the look-and-feel of the instance where it is being executed. To maintain the look and feel of DHIS” we also use the official UI components that are available in the “d2-ui” library.

To manage the communication between the different layers for the advanced export app we use a shared-state store service called Redux[22]. This communication layer synchronizes the state of the foreground user interface and the background logic services while reducing the coupling between components.

[21] https://github.com/facebook/create-react-app
[22] https://github.com/reduxjs/redux

All the logic to fetch metadata elements and obtain the dependencies is handled by a single service called “Extractor”. We keep the extraction as a single source of truth by using the “Singleton Design Pattern”. This design pattern reduces any possibility for data inconsistencies.

We provide public methods for initial set-up, element fetching and concurrency synchronization. All these methods act as an internal API for the application asynchronous logic.

```
ExtractorClass.prototype.init = function (builder) {...};

ExtractorClass.prototype.initialFetchAndRetrieve = async function (elements) {...};

ExtractorClass.prototype.fetchAndRetrieve = async function (json) {...};

ExtractorClass.prototype.recursiveParse = async function (element, type) {...};

ExtractorClass.prototype.parseElements = async function (elementsArray) {...};

ExtractorClass.prototype.shouldDeepCopy = function (type, key) {...};

ExtractorClass.prototype.attachToExecutor = async function () {...};
```

Figure 18. Advanced Export App’s Extractor public methods

In the Extractor class we also use different external libraries. For example, we use “lodash” as a helper library to manipulate the resulting package, “axios” to perform the API calls between the application and the DHIS2 instance or “traverse” to visit the metadata tree’s nodes.

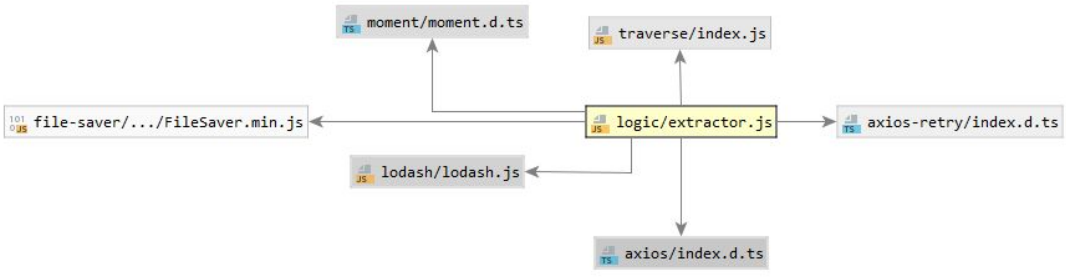


Figure 19. Advanced Export App’s Extractor library dependencies

8.1.3. Application evolution

Since the project is agile-based, we released an initial version of the application showcasing the user interface and basic functionality. We only included in that release the creation of a package that included the selected elements without any dependency fetching.

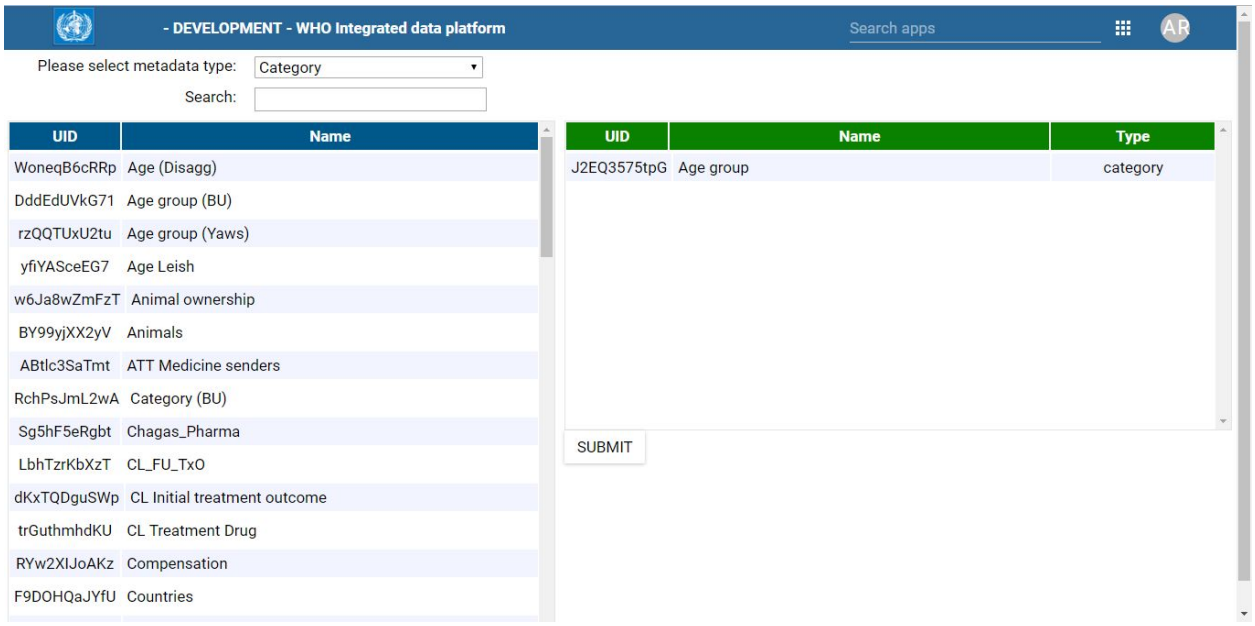


Figure 20. Advanced Export App v0.1.0

That initial version allowed us to verify with the stakeholders that we understood the requirements behind the application. We also received feedback on what features we should focus on.

The main feature the stakeholders needed was to browse the metadata catalog with ease. Some metadata collections in the WHO DHIS2 instances contain more than 100.000 objects and our application should be able to visualize over 250.000 objects at the same time.

We were also asked to implement collection grouping and filtering over the internal DHIS2 id and the display name of the object. That would simplify the selection of any object of the instance while reducing the complexity of a paginated table as originally intended.

As a side effect of the feedback obtained, we decided to use modern table react components instead of traditional HTML tables. That improved both the user experience and performance.

The second version we released to the stakeholders already included the grouping and filtering to browse the metadata collection and the main objective of the release was to verify the user experience evolved as expected.

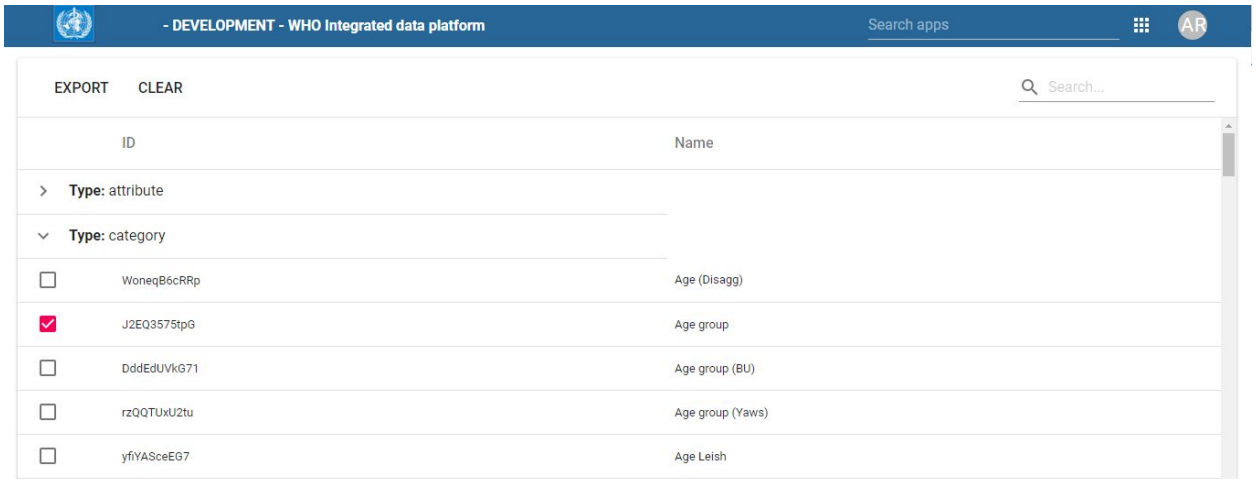


Figure 21. Advanced Export App v0.2.0

The feedback obtained from that second version was good but some end-users pointed out that since we could potentially hide selected items due to filtering and/or grouping we could introduce again the two tables approach of the first release. That change would allow the user to verify the selected objects before exporting.

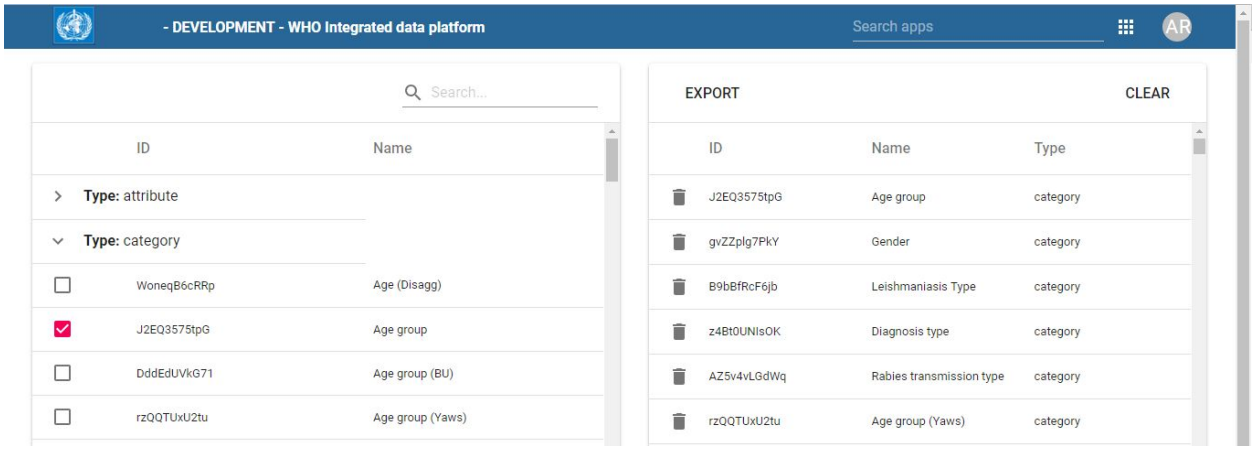


Figure 22. Advanced Export App v0.2.1

The front-end interface was pretty much decided after that release but we lacked the ability to visualize which dependencies the algorithm had included in the resulting package. We decided to adapt the checkbox to include a three state icon (user-selected, dependency, ignored).

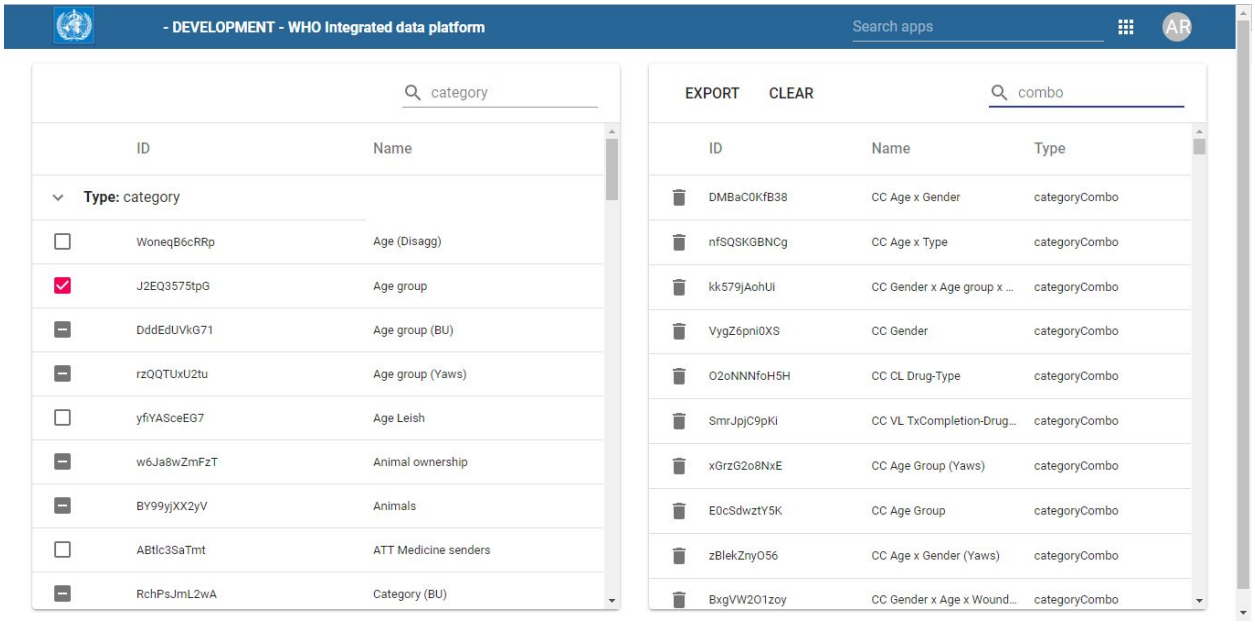


Figure 23. Advanced Export App v0.2.2

Even though we finally had a version the user felt comfortable to use, we only included the core functionality based on the stakeholder requirements. We still had to focus on the advanced features that we were missing.

We introduced two different panels to accommodate the missing features, a basic options menu meant for any user of the app and an admin menu that allowed in-depth app configuration.

The first “options” menu includes the ability to remove user related identifiers or organisation units[23] of the resulting metadata package so it can be successfully imported on any DHIS2 instance. Since the user could also create a metadata package for a compatible DHIS2 instance we allowed them to configure it per export.

[23] An organisation unit is a DHIS2 concept for hierarchical institutions (ie: countries, hospitals…) WHO uses organisation units to manage the medical centers of all the world by country and region. The organisation unit collection consists of 109,075 different entities.

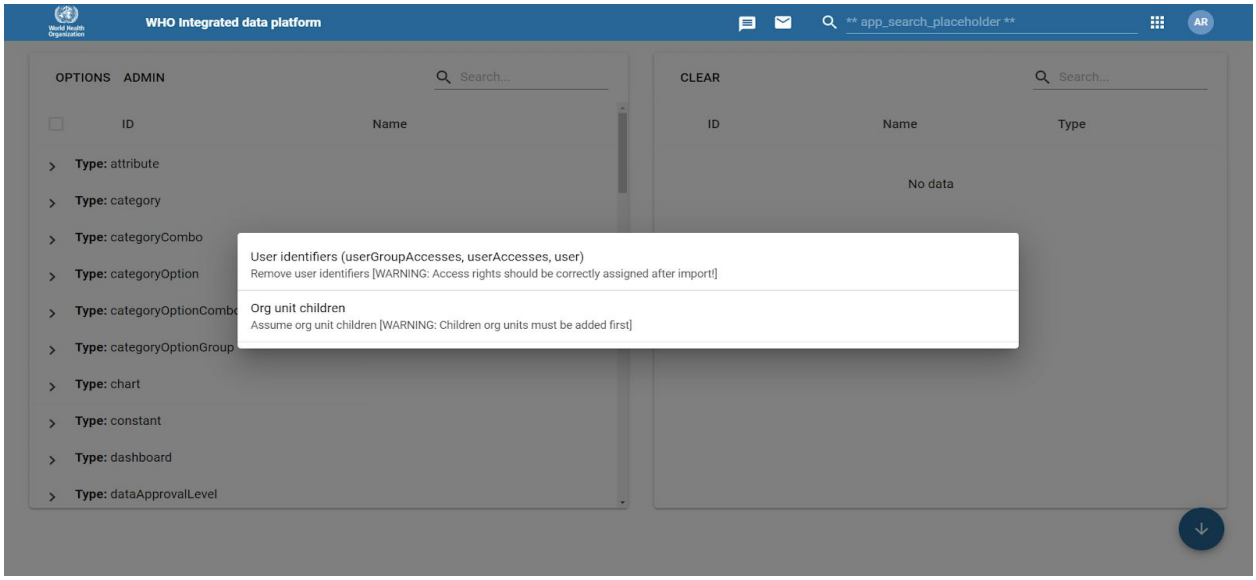


Figure 24. Advanced Export App showcasing the “Options menu”

The second “admin” menu allows the user to select the include rules that the application algorithm will use to determine if a dependency should be fetched or ignored. This menu allows to create completely different metadata packages depending on the advanced user needs. We allow to store the rules for later usage in an instance-shared DHIS2 store called DataStore so that an advanced user is able to create advanced rules that anyone can later re-use.

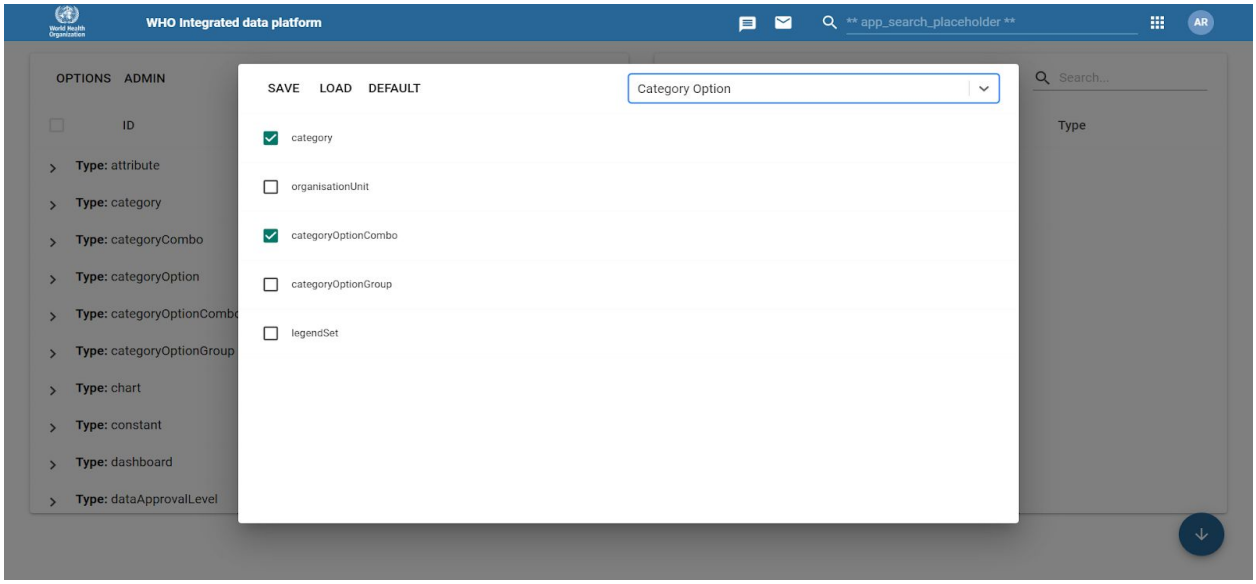


Figure 25. Advanced Export App showcasing the “Admin menu”

8.1.4. Conclusions

The advanced export app application development has provided the stakeholders with 11 different releases that contained new features and bug fixes.

In the last sprint meeting held in Barcelona (13th of June 2019) the stakeholders asked for some improvements that directly affect the algorithm and before concluding the thesis we will provide a final release.

Some of the new improvements include the addition of a new configuration option that allows to filter the properties included in the bundled metadata package.

Starting with version 0.2.5 (released the 6th of December 2018) we open sourced the application under the project GitHub organisation[24] meaning any interested DHIS2 user or developer has been able to download and use the application for the last 6 releases.

The application was also used as the main tool for the last data migration of the project DHIS2 instances. End users of WHO, UPC and external contractors used the application in a production environment with ease and provided helpful feedback.

The quality of the application allowed us to submit it to the “2019 DHIS2 Annual Conference” app competition.

Even though we were not selected as one of the three final finalists, the project stakeholders were able to showcase the application during the session WHO held in the conference[25].

The next steps for the application include improving the admin panel to browse existing blacklisting rules instead of accessing them by name, improve the logic for edge cases on complex metadata structures or simplifying the usage of the application by introducing a step-by-step wizard that allows any non-experienced user to benefit of the advanced features.

These improvements are discussed with more detail in the “Future work” section of this thesis.

[24] https://github.com/WISCENTD-UPC/Advanced-Metadata-Export
[25] https://www.dhis2academy.org/annual-conference-2019/

8.2. Bulk Load App

8.2.1. Original application

The Bulk Load application development began in 2016 by a former student during his master's thesis. He developed an initial version that implemented basic support to create and read excel templates from DHIS2 forms. Even though the application was enough for the project's needs back in that time, the stakeholders determined the app needed a refresh.

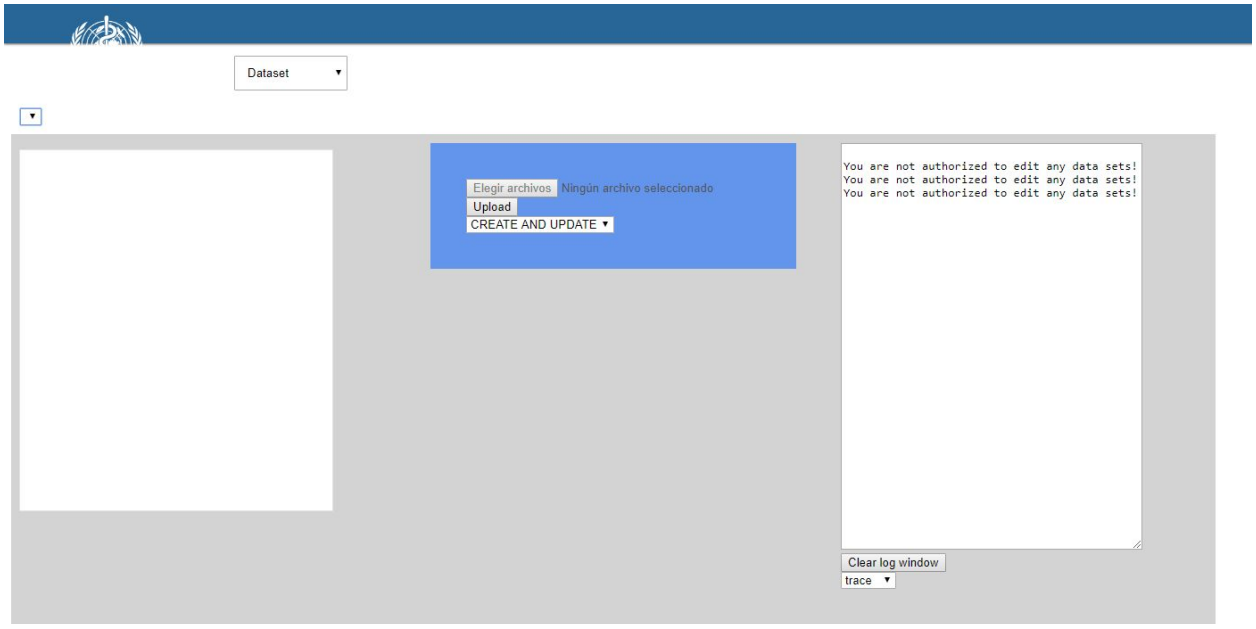


Figure 26. Original Bulk Load App developed in 2016

The original application relied on the standards from older DHIS2 versions and stopped working properly with recent updates to our instances. Since DHIS2 API endpoints have evolved dramatically in recent years, we determined a new application was needed.

Even though we had to re-write most of the code, we used the source available as a reference to determine the features we had to recreate in the new application.

Since the initial version was built using JQuery instead of React, we decided to re-use the same codebase we had created for the Advanced Export App. This simplifies the maintenance for both applications, as they share the same core and user interface.

8.2.2. Main objectives

The main objective for this application is to insert data into DHIS2 instances with offline excel spreadsheets. Since most of the users of the project live in third-world countries where Internet is unreliable or non-existent, we need to provide a way to collect datasets from Hospitals and field research institutions.

The application has two main features: building template spreadsheets and importing the data written in the previously created template spreadsheets.

In order to be useful, the application has to adapt to any DHIS2 form and build a standardized spreadsheet that mirrors the same look and feel. That includes creating disaggregation tables, allowing data validation for fixed value forms or reporting data from different regions and periods of time.

8.2.3. Technical overview

The bulk load app is also built in modern JavaScript using the React framework, in fact we based the development on a simplified version of the advanced export app. That means we also use the official “d2” and “d2-ui” libraries to communicate with the core back-end and maintain the look-and-feel of the instance where it is being executed.

We have three main services on the application: the connector that acts as a middleware to the current DHIS2 instance, the spreadsheet builder that creates the excel templates and the spreadsheet parser that reads and imports the data from the templates.

Since there is no official library that reads or writes excel files from JavaScript, we are using two different third party libraries to work with spreadsheets. For the template creation we use Excel4Node[26] and to read the data we use ExcelJS[27].

We are not tied to any excel management suite and if we find a better alternative, we could refactor the existing code and adapt to any other library. Also, the services that create or read the spreadsheets are independent and isolated from each other to support those changes.

[26] https://github.com/natergj/excel4node
[27] https://github.com/exceljs/exceljs

The service that handles the spreadsheet creation only exposes one method that from a given builder that contains the metadata of a DHIS2 form fills the different sheets of the Excel file and requests the browser to download the resulting file.

```
export const SheetBuilder = function(builder) {
    this.workbook = new Excel.Workbook();
    this.builder = builder;
    this.validations = new Map();

    this.dataEntrySheet = this.workbook.addWorksheet("Data Entry");
    this.legendSheet = this.workbook.addWorksheet("Legend", protectedSheet);
    this.validationSheet = this.workbook.addWorksheet("Validation", protectedSheet);
    this.metadataSheet = this.workbook.addWorksheet("Metadata", protectedSheet);

    this.fillDataEntrySheet();
    this.fillLegendSheet();
    this.fillValidationSheet();
    this.fillMetadataSheet();
};

SheetBuilder.prototype.fillDataEntrySheet = function() {...};

SheetBuilder.prototype.fillLegendSheet = function() {...};

SheetBuilder.prototype.fillValidationSheet = function() {...};

SheetBuilder.prototype.fillMetadataSheet = function() {...};

SheetBuilder.prototype.downloadSheet = async function() {...};
```

Figure 27. Bulk Load App methods for template creation

Even though the user only uses two of the sheets found in the excel file, we create four different sheets. The first one is where the user inserts the form data and the second one includes a legend that informs the user about the fields available in the DHIS2 form.

The last two sheets are protected and only used programmatically to implement the core functionality. The third sheet contains all the validation formulas used by excel and the last one includes the references and logic that allows the excel file to be imported back again into DHIS2.

Since most of the logic is done in the template building, the service that reads the spreadsheet is much more simple. It consists of two different methods, the first one goes row by row in the excel file and maps the values inserted with the form metadata. And the last one is a utility function that given an internal DHIS2 unique identifier returns the associated metadata.

```
/**
 * Import sheet information
 * @param builder:
 *      - d2: DHIS2 Library
 *      - file: File to be imported
 *      - element: Metadata from DHIS2 form
 * @returns {Promise<>}
 */
export function readSheet(builder) {...}

function parseMetadataId(metadataSheet, metadataName) {...}
```

Figure 28. Bulk Load App methods for spreadsheet parsing

Finally we have the last service that performs the metadata gathering from DHIS2 forms and performs the final data importing task. It consists in the following three public methods:

```
export function getUserInformation(builder) {...}

export function getElementMetadata(builder) {...}

export function importData(builder) {...}
```

Figure 29. Bulk Load App methods for back-end connection

The first one requests the user information, that contains a list of forms the user has access to. This allows the application to hide sensitive data that the user should not read or modify. The second one provides the metadata related to one of the forms and the last one requests the import task to the DHIS2 instance.

As you may have noticed, in both the Advanced Export and Bulk Load applications we have used the “Prototype Design Pattern”. JavaScript is an object oriented language that does not require classes. Instead of using classes you can build inheritance through a prototypical model.

8.2.4. Application features

Since the complexity of the application we introduced the first milestone release to the stakeholder when basic functionality was already implemented. DHIS2 supports two different kinds of forms: “Datasets” and “Programs”. Even though these two forms share the procedure to introduce data, they completely differ internally and include different features.

The users needed the application to provide support for both types of forms, so instead of working with the two of them at the same time we focused first on one type and later moved to include support for the other one.

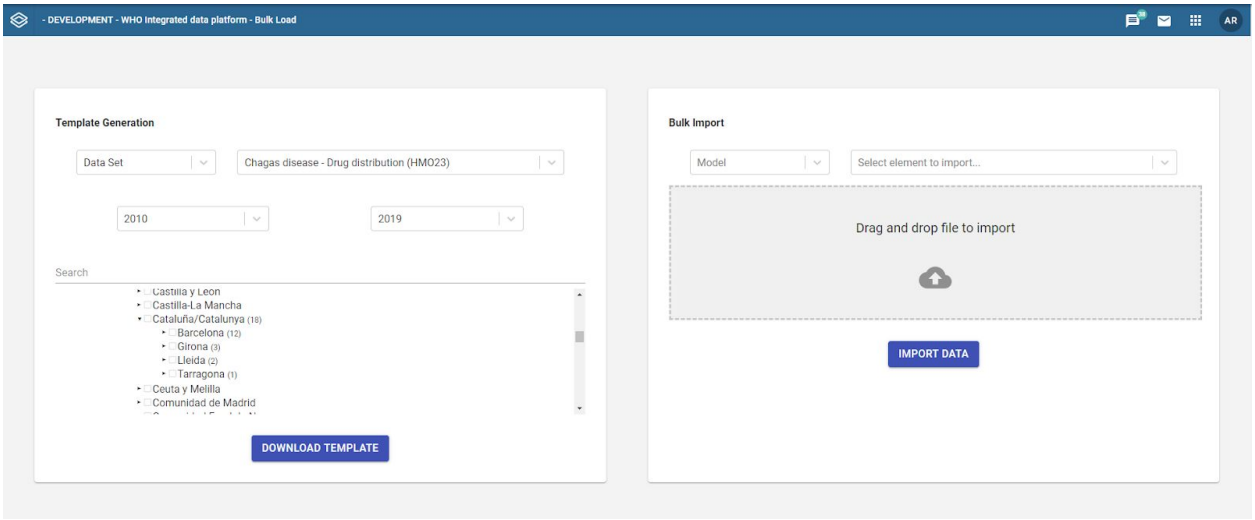


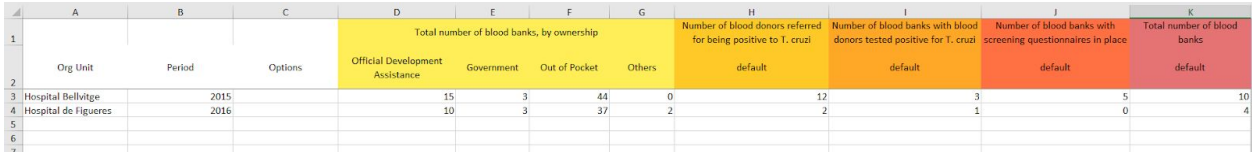
Figure 30. Bulk Load Application web front-end

The user experience consists of two different front-ends: the web application and the excel file the user interacts with. The web application is meant for system administrators and we decided to use a minimal interface that included all the available options.

In the left pane we added all the components required to generate an excel template spreadsheet. That includes filtering between “Datasets” and “Programs” and selecting the desired form to use as base. We also show a tree that allows the user to select the organisation units that the user will use to insert data[28].

[28] As explained in the previous chapters, an organisation unit is a hierarchical data structure. In this particular case we use the organisation units to select which medical institutions should introduce data.

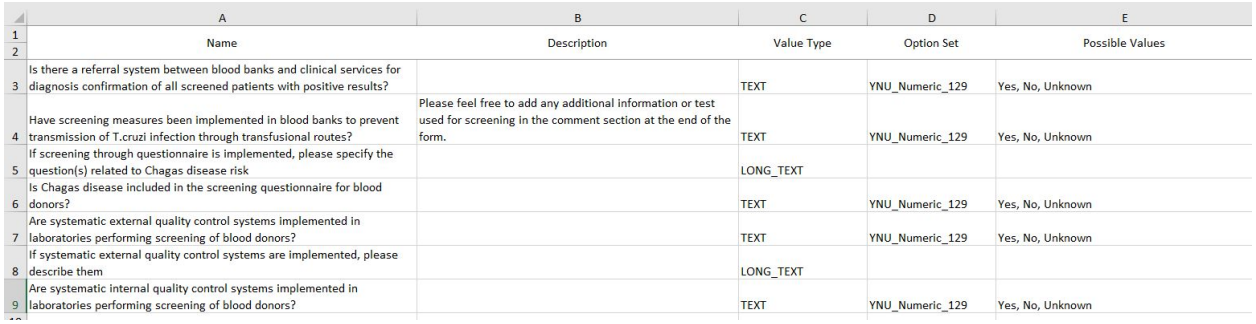
The second front-end we work with is the excel template the end-users will receive and fill with patient data or investigation results. We focused on building a user-friendly template that could be understood by any field investigator.



	A	B	C	D	E	F	G	H	I	J	K
1				Total number of blood banks, by ownership				Number of blood donors referred for being positive to T. cruzi	Number of blood banks with blood donors tested positive for T. cruzi	Number of blood banks with screening questionnaires in place	Total number of blood banks
2	Org Unit	Period	Options	Official Development Assistance	Government	Out of Pocket	Others	default	default	default	default
3	Hospital Bellvitge	2015		15	3	44	0	12	3	5	10
4	Hospital de Figueres	2016		10	3	37	2	2	1	0	4
5											
6											

Figure 31. Example data entry sheet for a dataset with disaggregation

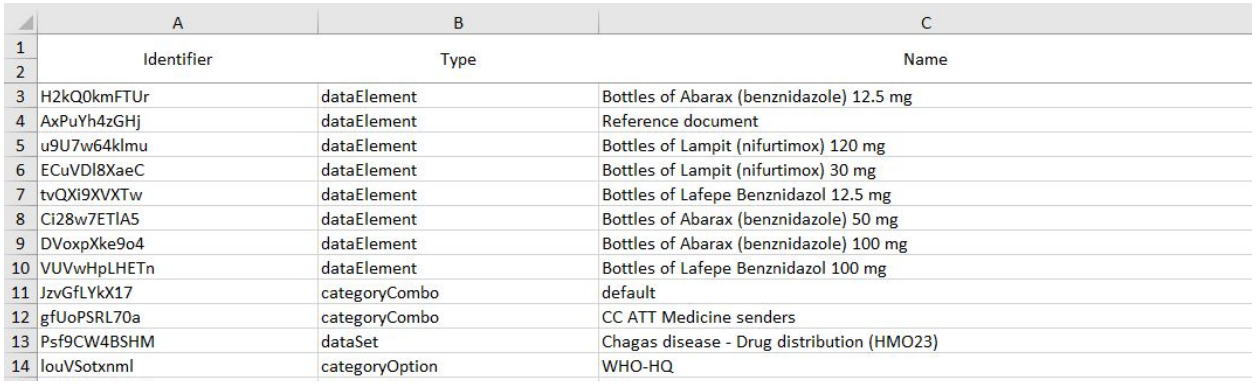
We also include a legend that explains the different form fields and showcases all the possibilities available for data input.



	A	B	C	D	E
1					
2	Name	Description	Value Type	Option Set	Possible Values
3	Is there a referral system between blood banks and clinical services for diagnosis confirmation of all screened patients with positive results?		TEXT	YNU_Numeric_129	Yes, No, Unknown
4	Have screening measures been implemented in blood banks to prevent transmission of T.cruzi infection through transfusional routes?	Please feel free to add any additional information or test used for screening in the comment section at the end of the form.	TEXT	YNU_Numeric_129	Yes, No, Unknown
5	If screening through questionnaire is implemented, please specify the question(s) related to Chagas disease risk		LONG_TEXT		
6	Is Chagas disease included in the screening questionnaire for blood donors?		TEXT	YNU_Numeric_129	Yes, No, Unknown
7	Are systematic external quality control systems implemented in laboratories performing screening of blood donors?		TEXT	YNU_Numeric_129	Yes, No, Unknown
8	If systematic external quality control systems are implemented, please describe them		LONG_TEXT		
9	Are systematic internal quality control systems implemented in laboratories performing screening of blood donors?		TEXT	YNU_Numeric_129	Yes, No, Unknown

Figure 32. Example legend sheet for a program with field validation

Finally we include the metadata with the internal DHIS2 instance unique identifiers to allow the cross-matching required for the data importing.



	A	B	C
1			
2	Identifier	Type	Name
3	H2kQ0kmFTUr	dataElement	Bottles of Abarax (benznidazole) 12.5 mg
4	AxPuYh4zGHj	dataElement	Reference document
5	u9U7w64klmu	dataElement	Bottles of Lampit (nifurtimox) 120 mg
6	ECuVDl8XaeC	dataElement	Bottles of Lampit (nifurtimox) 30 mg
7	tvQXi9XVXTw	dataElement	Bottles of Lafepe Benznidazol 12.5 mg
8	Ci28w7ETlA5	dataElement	Bottles of Abarax (benznidazole) 50 mg
9	DVoxpXke9o4	dataElement	Bottles of Abarax (benznidazole) 100 mg
10	VUVwHpLHETn	dataElement	Bottles of Lafepe Benznidazol 100 mg
11	JzvGfLYkX17	categoryCombo	default
12	gfUoPSRL70a	categoryCombo	CC ATT Medicine senders
13	Psf9CW4BSHM	dataSet	Chagas disease - Drug distribution (HMO23)
14	louVSotxnml	categoryOption	WHO-HQ

Figure 33. Example metadata sheet for a dataset

8.2.5. Conclusions

Even though the bulk load application has not been tested in production yet, we have provided the stakeholders with 6 different releases.

The main functionality from the original application is already available and we have improved the quality of both the web application and the template spreadsheets.

The original application lacked support for validation and the excel spreadsheet was mainly a comma separated value document.

Now we have the application ready to deploy excel templates to external medical institutions and receive data that the researchers can visualize and study.

Since version 2.1.0 (released the 4th of June 2019) we open sourced the application under the project GitHub organisation[29]. This means any user can benefit from the final application and install it on any DHIS2 instance.

The application has drawn interest from different organisations, for example “Médecins Sans Frontières” (MSF) has explicitly requested more information about the features implemented and they will likely start using it for their internal projects. We expect the same to happen with other companies and institutions.

During this thesis we have focused on mirroring the same functionality the old application had but we expect the application to receive new features in upcoming releases.

Some improvement ideas can also be found in the “Future work” section of this document.

[29] https://github.com/WISCENTD-UPC/Bulk-Load

8.3. Metadata Repository Script

8.3.1. Original scripts

The original script was one of the first applications developed by the UPC for the joint project with WHO. It was built using Java, a language that is not recommended by DHIS2 anymore, and it had several memory leaks that often ended in random application crashes.

It used Subversion to recreate a copy of the metadata found in the DHIS2 instance and had major performance issues when generating the first local clone. It also was designed to use older DHIS2 versions and does not handle metadata with the latest schema revisions.

8.3.2. Main objectives

The script main feature is to clone all the metadata into a local collection of JSON files, store them in a version control system such as Git and track the remote changes made in the metadata.

The script is based on two main parts:

- The first part of the script main purpose is to create independent copies of the metadata stored in DHIS2. Those independent copies are organized in a Git repository and we are able to differentiate what changes have been introduced over time.
- The second part of the script is to subscribe to metadata changes and update other DHIS2 instances with the local Git repository. That allows us to synchronize different instances in a regular basis with the latest changes following a master-slave pattern.

Since the original script already supported those functionalities we focused on migrating the existing logic to a language that is supported by DHIS2.

We chose JavaScript as the new language for the script to unify the languages we use in the project. That allows us to reuse code with ease between other applications such as the Advanced Export or the Bulk Load.

8.3.3. Technical overview

The Metadata Repository Script is built using NodeJS and uses the latest JavaScript standard. We created the script as a command line interface application. Even though we only support one command right now, it is expected to grow in the future with more features to manage metadata copies.

```
sfera@coruscant:/mnt/d/Workspace/UPC/Metadata-Repository-Service$ babel-node src/index.js help update

  Usage: update [options] <rule>

  Build and update a given rule

  Options:

    --help  output usage information
```

Figure 34. Usage help of the Metadata Repository Script

All the script configuration is exposed through a JSON compliant file, we support configuring the credentials for the remote DHIS2 instances, the remote Git repository credentials and the metadata to be fetched in the DHIS2 master instance.

```
{
  "rules": [
    {"name": "upc-dev-1"...},
    {"name": "upc-dev-2"...},
    {
      "name": "upc-dev-3",
      "debug": true,
      "originUrl": "http://who-dev.essi.upc.edu:8081/api",
      "originCredentials": {"username": "arico"...},
      "remotes": [...],
      "repo": "git@github.com:SferaDev/dhis230.git",
      "repoBranch": "upc-dev-3",
      "repoCredentials":{"publicKey": "/home/sfera/.ssh/id_rsa.pub"...},
      "commiter": {"name": "Alexis Rico"...},
      "metadata": [...]
    }
  ]
}
```

Figure 35. Configuration options of the Metadata Repository Script

8.3.4. Conclusions

This script was the last task introduced into the thesis and has suffered major drawbacks in the schedule. We mirrored most of the required functionality but it has not been used yet in production environments.

We fully support the first part of the original script which includes creating mirrors of the metadata stored in a master DHIS2 instance and detecting changes introduced to the JSON files. We also include functionality to push the local mirror to remote Git repositories such as GitHub.

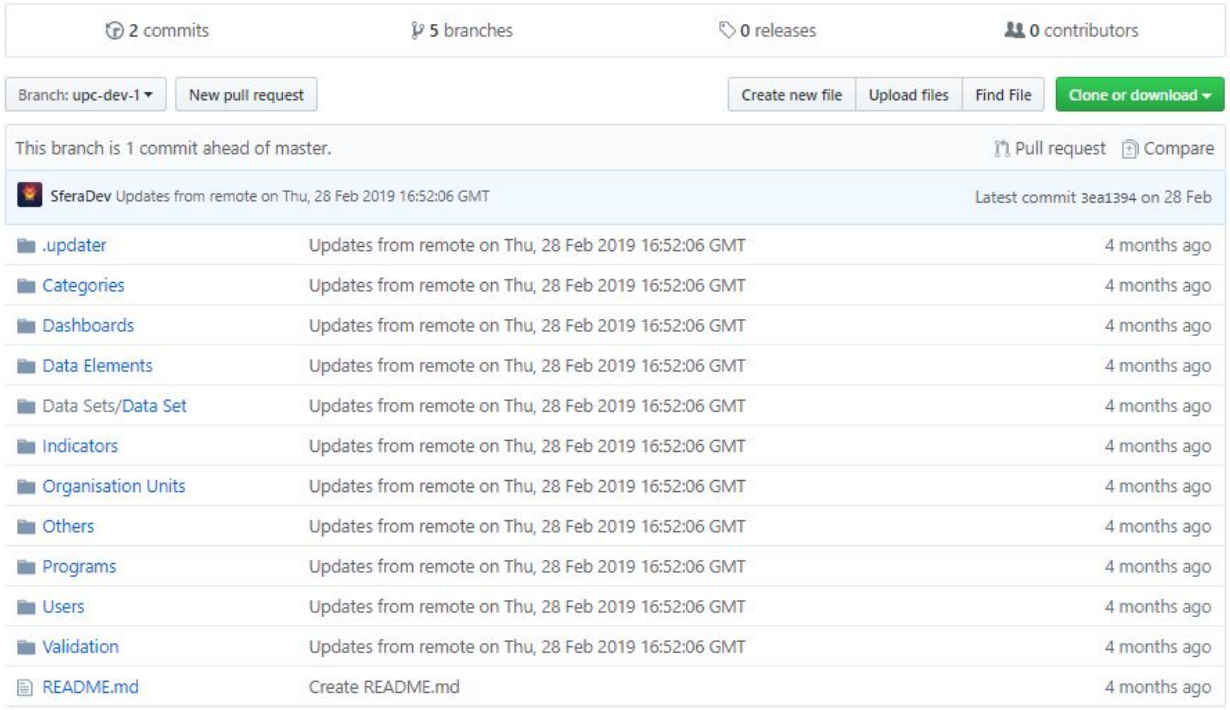


Figure 36. Remote GitHub repository pushed by the Metadata Repository Script

Due to unexpected requirement changes in the other two apps we did not have enough time to finish the second part of the script. That second part updates the slave servers with the changes of the master instance. More details about the implementation of those functionalities are detailed in the “Future work” section of this thesis.

8.4. Local development environment

Since the three applications we had to build are meant to be executed in a remote server, we had to design a working environment that allowed us to test and debug the applications locally.

The first problem we found is that a security feature in most browsers made our applications unusable while being on localhost. The "Cross-Origin Resource Sharing" (CORS) blacklisted all our network petitions to the remote DHIS2 instance. Luckily we are not the only ones that face this issue and there is a configuration property in DHIS2 that allows to whitelist select domains.

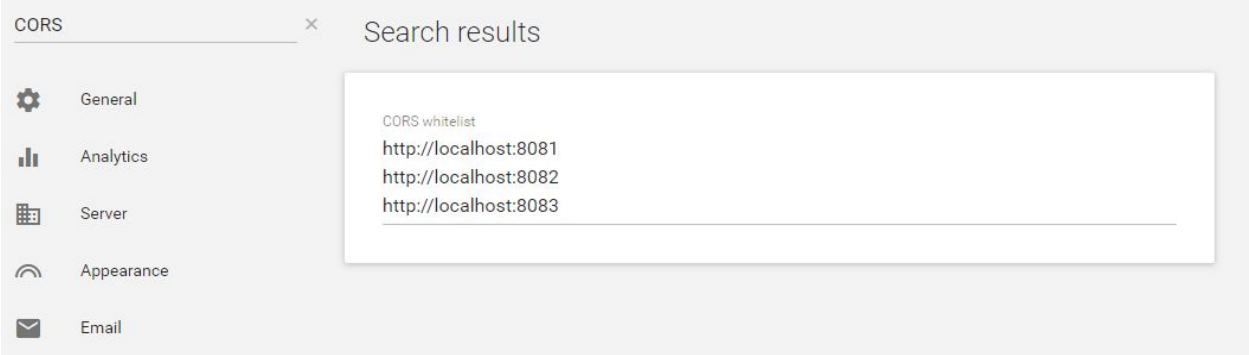


Figure 37. DHIS2 System Settings to whitelist CORS domains

The second issue we faced was to read the user credentials when deployed in the remote instance. The d2 library does not actually handle the authentication, instead you have to manually parse them and set them up during the library initialization.

We built a set of utility methods[30] that differentiated where the application was being run and either read the credentials from the environment variables or re-used the session cookies set by the DHIS2 instance.

Finally since we were working with public DHIS2 instances that other users relied on, we could not perform certain destructive actions that modified or deleted data. So we created[31] a dockerized local development server that exposes an empty DHIS2 instance. We can also seed the database with data obtained from the WHO and we can even open access to make it publicly available on the web.

[30] https://gist.github.com/SferaDev/8eb56a10d067b1a3e40613c8ead02f9a
[31] https://github.com/SferaDev/dhis2-backend

8.5. Application testing and requirement acceptance

Since the three applications generate artifacts that are difficult to test, such as JSON files from external metadata, excel files from external forms or git repositories, we did not create any end to end testing pipeline.

Instead of relying on automated tools that could ignore potential issues or detect false positives in the artifacts generated, we implemented a testing pipeline that included human interaction. During this testing pipeline we also included requirement acceptance with the stakeholders.

8.5.1. Development team

The development team, software engineer and UPC management, thoroughly tested the artifacts produced with each new version of the applications. With their knowledge of DHIS2, they verified if the artifacts were compliant with their specification and detected any newly introduced issues.

8.5.2. Field IT personnel

As soon as a milestone version was released, field IT personnel from WHO and the NTD department begin to use the new functionalities. They review the internal logic of the applications and review that the artifacts work properly in the production environments.

8.5.3. WHO Program managers

During the regular sprint retrospective meetings, the WHO program managers verify the new functionalities match the original requirements. They also test the application to review non-functional aspects of the application such as look and feel or user experience.

8.5.4. External DHIS2 users

When the projects are open sourced, any user from WHO or other organizations have the possibility to use and test the functionality in their specific domain. This serves as the final step in the testing pipeline.

9. Conclusions

9.1. Overview

The main objective of this final degree thesis was to integrate the student in the IT team that provides support to WHO while he obtained the requirements to develop new tools and applications the team required.

Even though the most important deadlines for the first application were met we have been forced to alter part of the initial planning. This was already expected due to the agile and constantly evolving nature of the project.

For example, we received feedback from our end-user testing team that required us to diverge from the planned tasks and implement the new, and unaccounted, functionalities the stakeholders needed.

To address those delays we have reduced the time planned for documentation and prioritized the most important features instead of applying all the original ideas we had. In any case, we have dedicated more than the estimated hour dedication for TFGs.

In any case, the thesis had three major problems to solve, the packaging of bundled metadata, bulk importing of data and version tracking of metadata changes. We determined that the best solutions to attack those three problems was by creating three different applications and we have successfully created those three applications.

Also the student has worked on at least four other applications that were not part of the thesis but that are critical for the project. As an example, he added some new functionality to an application co-developed by the UPC and the “Health Management Information System” (HMIS).

Finally, thanks to the thesis the student gained enough understanding about the internals of DHIS2 and he is now working as a private consultant for EyeSeeTea. EyeSeeTea is a leading consultancy company for DHIS2 systems and the student got to know them thanks to different collaborations during this project.

9.2. Objective fulfillment

To review the dedication of the student towards the thesis and the project where he has been working for over a year and a half, we would like to review the initial objectives for the thesis.

Due to the agile methodology of the project, new objectives have appeared since the beginning of the collaboration.

- **O1. Create a new application to Export Metadata**
  - O1.1. Allow the user to select any metadata of the running DHIS2 instance
  - O1.2. Recursively fetch metadata dependencies
  - O1.3. Create importable metadata packages in JSON format
  - O1.4. Remove properties that can not be shared between DHIS2 instances
  - O1.5. Allow creating rules to filter dependency fetching
  - O1.6. Store filtering rules in a shared data store
- **O2. Create a new application to Bulk Import Data**
  - O2.1. Mirror the capabilities of the old “Bulk Data Upload” application
  - O2.2. Allow the user to create an excel template for datasets and programs
  - O2.3. Allow the user to import a previously created template
  - O2.4. Build user friendly excel files that include a legend with usage instructions
  - O2.5. Validate values of dropdown lists and add conditional formatting
  - O2.6. Add period selection and dataset options to the excel file
  - O2.7. Protect sensitive sheets of the template excel file
- **O3. Create new scripts to handle Metadata Synchronization**
  - O3.1. Traverse the entire metadata collection of a master DHIS2 instance
  - O3.2. Track changes to the metadata in a remote git repository
  - O3.3. Allow configuration for multiple synchronizations through a JSON file
  - O3.4. Update slave DHIS2 instances with the changes detected to master
- **O4. Add new features to other applications of the organization**
  - O4.1. Allow pagination in the “Data Collection Reminder” application
  - O4.2. Refactor the “Data Sending” application to run with latest DHIS2 versions
  - O4.3. Fix existing issues with the “HMIS Dictionary” application

Figure 38. Final objectives of the thesis

9.3. Planning deviations

Overall we have respected the estimated hour assignment that we initially set for the thesis development. But as we included new objectives, we also have seen an increment in the real hours that have been finally dedicated on development and testing tasks.

TASK	ESTIMATED HOURS	REAL HOURS
Initial Planning and Scope	30.00	30.00
Viability analysis	60.00	55.00
Advanced Metadata Export	245.00	**325.00**
Initial design and set-up	20.00	20.00
Feature development	150.00	**205.00**
Integration test with users	60.00	**90.00**
Final documentation	15.00	10.00
Bulk Load	220.00	**290.00**
Initial design and set-up	15.00	15.00
Feature development	130.00	**160.00**
Integration test with users	60.00	**100.00**
Final documentation	15.00	15.00
Metadata Synchronization	180.00	75.00
Initial design and set-up	10.00	10.00
Feature development	120.00	50.00
Integration test with users	40.00	10.00
Final documentation	10.00	5.00
Other applications	0.00	**55.00**
Total development	**735.00**	**830.00**
Product Management	**120.00**	**140.00**

Figure 39. Final task planning

9.4. Future work

9.4.1. Synchronization wizard in the Advanced Export App

As we have seen in section 7.1.4 of this thesis, the stakeholders have found out that the usage of the application was too technical and that required additional training to use the advanced features.

This wizard could be planned for the next big release (0.3.0) of the application and we would separate the metadata and dependencies selection, allowing the user to limit the number of dependencies to be exported.

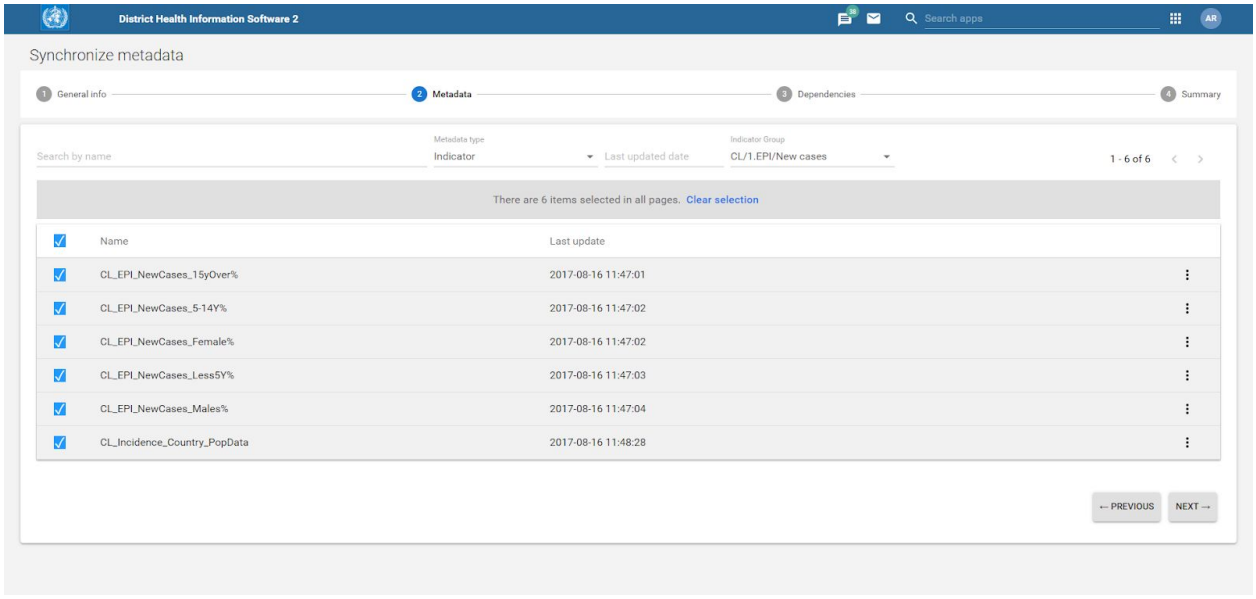


Figure 40. Advanced Export App's new Wizard mock-up

9.4.2. Dependency blacklisting in the Advanced Export App

The current way to exclude dependencies is through a blacklist that is available on the "Admin menu". But we do not currently support circular blacklisting of metadata dependencies.

This functionality was not originally accounted and we expect future versions of the application to improve the whole "Admin menu" to make it even more powerful and user friendly.

9.4.3. Tracker programs in the Bulk Load App

DHIS2 allows different types of forms, in chapter 8.2.4 of this thesis, we explained that the two main forms we use in this project are “datasets” and “programs.

However there is a special type of program that allows registration and follow-up. It is called “tracker program” and we do not currently support that kind of program. This is not a blocking issue, since WHO does not use that kind of forms for the neglected tropical diseases.

But since the application is now open sourced and available to any user, we should allow this kind of forms in the Bulk Load App to be fully compliant with the user’s needs.

9.4.4. Advanced validation in the Bulk Load App

In the excel templates that we generate through the bulk load application we currently support validation of collection items (such as list dropdowns) but we do not verify that the dates or numbers the user is inserting follow the DHIS2 standard format.

We have detected some issues with the application in some DHIS2 instances of Nepal because they use a different calendar system and we should also try to fix this behaviour in future releases of the application.

9.4.5. Slave Synchronization in the Metadata Repository Script

As we mentioned in chapter 8.3.4 of this thesis, we had not enough time to finish the second part of the script. That second part pushes the detected changes to slave DHIS2 instances and acts as a message broker.

We have partial support in the code and it’s almost finished, anyway before pushing it to production additional testing is needed. We do not find this issue to be blocking because we still have the original metadata synchronization scripts that are updating the slave instances regularly.

References

WHO NTD Official Page

https://www.who.int/neglected_diseases/en

WHO WISCENTD Official Page

https://www.who.int/neglected_diseases/disease_management/wiscentds/en

ESSI DTIM WISCENTD Official Page

http://www.essi.upc.edu/dtim/projects/wiscentd

ESSI DTIM WIDP Official Page

http://www.essi.upc.edu/dtim/projects/widp

Performing data lineage for an ingestion system to a data lake (Garnica Caparrós, Marc)

https://upcommons.upc.edu/handle/2117/99453

Parametrization of the chagas disease surveillance system for the WHO (Mourin Marin, Eric)

https://upcommons.upc.edu/handle/2117/100881