UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Facultad de Informática de Barcelona

Bachelor's degree in Informatics Engineering

Major in Computing

Development of the web app to enable interoperability and automated data exchange for specialised outbreak tools in developing countries.

June, 2022

Author: ***Antoni Bergas Galmés***

Director: Petar Jovanovic

Database Technologies and Information Management Group (DTIM)

Abstract

This project focuses on the development of a data interoperability web app for WHO's outbreak investigation tool Go.Data in developing countries, in the context of disease outbreaks or pandemics (e.g., COVID19 or Ebola). The app will operate within the DHIS2 platform, and it should provide dynamic metadata synchronisation between DHIS2 and Go.Data, as well as individual and aggregated data exchange. This project mainly involves the use of MongoDB, JavaScript with ReactJs Framework and Node.js.

Este proyecto se centra en el desarrollo de una aplicación web de interoperabilidad de datos para la herramienta de investigación de brotes Go.Data de la OMS en países en desarrollo, en el contexto de brotes de enfermedades o pandemias (por ejemplo, COVID19 o Ébola). La aplicación operará dentro de la plataforma DHIS2 y debería proporcionar sincronización dinámica de metadatos entre DHIS2 y Go.Data, así como intercambio de datos individuales y agregados. Este proyecto involucra principalmente el uso de MongoDB, JavaScript con ReactJs Framework y Node.js.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and context

1.1 Introduction

Throughout history, humanity has been the victim of different epidemics, which are diseases that infect a large number of people in a short period of time. The most relevant epidemic in human history was the black plague[7], which claimed the lives of between 25 and 50 percent of Europe's population by the fourteenth century. Recently, in 2019, surfaced COVID-19, a pandemic that has already killed about 6 million people worldwide[8].

Due to the lack of technology and means of communication, there is little record of where the Black Death affected the most and there was not much monitoring of the outbreaks that were appearing. These references were very disparate, not having been made for demographic purposes but for tax purposes and being in a large number of different languages[7].

Nowadays, when dealing with any type of pandemic, it is considered essential to record as much data as possible on the disease and contagions for a control of the outbreaks, analysis of its evolution, the possible variants of the disease, its spread, etc, in order to reduce as much as possible the negative impact that it may cause to a community.

Figure 1.1: In May 2020, medical personnel in So Paulo treat a COVID-19 patient in critical condition in an ICU.[2]

The World Health Organization (WHO), a specialized agency of the United Nations responsible for public health, among its various functions, plays a very important role in the collection, storage, analysis, monitoring and management of data on epidemic diseases specially in developing countries. To carry out these objectives, the WHO uses its own tool called Go.Data. A goal to highlight of the Sustainable Development Goals (SDG) that WHO intends to carry out, we highlight the third one; *Ensure healthy lives and promote well-being for all at all ages.*

This Degree Final Project aims to continue the development of the web app to enable interoperability and automated data exchange for specialised outbreak tools in developing countries.

1.1.1 Context

The Degree Final Project project is done within the framework of the Facultad de Informática de Barcelona (FIB), in the progress of a part of an international development cooperation project called *Strengthening countries in adopting specialized outbreak management software suite (Go. Data) by enabling interoperability, promoting secure data exchange and automation*[1], together with the Database Technologies and Information Management (DTIM) research group, led by Prof. Petar Jovanovic with the collaboration of the World Health Organization (WHO).

This project is financed by the UPC's Centre for Development Cooperation (CCD) and all the related information of the complete project is available at its website framed in: Form modality A; Axes of the UPC plan; Typology 1: International development cooperation projects.

[1]URL of the project: https://www.essi.upc.edu/dtim/projects/COVID-19

1.1.2 Concepts

In this section, concepts, entities and platforms are explained for a better understanding of the current situation and the problem.

World Health Organization (WHO), GOARN and Go.Data

The World Health Organization (WHO) is a specialized agency of the United Nations founded in 1948, which aims to achieve the highest level of health for all people[9].

The Global Outbreak Alert and Response Network (GOARN) is a network of technical and public health institutions, labs, non-governmental organizations (NGOs), and other organizations that strive to monitor and respond to epidemic threats. One of GOARN's most notable partners is the World Health Organization (WHO), with whom it collaborates closely. Its objectives are to investigate and analyze diseases, assess the risks that certain diseases pose, and increase worldwide disease response capabilities[10][11].

Go.Data is a specialized outbreak management tool, developed and maintained by the GOARN[10] of the World Health Organization (WHO), and currently in the process of adoption and implementation in various countries. In the moment of pandemics or local disease outbreaks, countries should be able to easily exchange outbreak specific data with their day-to-day data management tools, so that they have a more complete picture of their health system, its current saturation, and plan future steps in fighting the outbreak.

Figure 1.2: The implementation of Go.Data as a tool to support contact tracing and surveillance of COVID-19 in Argentina[3]

District Health Information Software (DHIS2)

DHIS2 (District Health Information Software) is utilized in more than 60 nations worldwide. The Health Information Systems Programme developed DHIS2, an open source software platform for reporting, analysis, and dissemination of data for all health initiatives (HISP)[4].

The Department of Informatics at the University of Oslo coordinates the DHIS2 platform's primary development operations, which are supported by NORAD, PEPFAR, the Global Fund to Fight AIDS, Tuberculosis, and Malaria, UNICEF, and the University of Oslo[4].

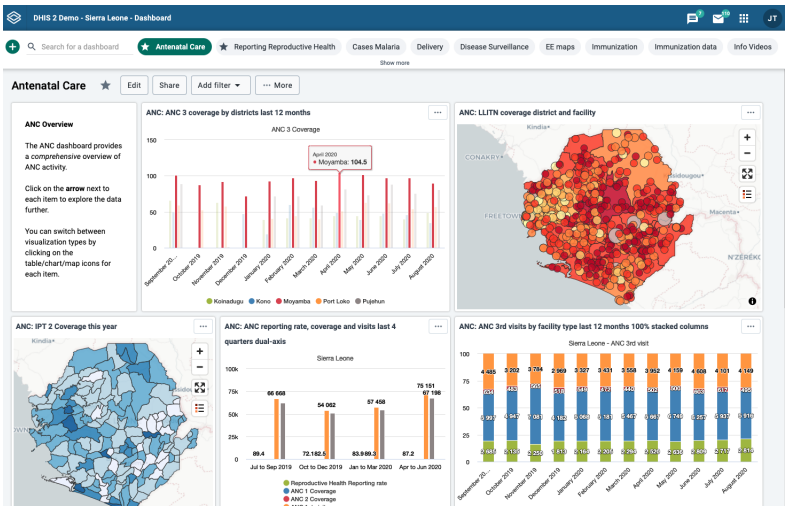

Figure 1.3: DHIS2: Dashboard [4].

Metadata and data in the context of the DHIS2 and Go.Data platform

Some concepts related to the DHIS2 and Go.Data **metadata** that will be necessary in order to understand the objectives and tasks that will be carried out throughout this project are collected and explained below.

- **Organization units**: Is the metadata that refers to the administrative unit levels where the data can be reported and it is also used to assign teams for contact tracing and follow-up. It is built hierarchically so that each 'Organization unit' can be subdivided into more organisation units at a lower level. This concept is called **'Locations'** within the Go.Data platform context.
- **Programs**: The programs in DHIS2 are groups of information fields in order to homogenize the information that is collected from each entity. A program contains a selection of attributes (Name, age...), it can contain different program stages to which the entity may or may not be linked, and it includes the locations associated with the program. For example, in the case of a program to collect cases of COVID-19, it contains program stages such as *Lab Results* and *Health Outcome*, it contains up to 14 different attributes and is associated with the Organization Units that are selected (which are in the instance). A program can be of the type *Tracker Program* or *Event Program* depending on whether or not specific information about the entity is saved, respectively (for instance, the name attribute would be specific information about the entity).

From the information in a program, the information needed to create an **Outbreak** in Go.Data can be obtained. With the difference that there can be many outbreaks for each program created in DHIS2. In Go.Data, Event programs are not considered either, since data is always associated with the entity and it contemplates data processing as a whole, as DHIS2 does.

- **Program Stage**: Continuing with the example of the COVID-19 program; Depending of the at which the case is being registered, some data or others will be collected. For example, data on whether or not the patient has survived the disease cannot be collected if they have just been infected, and laboratory results cannot be recorded if they have not yet been performed. Therefore, each case may or may not be linked to the program stages of the program depending on the state in which it is found.

  Each program stage has a set of **data elements** assigned to it, for example, the program stage *Lab Results* contains the data elements *Lab Test Results* or *Lab Date Test Results* among others.

  In Go.Data the program stages are not contemplated.

The **data** types with which DHIS2 and Go.Data will work are explained below.

- **Case**: The most basic data type that is collected in both DHIS2 and Go.Data platforms.
- **Contact**: In the case of DHIS2, a contact belongs to a program other than the one where the cases are collected, in the case of COVID-19 there would be the *COVID-19 Case-based Surveillance* and *COVID-19 Contact Registration and Follow-up* and a relationship between the two programs would be defined. In the case of Go.Data, the workflow is completely different, the cases are associated with an outbreak and the contacts with the case. There is also the **Contact of Contact** data type in both Go.Data and DHIS2 platforms.
- **Event**: The event is a case in which the entity's personal information is not collected. An event is only seen on the DHIS2 platform.

1.1.3 Context of UPC and WHO collaborations

There exists a long running collaboration of the UPC team with the WHO. More specifically, with the department of Neglected Tropical Diseases. Since April 2020, this collaboration has been extended to GOARN and Go.Data development team under the CCD project (CODE 2020-V011). During the COVID-19

project, the initial results have been shown in terms of an interoperability app for exchanging data and metadata between Go.Data and DHIS2, and a security extension allowing secure transfer of data from a Go.Data instance. In this project, we aim at enhancing the previous results, and extend and generalize allowing interoperability of Go.Data, and assisting countries in deploying such components and allowing smoother adoption of the Go.Data tool.

1.1.4 Identification of the current situation and problems

Currently, the counties are using various data platforms for collecting and analyzing health data. Many countries in the Global South (e.g., Africa, South-East Asia and Western Pacific regions) are using the DHIS2 platform as surveillance tool for various diseases (HIV, Malaria, TB, Neglected Tropical Diseases, Non-communicable diseases, etc.), and even as the national health information systems in some cases. Although these platforms excel at providing strong support for disease surveillance, and storing and sharing health related data, they lack at supporting outbreak management actions, like contact tracing, visualizing transmission networks, and in general, controlling the outbreak. In addition, besides WHO-recommended information, countries are very often establishing other set of specific variables and indicators to be collected and monitored.

GOARN through WHO offices or Ministries of Health in the countries, are assisting in deploying their outbreak management tool Go.Data. However, introducing additional tools in countries without many technological and personal resources increases the risk that the adoption of these tools either fails or they are not used up to the fullest potential. Therefore, the main problem that this project deals with is enabling smooth adoption of the Go.Data tool by providing interoperability with the countries' health information systems, support secure exchange and sharing of outbreak data, and assisting countries in the deployment and training activities. Special attention is also given to the complexity of a diverse set of variables collected in each country, thus requiring easy and effective configuration of the developed components.

1.1.5 Beneficiaries of the project (direct and indirect)

The direct beneficiaries of this project are WHO personnel in charge of implementing their outbreak management tool Go.Data, through the Global Outbreak Alert and Response Network (GOARN).

Indirect beneficiaries are the Global South (as well as other interested countries) that are adopting Go.Data as the outbreak management tool, in order to better manage local disease outbreaks or pandemics. By enabling seamless exchange of data with their day-to-day health platforms, the countries will have

timely information on the health system saturation and could plan next steps in the fight against the burning health issues.

Finally, the indirect, but the most relevant beneficiaries are citizens of such countries that due to the faster reaction and containment of the outbreaks have better chance of avoiding contracting the disease and lastly have better quality healthcare available.

1.2 Analysis of existing solutions

In this section, two previously created apps to solve this problem will be analyzed. Both apps use node.js (an open source development platform for executing JavaScript code server-side)[12] and the React Framework (a declarative, efficient, and flexible JavaScript library for building user interfaces)[13]

1.2.1 Web App: *Go.Data - DHIS2 Interoperability App*

This web app; *Go.Data - DHIS2 Interoperability App* [14], available on Github, was developed in the past by students (within the COVID-19 project) of the UPC. At figure 1.4 it can be seen a schema of how its workflow is. It can be installed on the DHIS2 platform and offers several use cases to the user in order to sync data across Go.Data and DHIS2 platforms to solve the problematic listed above. The use cases are:

1. Send metadata: Create Go.Data outbreaks from the DHIS2 program info, send the organization units hierarchy successfully parsed.
2. Send registered cases/contacts in DHIS2 to Go.Data, for further detailed case investigation and contact tracing, or purely to visualize the relationships more easily. The option sets of the data elements are also sent and the cases are successfully linked at the Go.Data instance after the data transference.
3. Send cases/contacts from Go.Data to DHIS2.

This solution is composed of different parts that can operate independently:

- API wrapper for each platform
- Node.js script to execute and test the use cases directly from the console (Using the API wrappers)

- React framework app that uses the Node.js script to use it directly from the DHIS2 platform.

Figure 1.4: Workflow of the *Go.Data - DHIS2 Interoperability App* [5]

Problems of this solution

Although it solves the problem, it presents a lack of scalability due to the way it is configured before starting a data transfer, specifically, it presents problems when the data elements of the program stages have to be mapped. In fact, the development of the App focused on a specific DHIS2 program and specific use case for an instance at Sierra Leone and would require a total restructuring of the logic to be able to use the App transversally with any program.

It also has some optimization issues when submitting cases from DHIS2 to Go.Data which are prohibitive for some internet browsers and some computers (especially those in developing countries) and stability issues due to problems with a huge chain of requests to the DHIS2 server during the process.

Finally, the interface is not optimal and a more accurate report of errors is necessary when there is an error in a data transference.

1.2.2 Web App: *godata-dhis2-interop*

Recently, from the WHO team, developed by the hand of Murod Latifov a extension of the problem prevously defined providing greater flexibility when exchanging data. [15] Unlike the previous app, it seems to be more flexible at the configuration level and contains some very useful automatisms for detecting

the schemes used in both Go.Data and DHIS2 platforms to exchange the data.

1.2.3 *Data Mapping* template strategy

An interesting point of this App is the way in which they solve the main questions of the data conversion algorithm:

1. Where do I get the data from?
2. Where do I put the data?
3. In what format do I put the data?
4. Do the values have to undergo any transformation? And the keys?
5. Are there constant values with constant keys?

All the answers to all these questions are gathered and saved in JSON format and the user is given the possibility to create and modify this JSON.

Although the implementation of this strategy presents at first sight serious problems of performance, security and user experience. This strategy: the data structure, the dot notation usage and the concept may be strengthened throughout the development to meet our goals.

Problems of this solution

This solution has a more difficult code to understand and a less modular design than the previous app. Also, it does not represent a complete interoperability between the different platforms since it only has the transmission of data from DHIS2 to Go.Data. Another factor that could be improved would be the suboptimal use of the database to store certain configuration data, replacing it with the option recommended by DHIS2: Data Store[1]. (Which is used in the previous app)

In terms of usability, this app has serious problems since it is very difficult for the end user to understand, since it shows a JSON that can be edited almost freely but with very illogical rules and meaningless keywords.

The performance and efficiency is not very good, there are continuous bugs and some inexplicable screen freezes.

1.2.4 Contrast of both apps

The table 1.1 summarizes the advantages of both apps.

Go.Data - DHIS2 Interoperability App	***godata-dhis2-interop App***
Modular design	Scalability
Cleaner code and more documentation	Mappings detection automatism
Optimal use of storage	Better User Interface
More use cases	

Table 1.1: Table of the advantages of each app. [Own elaboration]

1.3 Justification and project scope

Once the existing solutions have been analyzed, it is concluded that it is not necessary to reinvent the wheel since there is code from both apps that can be used to build a solution that meets much of the needs presented.

1.3.1 Objectives

With this project we intend to continue with the development of this app, evaluating the different parts of the already created apps and integrating the parts that are concluded that work best.

However, it will inevitably be necessary to improve some parts of the code and develop algorithmic strategies that can lead to its resolution and implement the one that guarantees the best performance according to the established requirements.

In addition, user interfaces should be analyzed in depth, their problems identified and an interactive system developed properly given the circumstances.

Overall, the final objective will be to generate an app with all the technical pretensions (especially enabling the exchange of data, metadata and geographical hierarchies between Go.Data and the DHIS2 platform) and with a good UI so that it can be used intuitively by end users.

The objectives that the app has to satisfy can be broken down with the following list:

1. Enable the exchange of metadata between Go.Data and the DHIS2 platform.

   (a) Geographical hierarchies

2. Enable the exchange of data between Go.Data and the DHIS2 platform.
   (a) Exchange of outbreaks
   (b) Exchange of cases
   (c) Exchange of contacts
3. Improve User Interface
   (a) Redesign of the critical parts using the recommended stylesheet from DHIS2.
   (b) Feedback analysis from the final user
4. Improve algorithms efficiency
   (a) Analysis of the current algorithms used to parse and send/receive data
   (b) Implement possible improvements

1.3.2 Organization

In order to accomplish the objectives, a weekly follow-up will be done with the supervisor together with a Gantt diagram that will mark the times of the tasks. In addition, weekly meetings will be held with the WHO team to obtain their feedback on their aspirations and inform them of the current status of the project.

As this is an urgent project for the WHO, depending on its aspirations or changes in priorities, a certain flexibility will be needed to gradually change the approach or order of execution of the plan that is established.

Apart from this, to validate that the objectives are achieved, there are some dates indicated in which the product is presented to end users. These events will be of great importance in order to evaluate some aspects of the app, such as whether the user interface is intuitive.

Meanwhile, all progress will be updated in a repository of the WHO Information System to Control/Eliminate Neglected Tropical Diseases (WISCENTD) organization on Github.

1.3.3 Possible risks

Throughout the development of the thesis there are some potential risks and obstacles that we may have to deal with.

- **Bugs found in the currently implemented algorithms.** It can be caused by the update of some data schemes that cause incorrect parsing of the data.
- **Early deadline of the project.** It can be caused by an incorrect assessment of the duties and their duration.
- **Bugs in some libraries or difficulties understanding them.** Technical challenges may occur in these kind of initiatives that we were not expecting.
- **WHO or country priorities change**. Depending on the epidemiological situation some countries may require fast solutions for different problems.

Further information about the risks and their possible alternatives can be found in section 2.7.

Chapter 2

Planning

2.1 Planning

In order to finish this Degree Final Project on the estimated date, and meet the objectives previously set, in this section it is carried out a temporary planning of the project divided into tasks.

The project begins on February 14, 2022 and its completion is scheduled for May 27, 2022. In total, the development of the project would take place carried out over 102 days and with an estimated duration of 300 hours taking into account an approximate work of 20 hours per week on average.

The daily dedication will be approximately 3 hours, from Monday to Friday in the afternoon, and development tasks and follow-up meetings will be carried out entirely remotely.

2.2 Task definition

Following, it is presented all the tasks that will be carried out along the project. For each one, it is given a description, duration and dependencies with the other tasks. The project management is probably one of the most important group of tasks for the project. It defines the scope of it, the tasks and plans its distribution. Below are shown the multiple tasks for the project management.

- **Context and scope.** We have to indicate the general objective(s) of the project, contextualize it and justify the reason for selecting this subject

area.

- **Project planning.** To achieve the project deadline we need a good planning for all the tasks. This will help us to know in which tasks we have to focus on more and which are the critical ones.
- **Budget and sustainability.** When doing a project it is very important to know what will be the total cost of it and the impact that will produce its development. Hence, this task focuses on making a budget and analyzing the sustainability of the project.
- **Final project definition.** We have to group the project done in the previous tasks, modifying the parts that were wrong.
- **Meetings.** Face-to-face appointments with the project's tutor and another one with the WHO team are scheduled every week. We'll talk about the current situation and the next steps to take. Due to the possibility of unexpected meetings, we have extended time to the schedule.

This project has a big part of reading and understanding code written by somebody else. Hence, before starting to program the missing parts of the current software or solving some problems its essential to understand the code, explore the libraries used and to take a few notes about the initial workflow. This part is divided in three tasks:

- **List, download and classify** all the resources that may be useful throughout the project, be it documentation of the various API's, libraries or repositories of interest.
- **Study** the collected documentation.
- **Install** all the software required.
- **Test the correct functioning** of implemented use cases, document the results and the missing cases.

Objective 2 (Section 1.3.1: *Enable the exchange of data between Go.Data and the DHIS2 platform.*) corresponding to the exchange of data between platforms is the most important part in this project and will have a higher priority when allocating the available time. It has three different tasks:

- **Program the cases exchange** from Go.Data to DHIS2.
- **Program the contacts exchange** from Go.Data to DHIS2.
- **Design and program** the an efficient converter algorithm to parse the data before sending it to DHIS2.

Following the proposed objectives, a not insignificant part is to program a good intuitive user interface that does not cause lag. To do this, we will add the last two tasks to this list:

- **Design the UI**, especially the critical parts, taking the most advantage of the already created software by using the recommended stylesheets from DHIS2.
- **Analyze the programmed algorithms** and the already implemented ones and draw conclusions.

2.3 Resources

Every project needs resources to be able to organize it properly and carry out its correct development. These resources have been divided in 4 different groups: human, hardware, software and material resources.

2.4 Human recources

Several human resources can be found in this project. To begin, the developer is in charge of the project's development. He will be responsible for planning, experimenting, analyzing, and documenting the project. The project instructor, on the other hand, is in charge of leading and guiding the researcher through the project's progress. The WHO team, on the other side, will direct the project and submit their goals. Finally, during the first month of the project, the GEP instructor is responsible for assisting the researcher in properly managing the study.

2.5 Hardware resources

One of the essential resources needed is a computer. In this project it will be used a laptop computer with the following components:

- 8GB of RAM
- Intel(R) Core(TM) i7 6700HQ CPU @ 2.60GHz with 8 kernels.

Moreover, we also have to take into account all the resources for the connection to the network (e.g. the router). Occasionally, department servers are also used

to perform a test in a more realistic environment and to share it more easily with external people (WHO).

2.6 Software resources

We need multiple software resources. Each one will help us in a specific part of the project. To be able to manage the meetings we will use *Google Calendar*, to meet the tutor we will use *Google Meet* and to meet the WHO team it has been decided to use *Microsoft Teams*. All the information for the project will be saved in several *GitHub* repositories. The Gantt chart will be created using *Ganttproject*. We will need some programming language(s) to code the whole project including the Linux knowledge create the environment. Besides, we will use Overleaf as our text editor and *Visual Studio Code* as our code editor.

2.7 Risk management: alternative plans

During the course of the project there may appear obstacles that hinder its progress. The potential risks and obstacles have been introduced in Section 3.3. In this section, we will present how can they be solved by introducing new tasks and rearranging the planning. Moreover, we say for each one which is the level of risk cataloging with the values from 1 to 5 being 5 the one with the highest risk.

- **Bugs found in the currently implemented algorithms.** It can be caused by the update of some data schemes that cause incorrect parsing of the data. This is also a very probable risk to face and it has to be reserved a space of time for the fix of the algorithms. Estimated overtime: **20 hours**. *Risk level: 5.*
- **Early deadline of the project.** It can be caused by an incorrect assessment of the duties and their duration. It's extremely likely that it will happen because we're conducting the planning before we do anything. This can be remedied by performing a second planning phase at a later stage of the project. We'd have to repurpose the computer, Ganttproject software, calculator, GEP tutor, and researcher. Estimated overtime: **20 hours**. *Risk level: 3.*
- **Bugs in some libraries or difficulties understanding them.** Technical challenges may occur in these kind of initiatives that we were not expecting. When programming, it is fairly usual to use third-party libraries. Some library functions may not function properly owing to an

ID	Name	Time (h)	Dependences
T1	Project Management	85	
T1.1	Context and scope	20	
T1.2	Project planning	10	
T1.3	Budget and sustainability	15	T1.2
T1.4	Final project definition	20	T1.1,T1.2,T1.3
T1.5	Meetings	20	
T2	Initial Understanding	20	
T2.1	List, download and classify	2	
T2.2	Study the resources	10	T2.1
T2.3	Install the software	4	T2.1
T2.4	Test the correct functioning	4	T2.3
T3	Data exchange	80	T2.3
T3.1	Program the cases exchange	30	
T3.2	Program the contacts exchange	30	T3.1
T3.3	Design and program the parser	20	T3.1,T3.2
T4	User Interface	15	
T4.1	Design the UI	5	
T4.2	Optimization and algorithm analysis	10	
T5	Project documentation	80	T1,T2,T3,T4
T5.1	Collect all the information obtained	10	
T5.2	Write the documentation	70	T5.1
T6	Bachelor thesis defense preparation	20	T1,T2,T3,T4,T5
Total		300	

Table 2.1: Table of the tasks [Own elaboration]

unexplained issue or have inadequate documentation. Waiting for a library update that fixes the bug or attempting to recreate the function from scratch, accepting the extra time cost, are two options for solving this problem. Estimated overtime: **10 hours**. *Risk level: 2.*

- **WHO or country priorities change**. Depending on the epidemiological situation some countries may require fast solutions for different problems. Taking into account that this project is within a real environment and other WHO precedents, this may pose a real risk to the objectives, planning and expected results of this project. *Risk level: 5.* [1]

Due to all these risks and obstacles we have overestimated the time of some key tasks.

[1] Estimated overtime not defined because as it affect to the objectives of the project, it would directly affect to the project planning and expected results.

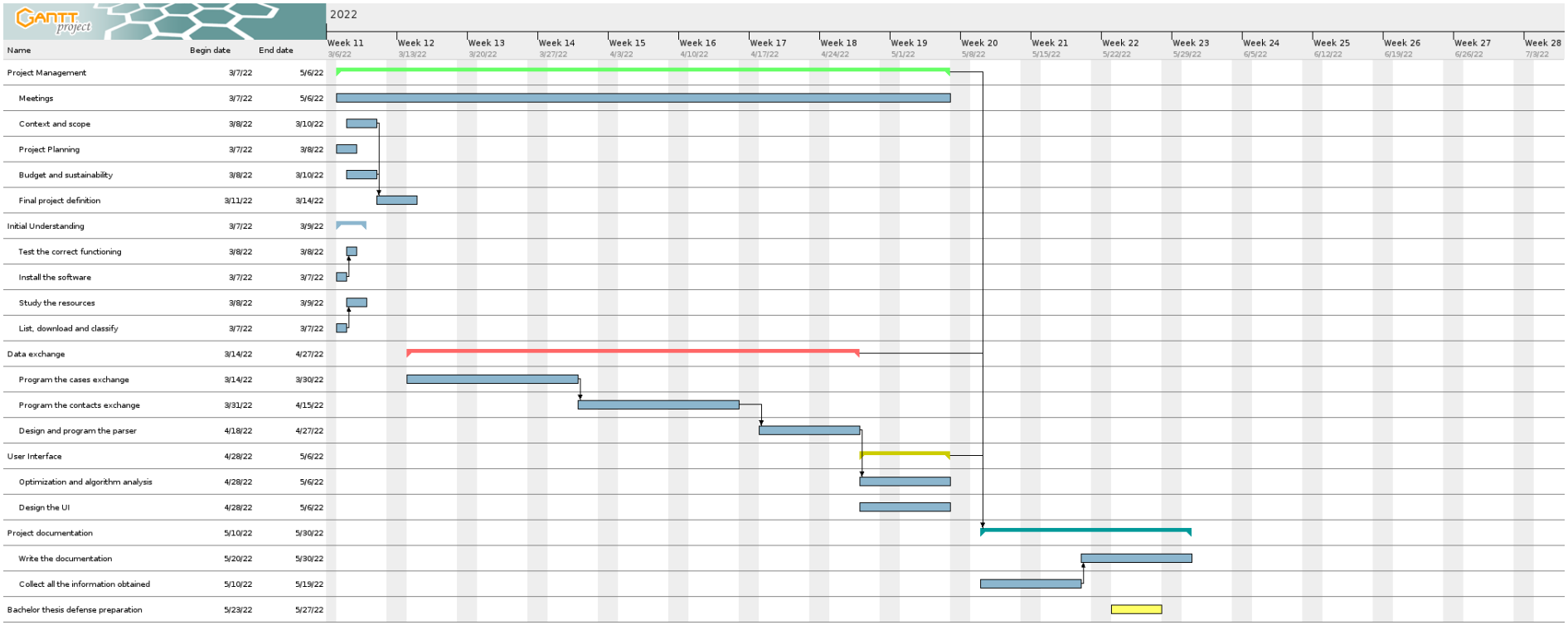

Figure 2.1: Gantt diagram [Own elaboration].

Chapter 3

Budget and sustainability

3.1 Budget

It will be described the elements to consider when doing the budget estimation,which includes personnel costs per task, generic costs and other costs. Moreover, it will be defined management control mechanisms to control the deviations that can appear in the project due to unforeseen obstacles. Finally, it will be answered some questions regarding the sustainability aspect in the project.

3.1.1 Personnel costs per activity

In this section it is computed the total cost for each task defined in Section 2.4. The cost for one task will be calculated summing the cost of the work of the personnel. The cost for each worker will be computed multiplying his cost per hour by the amount of time that they will be involved in the activity.

This project is part of a larger project, and only the tasks involving the part of the project that directly impacts it will be considered.

In this project there are 4 types of personnel, each one with a different cost per hour. Firstly, the project manager is responsible for the planning and correct development of the project. The GEP tutor, the tutor and me will play this role. Secondly, the programmer and the tester have to program the code and verify its correct functioning. This two roles will be played only by me. Finally, there is the technical writer who has to document everything that involves the development and results of the project. A recently incorporated member to the

team, *Mateo Jácome*, will play this role. Following, it is shown the annual salary of the different project roles.

Now, we can compute the total cost for each task. Table 3.1.1 shows the distribution of time for the personnel for each task, and its total cost. This is known as CPA.

Role	**Annual Salary (€)**	**Total including SS (€)**	**Price per hour(€)**
Project manager	39.004	50.705,2	28,97
Programmer	26.198	34.057,4	19,46
Tester	20.592	26.769,6	15,29
Technical writer	26.263	34.141,9	19,50

Table 3.1: Annual salary of the different project roles. It has been estimated the total number of hours worked per year, which is 1750. Information obtained from [6].

ID	Name	Total Hours (h)	Hours				Cost (€)
			Project manager	Programmer	Tester	Technical writer	
T1	**Project Management**	**145**	**85**	**20**	**20**	**20**	**3547.45**
T1.1	Context and scope	**20**	20	0	0	0	579.4
T1.2	Project planning	**10**	10	0	0	0	289.7
T1.3	Budget and sustainability	**15**	15	0	0	0	434.55
T1.4	Final project definition	**20**	20	0	0	0	579.4
T1.5	Meetings	**80**	20	20	20	20	1664.4
T2	**Initial Understanding**	**16**	**0**	**15**	**0**	**1**	**311.4**
T2.1	List, download and classify	**2**	0	1	0	1	38.96
T2.2	Study the resources	**8**	0	8	0	0	155.68
T2.3	Install the software	**4**	0	4	0	0	77.84
T2.4	Test the correct functioning	**2**	0	2	0	0	38.92
T3	**Data exchange**	**73**	**0**	**54**	**19**	**0**	**1341.35**
T3.1	Program the cases exchange	**28**	0	20	8	0	511.52
T3.2	Program the contacts exchange	**28**	0	20	8	0	511.52
T3.3	Design and program the parser	**17**	0	14	3	0	318.31
T4	**User Interface**	**15**	**0**	**15**	**0**	**0**	**291.9**
T4.1	Design the UI	**5**	0	5	0	0	97.3
T4.2	Optimization and algorithm analysis	**10**	0	10	0	0	194.6
T5	**Project documentation**	**31**	**0**	**0**	**0**	**31**	**604.5**
T5.1	Collect all the information obtained	**6**	0	0	0	6	117
T5.2	Write the documentation	**25**	0	0	0	25	487.5
T6	**Bachelor thesis defense preparation**	**20**	**20**	**0**	**0**	**0**	**579.4**
Total		**300**	**105**	**104**	**39**	**52**	**6676**

Table 3.2: Personnel cost for each task defined. [Own calculations]

3.1.2 Generic costs

Amortization

One aspect to take into account is the amortization of the material resources used in the project. It is considered an average of 3 working hours per day. during a total of 102 days. It is estimated that an all of the project has been carried out using the laptop computer. The equation to compute the amortization for each resource is the resource price (The price of the laptop is 1200€including the peripherals) divided by the years old of the product(5 years), by the days that this product will be used (102 days) and by the hours per day (3 hours). At the end, it can be multiplied by the number of hours used (300h) and we get a total of **240€**.

Electric cost

Regarding the electricity cost, average cost of the kWh is 0,2€[16] the last 4 months. We only count the expenses of the hardware when they are turned on. I should remark that we have to add also the monitor expenses. The power of the The laptop is 280W, the total hours used is 300 so the total kW of consumption is 300*280=84000 which is 84 kW what computes (0,2*84) a total of **16,80€**.

Internet cost

The internet rate costs 40€per month. Taking into account that the project lasts 4 months and that the working hours per day are 3 the internet cost is 4 months * (100€/month) * (3h/24h) = **50€**.

Water cost

The water cost in my zone area costs around 25€/person. Using the equation in the previous section and using the same number of working hours per day, the water cost is the following: 4 months * (25€/month) * (3h/24h) = **12,50€**.

Travel cost

As the project is developed completely remotely, the transport cost is nil; **0€**.

Generic cost of the project

Table 3.1.2 summarizes all the generic costs of the project introduced in the previous sections. The total cost is computed summing the CPA cost with the CG cost (generic cost).

Concept	**Cost(€)**
Amortization	240
Electric cost	16,80
Internet cost	50
Water cost	12,50
Travel cost	0
CG Cost	**319,30**

Table 3.3: Generic cost of the project. [Own calculations]

3.1.3 Other costs

Contingencies

During the development of the project it can appear unforeseen events, which take part of our budget. For this reason, it is always necessary to prepare a fund of contingency to be prepared to face these events. To the total cost (CPA + CG = 6989,98), we have to add a 15% of contingency margin. With this, the computed contingency cost is **1048,50€**.

Incidental costs

We have also to take into account the cost of applying alternative plans in case unexpected events occur during the course of the project. The alternative plans can be found in previous sections. Table 3.1.3 show the total cost to solve these events. The cost for each incident is computed multiplying the price it would cost by the risk probability that the event occurs.

Concept	**Estimated cost(euro)**	**Risk(%)**	**Cost(€)**
Bugs found in the currently implemented algorithms(20 hours)	500	80	400
Early deadline of the project (20 hours)	250	40	100
Bugs in some libraries or difficulties understanding them (10 hours)	500	20	100
Total			**600€**

Table 3.4: Incidental costs of the project. [Own calculations]

3.1.4 Total cost

The total cost expected for the project is found in Table 3.5, computed using all the justified costs calculated in the previous sections.

Concept	**Cost(€)**
CPA cost	6676
CG cost	319,30
Contingency cost	1048,50
Incidental cost	600
Total	**8643,80**

Table 3.5: Generic cost of the project. [Own calculations]

3.2 Management control

In this section, we will define control indicators that help to supervise cost deviations during project execution. Obtaining the difference between real and estimated resources utilized is a control technique for measuring and avoiding deviations.

While doing each task described in the planning section, I will calculate its individual deviation from the estimated cost. The following formulas will be used:

- **Estimated cost:** Estimated cost of the task.
- **Real cost of the task.** Hence, I shall recalculate the CPA, CG, contingency cost and incidental cost for each one to check any incidences that may increment its cost. Doing so helps to identify which part of the task has been miss-estimated in order to re-plan it.
- **Deviation:** Difference between the estimated cost and the real cost of the task.

We can easily see and understand where and why a deviation happened, as well as how much the deviation cost, with these indicators. If the overall cost variation is negative, we will have to spend the money put aside for contingencies.

3.3 Sustainability

3.3.1 Self-assessment

The proposed survey about sustainability has made me consider the repercussions, causes, and solutions to today's economic, social, and environmental issues.

Mentioning my strengths first, I knew a few concepts from my recent participation in the second edition of the *Blockchain4SDG* contest. Thanks to that, I knew by hearsay the Sustainable Development Goals (SDG), I also knew what they were, the 2030 Agenda and some important points of sustainability.

Another source of knowledge comes from taking the subject *Free Software and Social Development* and I think that as far as the software is concerned this project is highly sustainable making a complete usage of free software and sharing their software. All the code programmed will be shared to everyone using the Github platform.

But, to be honest, the poll has made me aware of all the concepts and techniques about sustainability that I had absolutely no idea about. The more questions I answered, the more I got the idea that I don't know the various sustainability problems, not the tools to detect them and to deal with them.

When I finished answering the questions and seeing that of almost all the questions I had answered that I knew little about the concepts, I did a little research on the point of view of the three different dimensions and I understood the importance to take into account the sustainability when planning and developing a project.

This poll also made me realize about the importance of some initiatives such as *Blockchain4SDG* and subjects as SLDS since if it were not for it, I would not know anything about the importance of sustainability in all aspects.

3.3.2 Economic dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

The reflection on the estimated cost for the project, as well as the management control, can be found in section 3.1.4 of the document. It has taken into account the human, hardware, software and material resources. The salary for the personnel has been obtained searching through the internet. Moreover, we have

also measured other costs (contingencies and incidental costs) and their effects in the budget.

Personally, I did not expect such a high figure to be reached, however costs could be reduced if an office for workers was used (if the cost of travel cost does not increase much).

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)?

This project is subsidized by CCD. In case we need any tool that requires an additional cost, it is good practice to talk with the supervisors to negotiate if they can provide it. Moreover, WHO in case of urgency or need has offered to add budget to increase the team.

How will your solution improve economic issues (costs ...) with respect other existing solutions?

In our case, if we are referring to another existing solutions implies to hire people to manually write information from one side to another. So, it is much cheaper and scalable to design a program that does the work.

3.3.3 Environmental dimension

Regarding PPP: Have you estimated the environmental impact of the project?

I have not calculated the project's environmental effect. We have calculated the amount of electricity that will spent during the project development and travelling won't be required, so, there will be no contamination in that aspect.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

Yes. The only resource that could be reused is the programmed code by other researchers and a proper usage of the current software. As said before, if the travel cost is not high, an office would be useful to reduce impact(Cost and environment impact).

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)? How will your solution improve the environment with respect other existing solutions?

The project would make a contribution in monitoring disease outbreaks, some of them caused by environmental conditions. The project would help the development of prevention against these scenarios. All this in less time and greater efficiency than other solutions.

3.3.4 Social dimension

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

In the short term, it gives me knowledge that I can use in my current job since similar technologies are used. In the long term, it brings content to my resume, experience holding meetings in English and participating in a large-scale project, among others.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?

The problem is currently solved with a software with limitations and a recurrent errors when data exchange is produced between Go.Data and DHIS2 instances.

How will your solution improve the quality of life (social dimension) with respect other existing solutions?

The currently used software causes a lot of headaches to the DHIS2 users that pretend to exchange data, making them to send mails repeatedly asking for help and showing error messages. On the other side, the quality of life of the involved developers will improve also if they can progress in other projects instead of solving strange errors from the same software.

Regarding Useful Life: Is there a real need for the project?

Absolutely, this project is the progress of a component of an international development cooperation initiative known as *Strengthening countries in adopting specialized outbreak management software suite (Go. Data) by enabling inter-*

operability, promoting secure data exchange and automation, together with the Database Technologies and Information Management (DTIM) research group and is a direct collaboration with the WHO. Ergo, the app is constructed directly based on their requirements.

Chapter 4

Developement

In the following chapter, the development part of the project is described. From now on, in this document we will refer to the *Murod Latifov* app (See *godata-dhis2-interop-app*, section 1.3.1) as *"the app"*, since it will be build using the code that was previously developed in this repository. All of the code can be found in the Github repository:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app

During the development of the project, in order to make the commits of the `main` branch correspond to stable versions of the app, it is decided to create a development branch called `develop`. So, if you want to track all the changes across commits, you should also check the `develop` branch:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/tree/develop

4.1 Deep analysis of the app

This section describes the second task labeled *T2: Initial Understanding*. As described in Table 2.1.

4.1.1 Brew explaination of how the app works

For the data exchange between DHIS2 to Go.Data, the credentials for each instance have to be introduced and a data mapping has to be created to adapt the data to the receiver schema.

Once the credentials have been introduced and the data mapping has been produced, a task can be created with the endpoints from which the data is collected, the endpoints from where it is received, the category of data and the data mapping to be used.

Finally, the task can be executed and, in consequence, the data parsed and transfered.

See the credentials form with Figures C.14 and C.13, the data mapping creation tool at the Figure B.4 and the task creator form at the Figure C.12.

4.1.2 Analysis of the AppHub rejection report

DHIS2 is a modular platform in which applications can be installed manually or download them from a marketplace called AppHub. So that an application to be published in the AppHub has to go through a series of mandatory reviews.

The development of this project is marked by the AppHub rejection report of the app that shows part of the current status of the software; the problems it presents and redefines the development priorities of the project.

For the WHO, the fact that the App appears on the AppHub is of vital importance since it makes the countries have greater confidence that the App is stable and reliable, that is, it encourages its use much more.

Unfortunately, the initial version of the *godata-dhis2-interop-app* created by *Murod Latifov* (Section 1.3.1), prior to my entry, is refused to be published on the AppHub after being reviewed by the DHIS2 team from the Oslo University.

As one of the objectives of the WHO is to have the app posted on the App Hub (Above the objectives of section 1.3.1), this objective becomes added as the highest priority.

To get more details about the motivations the app rejection, a meeting was arranged with the team that manually reviews the submitted apps and they presented all the problems for the approval of the app and classified the issues into two groups: *Major issues and minor issues.*

- **Major issues:** App issues that prevent it from being published.
- **Minor issues:** App issues that do not affect the App Hub submission but it is recommended to check out.

With regard to all these considerations, to the objectives of this project specified in section 1.3.1, we add the objective of *Submit successfully the App*

in the App Hub with the major issues as sub-goals and also add some minor issues that must be taken into account. The new objective decomposed after analyzing the rejection report and meeting with the DHIS2 team is:

1. **Submit successfully the App in the App Hub**
   (a) Security issues
      i. Credentials stored in plaintext
      ii. Credentials logged on the console
      iii. Delete exposed undocumented services
   (b) Delete constants resource as a *"pseudo-datastore"*[1]
      i. Reserve Data Store[1] namespace using *dataStoreNamespace*
      ii. Use the encrypted parameter especially credentials for DHIS2 or Go.Data
   (c) Change the app logo to one that does not violate DHIS policies
   (d) Important *minor issues*
      i. Errors that make the app work only in certain environments
      ii. Delete or fix components that cause a huge chain reaction of requests and renders
      iii. Delete infinite recursions

4.1.3 Code quality and security

Once the report in the section 4.1.2 has been analyzed, a study of the code and the functionalities of the app is carried out. To review the status of the app before my entry into the project we will consider the commit `6dddf8c69e` as the latest one without my intervention. You can review the files in the following link:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/tree/6dddf8c69e

After a quick look at the project code, you can see a general disorganization of the files and a lot of duplication in the code. This point is important since it will mark the times to complete the objective mentioned in section 4.1.2 and the effort needed to fix each bug or to implement each change. In other words, it is important to measure the scalability and maintainability of the app to specify a date for a new submission to the AppHub.

Sonarqube report analysis

For the purpose of make a more formal analysis and know the exact point where the app is, we trust the open-source platform Sonarqube and execute an analysis with it to the source folder, `/src`, of the app. In figure 4.1 we see that the report informs us that the app contains some bugs, more than 400 code smells and, the main problem of the app, that there are too many **code blocks duplicated**.

In our case, relating it to the aims of the DHIS2 team, the fact that it contains so many duplications will have a very specific effect on the amount of time that objective b) (*Delete constants resource as a pseudo-datastore*) will require since the *dataStore* usage is necessary in many references in many parts of the app. Because this specific change directly affects the entire app, an exhaustive test of all use cases will have to be done afterwards.

On the other hand, during these testing phases, all the bugs and inefficiencies that the app presents must be solved. As the app has to be as robust as possible, the code must be thoroughly studied.

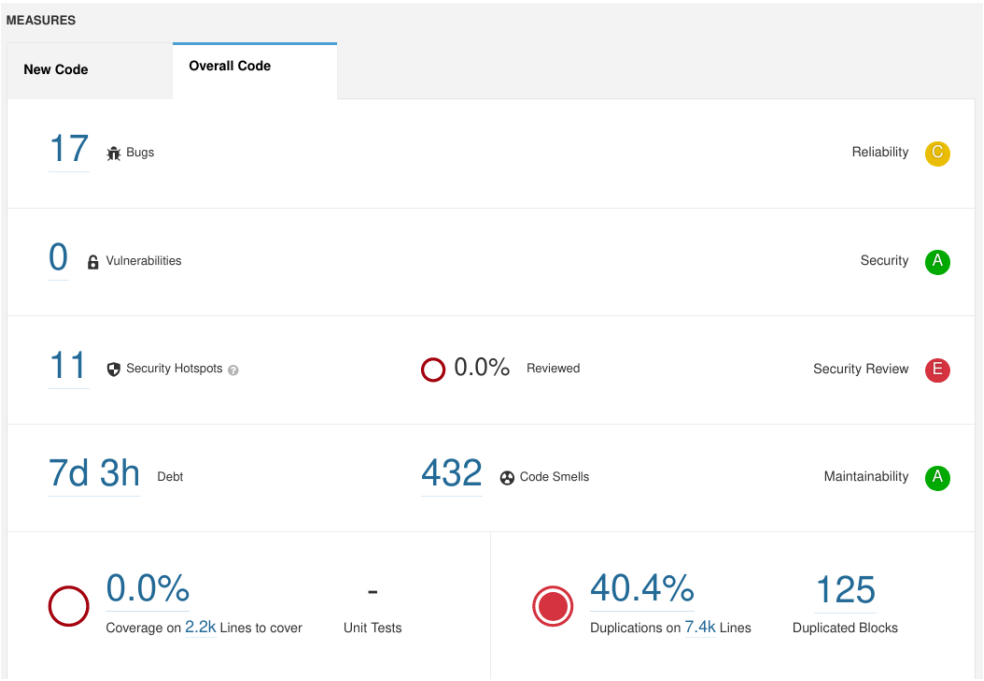

Figure 4.1: Sonarqube analysis. Commit `6dddf8c69e` of the app.

4.1.4 Previously implemented use cases testing of the app

Once the entire environment has been configured, and the app code has been analyzed, a part of the testing of the functionalities already implemented is carried out, paying special attention to the errors or inefficiencies that may occur, taking into account the report 4.1.2. It is worth mentioning that so that the app does not have to be installed in the DHIS2 platform every time a change occurs, the CORS[1] are configured between the app and DHIS2 in order to have a more dynamic development environment.

When running the application, various errors occur regarding the interaction of the constant database (when creating, editing or deleting a task or mapping). After an analysis of the response from the server, the error is solved by changing the parameters related to the metadata required by the constants database (May be due to changes of the required matadata between versions of DHIS2).

Although the data transfer works at a certain point, the truth is that due to the large number of app crashes, conflicts or simply the confusion that the app presents for its user, it is decided to start with the most structural change of the app (*Delete constants resource as a pseudo-datastore*. See section 4.1.2) and go solving through this, the various small bugs and inefficiencies that the code presents.

4.2 Development of the data exchange from DHIS2 to Go.Data

This section describes the third task labeled *T3: Data exchange*. As described in Table 2.1.

The main goal is to enable correct data handling and transfer using *data mappings* and *Task* forms.

To complete this task successfully, it is also necessary to completely eradicate all uses of the app from the constants database by using the *dataStore*.

Although the objectives (See 1.3.1), describe the exchange of Geographical hierarchies, outbreaks, cases and contacts, in subsequent meetings with the WHO team it is also requested to implement the exchange of *contacts of contacts* and *events*.

[1] *"Cross-origin resource sharing (CORS) is a standard mechanism that allows JavaScript XMLHttpRequest (XHR) calls executed in a web page to interact with resources from non-origin domains. CORS is a commonly implemented solution to the same-origin policy that is enforced by all browsers."[17]*

However, to prioritize that the new submission to the AppHub be done as soon as possible, efforts are focused on advancing the transfer of data from DHIS2 to Go.Data and leaving the transfer from Go.Data to DHIS2 for later submissions.

4.2.1 Implementation and application of the *Data Store*[1] database and removal of the *constants resource*

To solve this problem, we first analyze the use of the constants database to replicate it in the *DataStore* implementation.

Analysis of the constants database

For this analysis, in my app I use the button as shown in figure 4.2 to load the default values and various data entries that correspond to the default *data mappings* and *tasks* are saved to my local *constants* table.

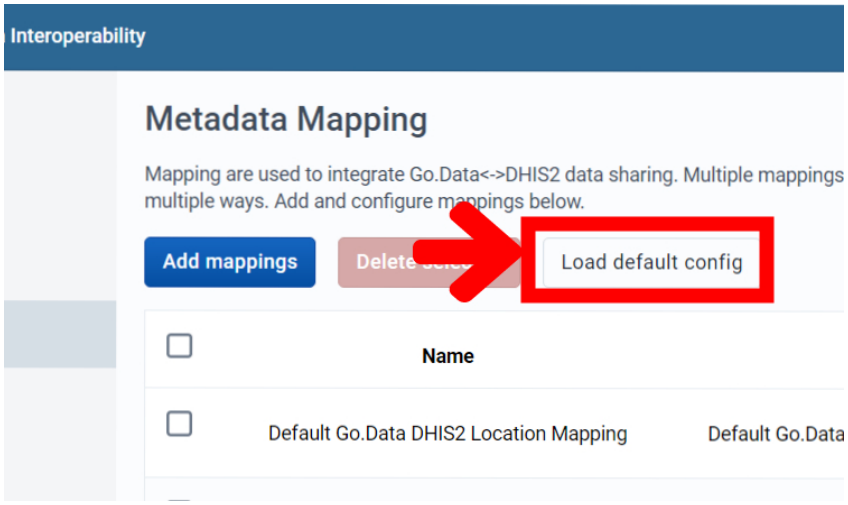

Figure 4.2: Button *Load default config* of the app that creates the default values.

Then, in my local instance of DHIS2 (`http://localhost:8080/`) I make a request to the endpoint to obtain a list of constants[2] which according to the documentation[18] is `/api/constants` but in order to obtain the data in JSON

[2]All of the requests to any DHIS instance require you to introduce the credentials or to provide a valid session cookie.

format in stead of XML and to get the unpaginated response. We ended up making the request to:

`http://localhost:8080/api/constants.json?paging=false`

```
{
  constants: [
    {
      id: "k8IQqmTiUj3",
      displayName: "Default Go.Data DHIS2 Location Mapping"
    },
    [...]
   ]
}
```

Listing 4.1: Fraction of the JSON response of the constants endpoint.

At the snippet 4.1 we can appreciate that the result of the request is a list of id's and a name associated with each id. The name (`"displayName": "... Mapping"`) makes it clear that this constant refers to the content of a data mapping. We can dig deeper and explore the specific value for the id `k8IQqmTiUj3` using the following endpoint:

`http://localhost:8080/api/constants/k8IQqmTiUj3.json`

```
{
  id: "k8IQqmTiUj3",
  displayName: "Default Go.Data DHIS2 Location Mapping"
  description: "[{\"godataValue\":[[{\"conversionType\":\"Go.Dat
a Location\"}],[{\"godata\":\"name\",\"dhis2\":\"name\",\"props\":
\"conversion\":\"true\",\"values\":{}}}, [...],{},{}]",
  value: -1000000
}
```

Listing 4.2: Relevant fields of the JSON response of the constant k8IQqmTiUj3.

```
{
  id: "NQjUfTsCADu",
  displayName: "Default Go.Data DHIS2 Location Task"
  description: "[\"http://dtim.essi.upc.edu:8082/dhis2/api/orga
nisationUnits.json\",\"https://dtim.essi.upc.edu/api/locations/imp
ort\",\"?paging=false,[...],false,\"k8IQqmTiUj3\",\"Go.Data Locati
on\",\"organisationUnits\"]",
  value: -1000002
}
```

Listing 4.3: Relevant fields of the JSON response of the constant NQjUfTsCADu.

We can infer from the data mapping snippet 4.2, and from its comparison with a task model 4.1 several important points about how these values are structured:

1. The `displayName` field is also stored with the other constant metadata, not just in the constants id's listing.
2. The `description` field is the one that stores most of the database information. It is a stringified JSON string. When the constant is fetched from the app to obtain the persistent data, the `description` value is parsed with `JSON.parse()` function.
3. The value from the *value* key is used to identify the type of the constant. For the data mapping the value -1000000 and for the task -1000002. The credentials were also stored using the constants table *identified* as credentials with the value -1000001, -1000003...

How the the *DataStore* and the *userDataStore* works

Before starting with this part, it is convenient to briefly explain how the *dataStore* resource works:

The *dataStore* resource is used to **store any data in from the apps**. The sharing settings of a *dataStore* determine who has access to its key. By default, all keys produced are open to the public (read and write). Additionally, if the app has reserved the namespace, access to a *dataStore*'s namespace is limited to the user's access to the accompanying app[1].

DataStore entries consist of a **namespace, key and value**. The combination of namespace and key is unique. The value data type is JSON[1].

The in addition to the *dataStore* which is shared between all users of the system, a user-based data store is also available. Individual users are associated with data stored in the ***userDataStore***, therefore **each user might have different data on the same namespace and key combination**. The logged in user will be connected with all calls to the *userDataStore*. This means that only the values connected with the currently logged in user can be seen, changed, removed, or added[1].

[3]The specific part where the namespace is reserved for the app is the following:

Integration of the *DataStore* and the *userDataStore* resource to the app

As you can check at the github repository[3], for the *dataStore* resource will be build using the reserved namespace `dhis2-godata-interop-configuration`.

In order for the app to work as close as possible to when the constants resource was used, the structure it presented will be maintained as much as possible.

For the migration to *dataStore/userDataStore*, following the guidelines of the official documentation, we are going to save the Go.Data and DHIS2 app credentials using *userDataStore* and the tasks and data mappings in the *dataStore*.

Due to the nature of these technologies, it will no longer be necessary to use the value in the `value` field to categorize the type of data stored; the value of key will be used. Keys `tasks`,`mappings` for the *dataStore* and `dhisuser`,`dhisuserpass`, `dhisbaseurl`,`godatauser`,`godatauserpass`,`godatabaseurl` for the *userDataStore*. Always using the encryption parameter as required by the DHIS2 team (Section 4.1.2).

On the other hand, it will not be necessary to save the value a in string format either since, as we have already mentioned before(4.2.1), the data type the *dataStore* supports is json.

For data mappings, the endpoint will be *(Base url of the instance)/api/dataStore/dhis2-godata-interop-configuration/mappings* (`/api/dataStore` is the base endpoint for the *dataStore*, `dhis2-godata-interop-configuration` the reserved namespace and `mappings` the key) and it will respond with **an array composed of: the mapping key and the displayName key**. In the mapping key will contain the value that was previously saved as the constant description, without converting it to a string.

Similarly, we proceed to do the same to solve the problem for the tasks.

As for the values stored in the *userDataStore*, these will simply return the string of the value in question.

Interaction of the app with the implementation of the *dataStore* and the *userDataStore* resource

In order for the app to interact with the database efficiently and easily, a library is created with a series of functions that will be useful in many parts of the code.

4

A function that will be very useful to us given our model in that consists in a json array, will be the `appendValue` function since this will add the value to the array if the displayName is not repeated with the values that already exist.

In the case of the *userDataStore*, a library is also created[5], but in this case with just two functions to add or get values.

All these functions, both those of the *userDataStore* and those of the *dataStore*, also check if the user has already created the namespace keys or not and logically add the keys if necessary.

With the help of all these functions, a general tour of the entire app is made, all uses of the constants resource are eliminated in favor of making use of the new libraries. The most relevant errors or bugs will be shown in more detail in the 4.3 section.

Taking advantage of this tour and the reconstruction of the app's logic, some detected bugs are also eliminated and the reported security issues are resolved (Points a) and d) of the new report objectives 4.1.2).

4.2.2 New use cases implementation and testing of data exchange

As suggested in the introduction to the 4.2 section, support for *contacts of contacts* and *events* is also required. On the one hand, *contacts of contacts* exchange has already been implemented from the beginning (It just needed the *dataStore* integration) and, on the other hand, in the case of *events* exchange it required to change the data transfer algorithm and the user interface.

1. The possibility of editing and categorizing a data mapping as the events type has been enabled.
2. In order for Go.Data to accept events (Converted into cases since events as such only exist in DHIS2) it was studied how the data is served by DHIS2, the requirements of Go.Data and the first data mappings were created by default for passing events to Go.Data.
3. The necessary considerations were added to the algorithm so that the events parsed correctly, since the JSON sent by DHIS2 organized the data differently from the case of tracked entities.

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/blob/main/d2.config.js

[4] This library can be found in `./src/utils/dataStore.js` from GitHub.

[5] This library can be consulted in `./src/utils/userDataStore.js` from GitHub.

Once completed, a more realistic testing stage happens by mainly Mateo and the WHO team. All the errors that are found or the suggestions that may be produced are communicated to me in weekly meetings.

4.3 Improvement of the user interface, app efficency and robustness

This section describes the third task labeled *T4: User Interface*. As described in Table 2.1.

Bugs are fixed and the user interface is improved as mentioned in the third goal of the 4.1.2 section. The interface is adapted to a user with less technical knowledge and is also changed based on the feedback from the WHO and the DTIM team itself. These changes are produced taking into account the aesthetic rules of the DHIS2 guidelines[6] and the current App Hub regulations (Third point: 4.1.2).

The most important efficiency and robustness problems of the app are also solved, as mentioned in objective d) of the 4.1.2 and point 4 of the initial objectives (Section 1.3.1).

4.3.1 Change the app logo

The old logo (4.3), which is a mix between the Go.Data logo and the DHIS2 platform logo, has two problems according to the AppHub acceptance requirements [19]: The first is the improper use of the DHIS2 logo since there is no explicit permission (From Go.Data we have the permission to use their logo) and the second is that the logo has to have a higher resolution, concretely, more that $512px \cdot 512px$[19].

[6] See all the guidelines at: https://developers.dhis2.org/docs/guides/apphub-guidelines/ [19]

Figure 4.3: Old app logo. Available checking the initial commit (`6dddf8c69e`). Alternatively you can directly check the logo at the following GitHub link: *https://raw.githubusercontent.com/WorldHealthOrganization/godata-dhis2-interop-app/6dddf8c69e/public/dhis2-app-icon.png*

In a routine meeting, it is agreed to leave the logo as the Go.Data logo, and after some cuts of the original Go.Data logo, the new logo is obtained with the size: $567px \cdot 567px$.

Figure 4.4: New app logo. Measures: $567px \cdot 567px$.

4.3.2 Data mapping editor redesign

Problem analysis

One of the most critical problems at the level of usability and understanding of the app for the user is the part of editing and creating the data mappings (See figure B.4) of the initial app. Apart from this, this section in particular presents

various additional problems at the level of efficiency and robustness. The main problems it presents are:

1. **Low scalability at the code level**: A different file is created with extremely similar algorithms, which makes improving this part of the app very tedious.
2. **Low scalability at the level of use**: From the edition of 50 fields the app starts to work slowly and from 75 it is practically unbearable leaving the screen blank in certain parts (There may be cases with the need to map more than 200 fields).
3. **Unintuitive for the user**
   (a) In order for the fields to be editable, the user has to generate in Go.Data, cases and contacts so that it can be used.
   (b) Cannot add keys.
   (c) The selected values part of a modal overrides the 'Copy to clipboard' functions of the library used for the data mappings state view[7].
4. **Not following DHIS2 aesthetic recommendations**: DHIS2 recommends using its style and React component libraries to follow a cross-sectional stylistic line in the instance.
5. **Wrong or incomplete default data mappings**: The data mapping defaults loaded with the button shown in the Figure 4.2 have to be corrected.

Solution

After analyzing the different form files (`./src/constants/CasesForm.js`, `./src/constants/ContactsForm.js`...), it is concluded that the library `react-json-view`[20] usage as the data mapping edit tool has to be substituted or removed as this is what causes the main problems described.

On the other hand, it is decided to unify all the logic related to the different forms as a single form, thus facilitating the algorithms correction, the organization and the bug resolving of the forms. It has been created a library to support this unification that can be found at the following GitHub link:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/blob/main/src/models/mapping.js

To replace the representation of the data mappings with the `react-json-view`, we opt for the use of the DataTable component from the list of recommended

[7] This library is the Open Source `react-json-view` package from npm [20].

components of the DHIS2 package (@dhis-ui) and make this table editable via modals. See the results at the figure B.4.

Regarding the element selection modal for the DHIS2 value, it was decided to give the user the freedom to choose the data source that he wants to obtain, as seen in figure C.9.

For the automatic key generation, Go.Data was initially approached to implement an endpoint that returns example models for each use case. With these models it is no longer necessary to make a request to a Go.Data case or contact to infer the attributes of the model and a more precise automatic generation is made than the one previously implemented since the example models will always have the maximum number of possible attributes already exposed. You can see an example of an autogenerated template by consulting the figure C.8 in the Appendix C.

The errors that presented the manual insertion of modals are also resolved: A *Submit* button is added and the sumbit function (The 'X' at the top right) of the button that intuitively would serve to Cancel or close the modal is eliminated. The modal is represented at Figure C.10, the *Autogenerate* button checks if the configured Go.Data version supports the new endpoint (Go.Data version has to be greater than 2.40), if not, the app will show a warning.

The possibility of adding the rows by means of a button and the possibility of deleting them by clicking on a trash can logo are implemented[8]. The delete logo is also obtained from the icon list that DHIS2 provides.

Defaults that need changes are changed and new defaults are added. This point is completed by the hand of *Mateo Jácome* and the WHO team.

It is worth mentioning that the way the persistent data of the app is structured is not modified, since the change affects how this data is represented, obtained and modified. You are able to compare the data mappings interface by comparing the figure B.4 with the figures C.7 or C.8.

4.3.3 Changes for the task schema

Some changes are made in the Task schema:

1. Fields that have not been implemented are removed, such as the *Is DHIS2 receiver?* checkbox.
2. Description field is added.

[8]The rows could already be deleted in the previous implementation but it was not intuitive

3. Only the query part of the API calls are required in the fields *DHIS2 endpoint pathname* and in *Receiver API endpoint* since the base of the URL will be built from what the user has entered in the credential forms (Figures C.14 and C.13).

4. Changed the style libraries for the view when a task is being executed to use those from DHIS2 (`@dhis-ui`) and improved the error reporting model.

5. After the removal of a JSON modal reference dependency that was being unnecessary used by the converter algorithm. Removed the input JSON field from the task schema.

Apart from these changes, it also begins with the creation of a library so that at the code level the obtaining of parameters can be managed and the algorithms associated with the task can be centralized, organized and ordered. This library can be found at:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/blob/main/src/models/task.js

4.3.4 Performance improvement

Changes to handling state and the useEffect and useState functions

One of the main problems that the app presents in terms of efficiency is due to the misuse of asynchronous functions in the initial version of the app. This causes infinite loops or a large chain of unnecessary renders as mentioned in the 4.1.2 report.

This is all due to a flawed implementation of React's native useEffect function. The bad implementation meant that every time a Promise was resolved, as the variable to which it was assigned was a state variable that depended on the useEffect, the entire function was re-executed from the beginning.

The creator's intended behavior for the app's useEffect function was to only run once when rendering the component. What the recently deprecated function `componentDidMount()` toward.

To resolve this error, we change the useEffect implementations created for each component to use the programming pattern 4.4. This gives you more control over what is executed, renders, and makes it easier to understand the code.

---

for the user

With this method, many unnecessary inefficiencies are avoided when loading each component, making the loading of each page much faster (and more robust since parallelization errors are also eliminated).

```
const processAll = useCallback(async () => {
  // Here we can use the await token
});

useEffect(() => {
  processAll()
}), [
  //Dependences
]);
```

Listing 4.4: React Pattern.

Creating a custom hook to obtain credentials

A very typical function that was done repeatedly was obtaining credentials for each component in the useEffect (The rendering problem is already solved in the 4.3.4 section) but it remains one of the most critical problems for the performance of the app in terms of duplicate requests to the server and inefficiencies at the processing level.

To solve this, a function is created that makes the requests in parallel when the credentials are obtained. Thus, obtaining is done up to 6 times faster. This function is called `getCredentialsFromUserDataStore` of the file `./src/utils/get.js` and which returns a promise that resolves to all credentials decrypted. Available in:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/blob/main/src/utils/get.js

Apart from that, in order to easily obtain the credentials within the React components, the custom hook 4.5 is created.

```
export const useCredentials = () => {
    const [status, setStatus] = useState({
        loading: true,
    })

    const fetchNow = () => {
        setStatus({ loading: true })
        getCredentialsFromUserDataStore().then(res => {
            setStatus({ credentials: res, loading: false })
        })
```

```
    }

    useEffect(() => {
        fetchNow()
    }, [])

    return { ...status }
}

// USAGE inside the component:
const { loadingCredentials, credentials } = useCredentials()
```

Listing 4.5: Custom Hook and usage.

Creation and analysis of new algorithms

The most critical part at an algorithmic level and in which we have to be more careful in the App is the part in which the data mappings are applied and the data transmissions between DHIS2 and Go.Data is carried out.

In the initial version of the App, an algorithm is already implemented that develops a part of the desired functionality (Transmission from DHIS2 to Go.Data) but it presents many efficiency problems, among which we can highlight:

1. **Lack of flexibility** to adapt to minimal changes or to cover a wider range of cases without generating **erroneous or inaccurate results**.
2. **Poor performance**: Its main problem was that it used state variables (with react's useState) and therefore rendered the component every time iterated.
3. **Unreadable code**, with very confusing variable names and incorrect library usage. Special mention to the misuse of the npm library; *dot-object*, where native implemented functions of this library had been reimplemented in much more inefficient and insecure ways at the initial App.

All this added to the fact that the algorithm in the initial version of the App was not exported from a specific site but was copied and pasted with some small differences in different components of the App, which made its maintenance and modifications very difficult.

Although at the beginning of the development an attempt was made to modify this algorithm and maintain its scheme, it ended up opting for a total reimplementation of the export algorithm from DHIS2.

Regarding the import functionality from DHIS2 (That is, passing and converting data from Go.Data to DHIS2), since this functionality was not in the initial App, it is also created completely from the beginning.

Next, an analysis of the performance of the algorithms is carried out, which can be found in the C.1 part of the appendix C.

First of all, let's define de as the number of data elements a program can have, a as the number of attributes, and n as the number of entries that the data mapping has.

- **Data export algorithm C.1.1:** In the snippet you can see that the algorithm is divided into parts. That said, we define the asymptotic upper odds:

  Let $f(n, de)$ be the estimate function and f_1, f_2 *and* f_3 functions that corresponds to the complexity of the parts defined in the snippet.

  By the property of the sum of the big O we know that :
  $f = f_1 + f_2 + f_3 = O(max(f_1, f_2, f_3))$
  and that
  $f_1 = O(de)$
  $f_2 = O(n^2)$ *Since always* $n >= a$
  $f_3 = O(n)$
  So $f(n, de) = O(n^2)$ (*since it always holds that* $n^2 > de$)

  Therefore, to transfer c cases of a program, we will work with an algorithm with an asymptotic upper odds function equal to $O(c \cdot n^2)$ since this algorithm is applied for each case.

- **Data import algorithm C.1.2**: Analogously to what has been calculated and exposed in the previous algorithm, this algorithm also has a complexity $O(c \cdot n^2)$ being c the number of cases and n the number of entries of the data mapping object.

4.4 Submission of the App to the AppHub

As soon as all the minimum requirements and objectives proposed were completed, the process of submitting the App to AppHub began.

To do this, the compatibility versions of the App were carefully reviewed through different versions of DHIS2 since this type of application can be very sensitive to small updates in the platform's data structure.

The default data mappings that the App offers the user if the default configuration of the App is loaded were also updated for the last time and the last

fringes of the official documentation of the App that is presented to AppHub were finished. The documentation is available at the following link:

https://worldhealthorganization.github.io/godata-dhis2-interop-app/

Finally, the release was also created on Github, which was sent to AppHub with its corresponding zip and, logically, with a tag associated with the last commit of the main branch. The release can be consulted at the following link:

https://github.com/WorldHealthOrganization/godata-dhis2-interop-app/releases/tag/1.0.0

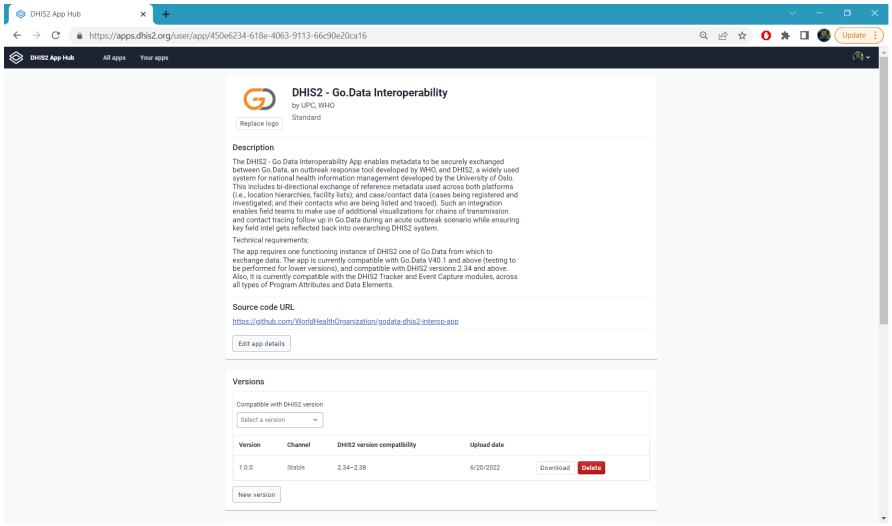

Figure 4.5: Screenshot of the AppHub after the submission. Revision status.

4.5 Development of the data exchange from Go.Data to DHIS2

This section describes the missing part of the third task labeled *T3: Data exchange*. As described in Table 2.1.

This part of development corresponds to the continuation of the 4.2 section, with the difference of what is described below, it could not be incorporated for the first version of the application submission on AppHub since it must go through a testing and documentation phase before being incorporated.

Having said this, a new data transmission algorithm has been created (And

analyzed at the section 4.3.4), accompanied by a new task model that is more scalable and compatible with the creation of data mappings that the App already contains, although necessary parameters will be added in the background when selecting certain fields such as the program identifier of the data element that has been selected.

In other words, the mapping creation process remains the same, but the resulting mapping includes new data required to enable this new data exchange function from Go.Data to DHIS2, such as the program identifier or the program stage identifier of the field that is selected.

In short, with the new task model and the new possibility of importing data to DHIS2, progress has been made in some characteristics claimed by the WHO team, such as:

1. **More robust and more descriptive model** when displaying and offering the possibility of filtering entities (cases, contacts...) prior to sending. An interface with paging and with flexible visualization of the columns is proposed depending on which the user wants to review to decide whether to mark it or not. You can compare figure B.1 from the initial version of the App with figures C.5 and C.6.

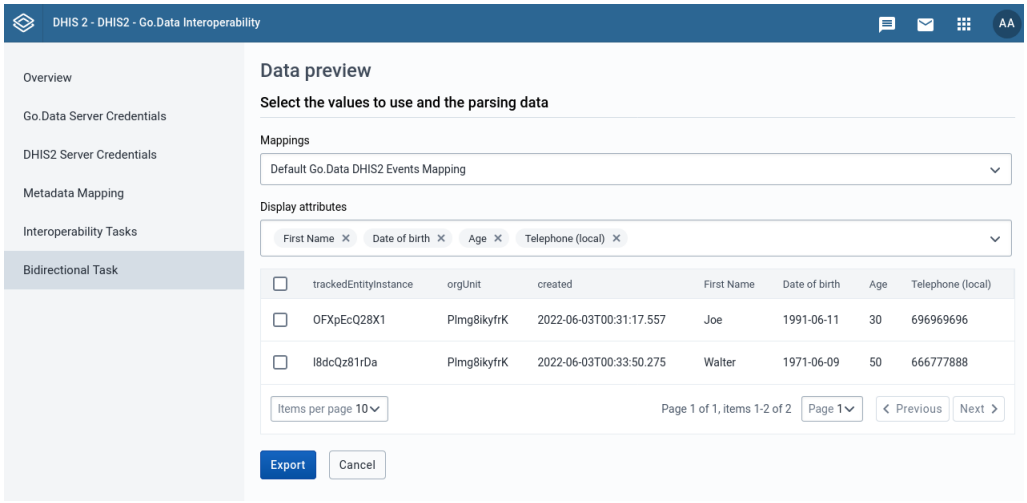

Figure 4.6: Export preview. Flexible visualization. First Name, Date of Birth, Age and Telephone (local) selected.

2. **More and better feedback to the user.** More and better response from the application when there are errors of some kind. Infinite loading screens by mistake or false positive messages in case of error are also

eliminated. Figure 4.7 corresponds to the old implementation and figures 4.8 and 4.9 corresponds to the new proposal.

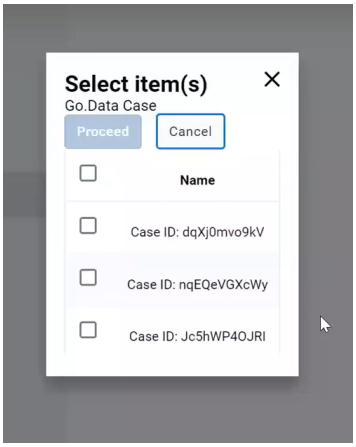

Figure 4.7: Initial version of the app. Case selector. No information about the case is provided.

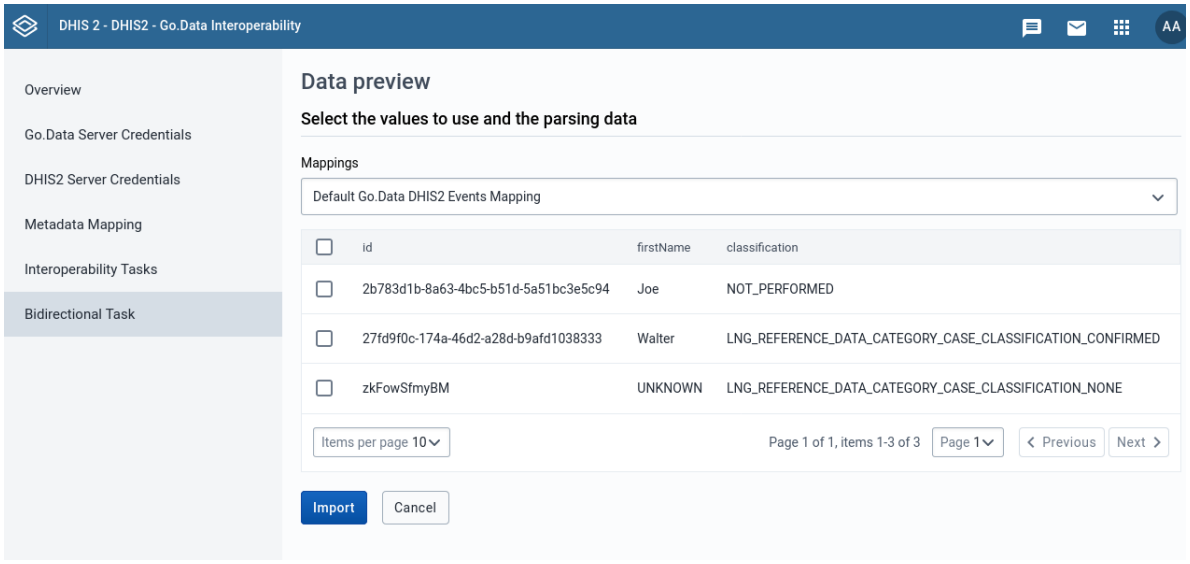

Figure 4.8: New data preview with pagination and the data mapping selector.

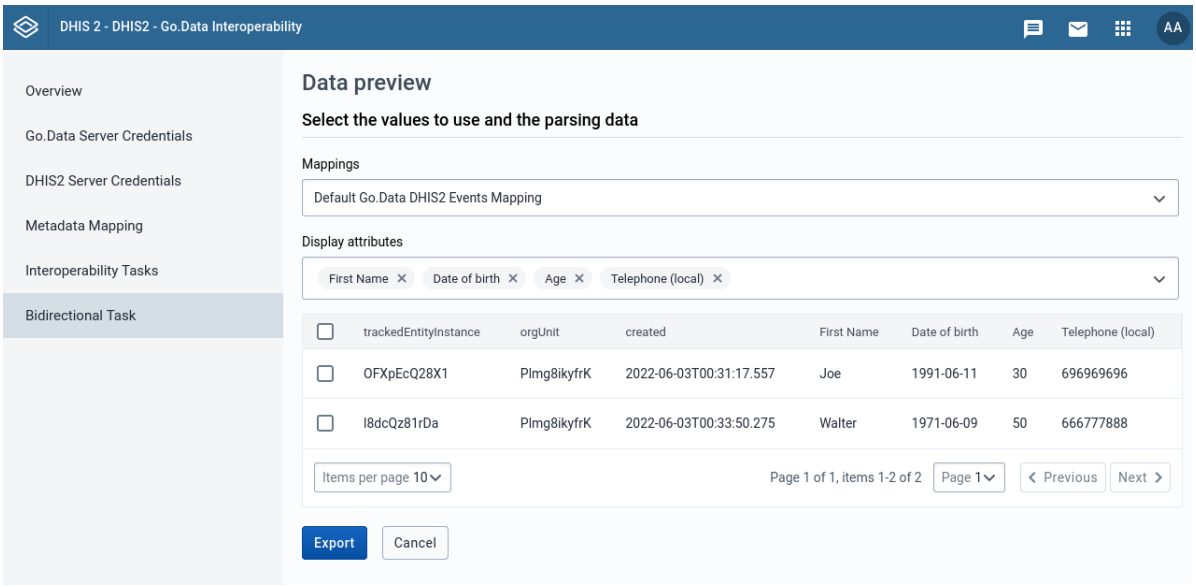

Figure 4.9: Export preview. Flexible visualization. First Name, Date of Birth, Age and Telephone (local) selected.

3. **New data conversion algorithm.** Some new algorithms are generated for the data import function and recreated for the DHIS2 data export function, thus allowing greater understanding and versatility when making modifications to parts or add new types of conversions. It is also changed because the previous implementation sometimes generated erroneous results. New functions are also incorporated, such as the possibility of adding a default value in case of using the type of conversion value-value (Values field of the table). The algorithm can be checked at the appendix section C.1.

4. **Data exchange from DHIS2 to any platform is enabled**. Thanks to the new algorithms and the increase in data collected in the data mapping phase, the use of the App is amplified from bidirectionality from between *DHIS2 and Go.Data* to between *DHIS2 and any other platform.*[9]

[9]This new feature is a side effect of the nature of the algorithm being implemented but it is not a feature required by the WHO team as their investment is focused on Go.Data and not on DHIS2.

4.6 Participation at the *WHO Global Consultation on Go.Data interoperability*

In the final part of the development phase, we participate in the *WHO Global Consultation on Go.Data interoperability*, a very formal event with the participation of around 50 technical experts involved in design and deployment of Go.Data interoperability solutions, together with specialists in digital health standards for interoperability.

In this event, Petar Jovanovic (Director of this thesis) and I show the progress of the App, we explain the proposal and we make a small demonstration of the most basic use cases. Some images of the event are shown in the appendix D.

You can also consult the videos that I created to demonstrate the App by accessing the following YouTube playlist:

https://www.youtube.com/playlist?list=PLrb-ex2QWRa58P3R-QMwbbqgbHagmLwW5

This playlist will also be used to accompany the documentation of the App and, thus, to the submission on AppHub.

Chapter 5

Conclusions

The Go.Data project is part of the WHO Global Strategy on Digital Health 2020-2025, launched under the vision of *improving health for everyone, everywhere by accelerating the development and adoption of person-centric digital health solutions to prevent, detect and respond to epidemics and pandemics.*

The onset of the coronavirus disease pandemic in 2020 quickly shifted the project's focus to assisting countries with Go.Data implementation for pandemic response.

The main objective of this final degree thesis was to collaborate in advancing the *Go.Data-DHIS2 Interoperability App* to enable efficient, robust and secure interoperability between Go.Data and DHIS2 so that Go.Data becomes a more useful and powerful tool to better track pandemic outbreaks and all their associated data such as cases, contacts, events, contacts of contacts...

With the rejection report of the App at the beginning of the development of this project, the organization of hours faltered since higher priority objectives were raised than those initially described in the 2.1 table, which consisted of satisfying the pretensions of the report and re-generate a submission to the AppHub with a new App.

In fact, throughout the development of this project, unfortunately all the risk forecasts set out in the 2.7 section have been fulfilled, adding a total of 50 extra hours that, added to the tasks added by the change in priorities, make a total of about 80 hours above schedule.

However, although due to these unforeseen events the submission on AppHub is missing the interoperability part from Go.Data to DHIS2, this part could be developed on time and may be incorporated in future versions of the App.

Therefore, despite the fact that the project required the maximum amount of overtime that had been estimated, in the end absolutely all the proposed objectives were met.

Another task to highlight in this project is the important restructuring of the code throughout the implementation of the tasks, making it a more scalable and sustainable App over time, which will facilitate future improvements for future developers of this software.

In another order of things, an important task of team work and coordination is also carried out on a small scale, with the director of the thesis Petar Jovanovic guiding the course of the project and Mateo Jácome writing the documentation and making important contributions in matters of user experience, as well as on a larger scale with the weekly meetings with the WHO team and without forgetting the enriching experience of being able to participate in the *WHO Global Consultation on Go.Data interoperability*.

Finally, the technical competences required for completing the project are detailed in Appendix A.

5.1 Future Work

Next, some important points are described that should be covered for future improvements of the App, divided according to their degree of priority.

5.1.1 Short Term

1. Testing and documentation of the part that enables the import of data to be able to update the version published on AppHub.
2. Import and export of DHIS2 option sets.
3. Link of the cases and contacts in the transmission from DHIS2 to Go.Data.
4. Make progress in making the task creation part more user-friendly.
   (a) Replace the manual entry of URLs with a more user-friendly system.
   (b) Selecting the Go.Data outbreak from a dropdown and displaying its name (not its id).
   (c) Show location name instead of location id.
5. Define if the application will work only in the same instance of DHIS2 in which the App is installed or if this App can work in other instances of DHIS2 even if it is not installed.

(a) If it is decided that it can only work on the same instance on which it is installed: Remove the Basic Authorization header as it is not recommended for security reasons.[1]

5.1.2 Mid Term

1. Create the possibility of programming some cron jobs from the App so that tasks are executed in the background every certain time.
2. Add the ability to import locations to DHIS2 from Go.Data.
3. Substitute the content of *Interoperability Tasks* and use the logic created for *Bidirectional Task*.
   (a) Make the configuration of a *Bidirectional Task* possible to be easily saved and executed at any time.

5.1.3 Long Term

1. Enable import and export of data mappings by CSV.
2. Change rendering to Server Side Rendering.
3. Improve the automatic generation of mappings, including, perhaps, creating an AI that performs the data mappings automatically.

[1] The *Basic access authentication* header `Authorization: Basic <credentials>` exposes the credentials since only the username and password separated by ':' are encoded in Base64.

Appendix A

Computer Science Technical Competences

CCO1.1: To evaluate the computational complexity of a problem, know the algorithmic strategies which can solve it and recommend, develop and implement the solution which guarantees the best performance according to the established requirements. [In depth]

On the one hand, a very significant reduction in unnecessary renderings of user interactions with the screen is evaluated and achieved (It was even reported by App reviewers from the Oslo team that, on occasions, the state of the App froze due to loops with pseudo-infinite renders) and, consequently, the number of requests to the different instances made by the App is also very significantly reduced.

All of this is managed through a better management and understanding of the states of the components in React with respect to the initial version of the App and a re-implementation of many functions that unnecessarily altered and rendered the state of the App.

On the other hand, the additional fields that will be necessary to enable bidirectionality when mappings are created (such as the identifier of the program and of the programStage in case of mapping to a Data Element) are studied and an algorithm is created and others that it offers are corrected. results without errors and with the best performance since the key algorithms are analyzed in detail.

Visit the section 4.3.4 to check most of the performance improvements and the main implemented algroithms evaluated.

CCO1.2: To demonstrate knowledge about the theoretical fundamentals of programming languages and the associated lexical, syntactical and semantic processing techniques and be able to apply them to create, design and process languages. [In depth]

In this project we work mainly on JavaScript (Using React as a framework) and throughout the project its use is maximized. From native functions (reduce, map, forEach, slice, split, etc...) prioritizing maintainability, readability and code comprehension to the creation of classes, prototypes or generator functions.

React's build and state management are closely studied to increase efficiency and reduce bugs.

Parallel and consequently to the use of React, HTML and CSS tags are used and it is integrated using the style libraries recommended by DHIS2.

On the other hand, skills are also demonstrated with the use of other language such as bash scripting (automated tasks are carried out such as periodic cleaning of the database, restarts or deployment of instances in external servers), queries are executed daily during the development in the MongoDB database model to verify and debug the results, and also works in Markdown to create the documentation or LaTex to write the final report of the thesis.

It also deals with data in different formats such as JSON or xml and requests (and error analysis) are made by editing and intercepting the requests in HTTP protocol (By using BurpSuite, Postman or using the same FireFox browser).

CCO1.3: To define, evaluate and select platforms to develop and produce hardware and software for developing computer applications and services of different complexities. [In depth]

The Go.Data and DHIS2 platforms are evaluated very deeply, trying to get the most out of both by evaluating and deeply studying their respective endpoints, their various data models, their required parameters, their conditions and their different security recommendations.

For the App that is defined and developed throughout this project, the DHIS2 platform is selected.

CCO2.1: To demonstrate knowledge about the fundamentals, paradigms and the own techniques of intelligent systems, and analyse, design and build computer systems, services and applications which use these techniques in any applicable field. [Enough]

The App that is developed throughout this project is an intelligent system which is the result of different iterations of analysis, design and construction of the application.

The intelligent systems analysis part is produced in two very different phases of the project:

1. Workflow analysis phase of the two previously made applications: The necessary information is extracted to use the material as much in our favor as possible in order to design or redesign the new parts of the App.
2. Weekly, during the project, before and after the implementation or modification of the App, an analysis is carried out and contrasted with the opinion of the director of this TFG and the WHO team.

In fact, a large part of the design and construction of the App after the analysis has been to enhance its operation towards a more versatile use and lay the foundations so that, on this basis, other automatic data self-detection techniques can be applied.

CCO2.2: Capacity to acquire, obtain, formalize and represent human knowledge in a computable way to solve problems through a computer system in any applicable field, in particular in the fields related to computation, perception and operation in intelligent environments. [In depth]

Especially in the data mapping part, it is from where human knowledge is obtained and represented, in this project the amount of data collected for each selection is enriched, it is formalized and the representation of these data is greatly improved, passing from a format Inefficient rendering of a JSON to a set of tables with preset values.

All these data mappings are used to solve the algorithmic challenges of automated conversions posed by the transfer of data between the Go.Data and DHIS2 platforms.

Visit section 4.3.2: Data mapping editor redesign.

CCO2.3: To develop and evaluate interactive systems and systems that show complex information, and its application to solve person-computer interaction problems. [In depth]

In the initial phase, the initial representation made by the App of the data is evaluated as poorly, betting on an extremely inefficient use of a representation in JSON format editable to the end user that had innumerable errors of use.

Throughout this project, it is replaced by a more robust and more user friendly system, betting on the representation and editing of data through an interactive table using the style sheets proposed by DHIS2 and adding the feature to the table of being able to automatically infer the conversion type to use.

CCO3.1: To implement critical code following criteria like execution time, efficiency and security. [Enough]

Following the AppHub recommendations, certain tasks are performed along these lines:

1. Security: Parts of the code where some credentials could be read in clear text are removed.
2. Security: Routes in which unfinished services were exposed and that represented a vulnerable vector for the application if it received a brute force attack are eliminated.
3. Security: The credentials displayed on the console are deleted.
4. Security: Security alerts, bugs, and security hotspots automatically reported by Sonarqube analysis, GitHub security bot alerts, and npm packages are updated, both those with security alerts and those with security hotspots.
5. Security, efficiency and execution time: All references and uses of the Constants Table are completely eliminated and the use of the *dataStore* and the *userDataStore* are replaced using the data encryption option in the case of the *userDataStore*.
6. Efficiency and execution time: Fix components with a huge chain reaction of requests and renders and infinite page renders.

In terms of efficiency and execution time, mention is also made of the implementations described in the first technical competence CCO1.1.

Appendix B

Initial state of the App

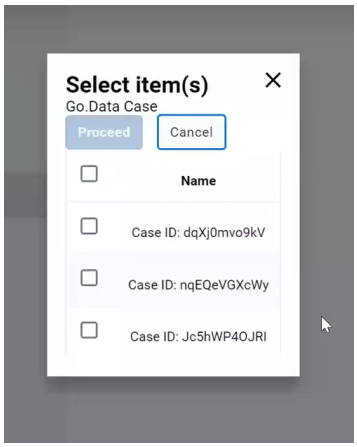

Figure B.1: Case selector. No information about the case is provided.

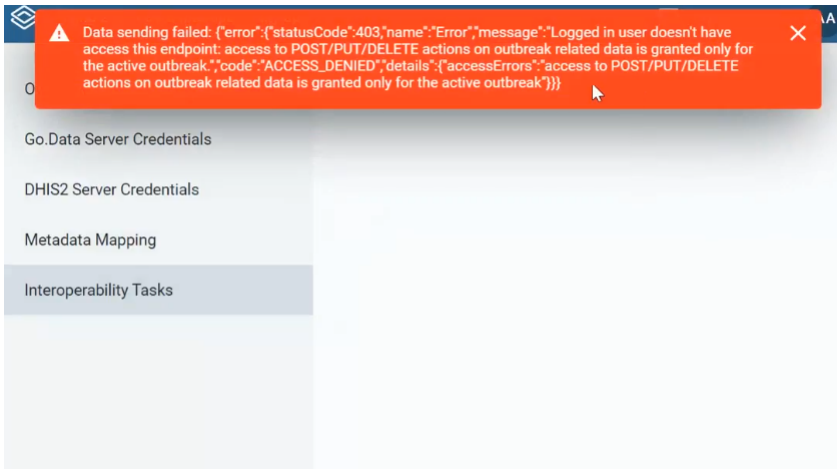

Figure B.2: Data sending failed. Feedback to the user.

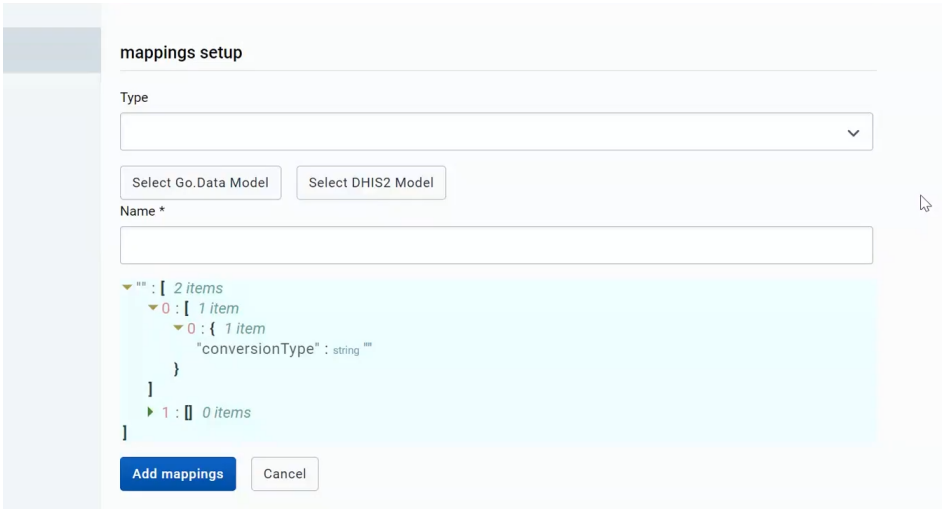

Figure B.3: Custom data mapping creator.

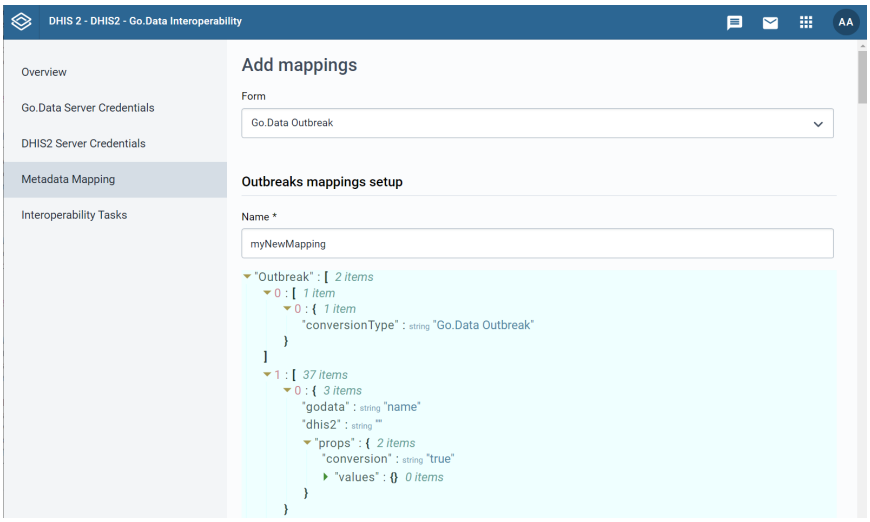

Figure B.4: Data mapping edit view of the Outbreak Form.

Appendix C

New (Improved) state of the App

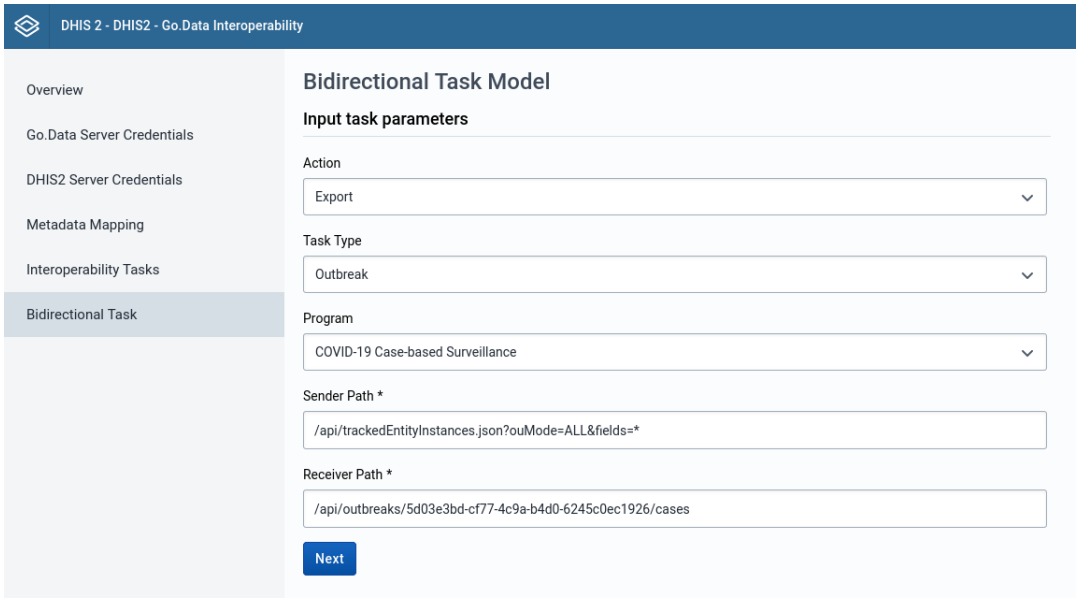

Figure C.1: New form for the bidirectional task.

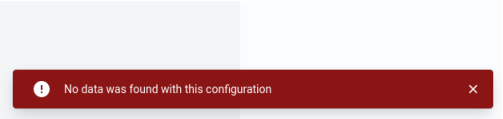

Figure C.2: Error feedback for the bidirectional task data preview

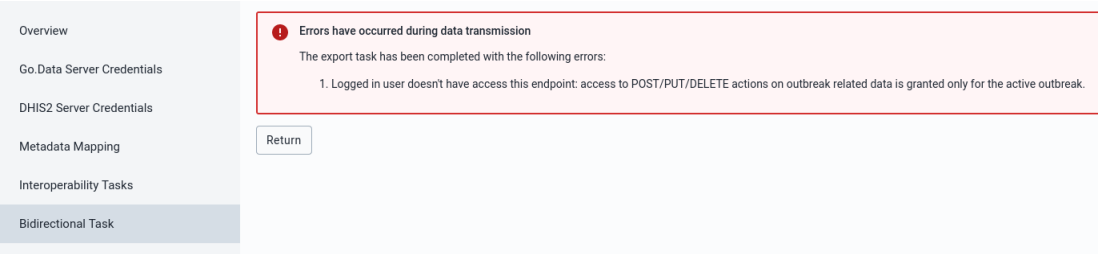

Figure C.3: Error feedback after task execution

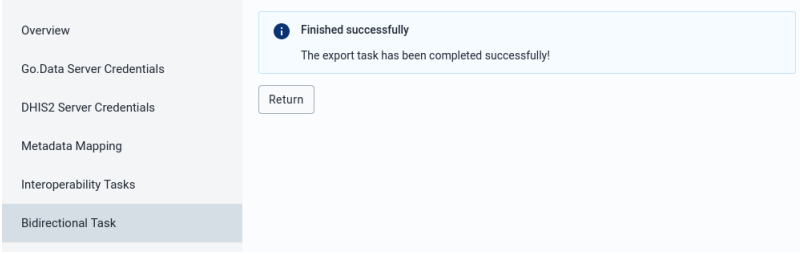

Figure C.4: Success feedback after task execution

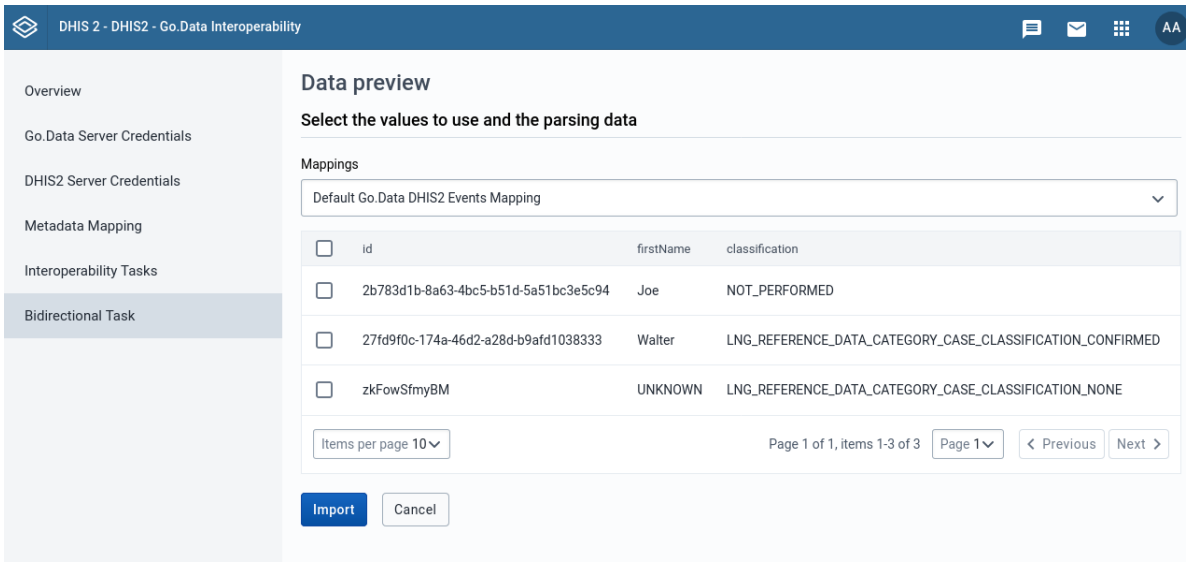

Figure C.5: New data preview with pagination and the data mapping selector.

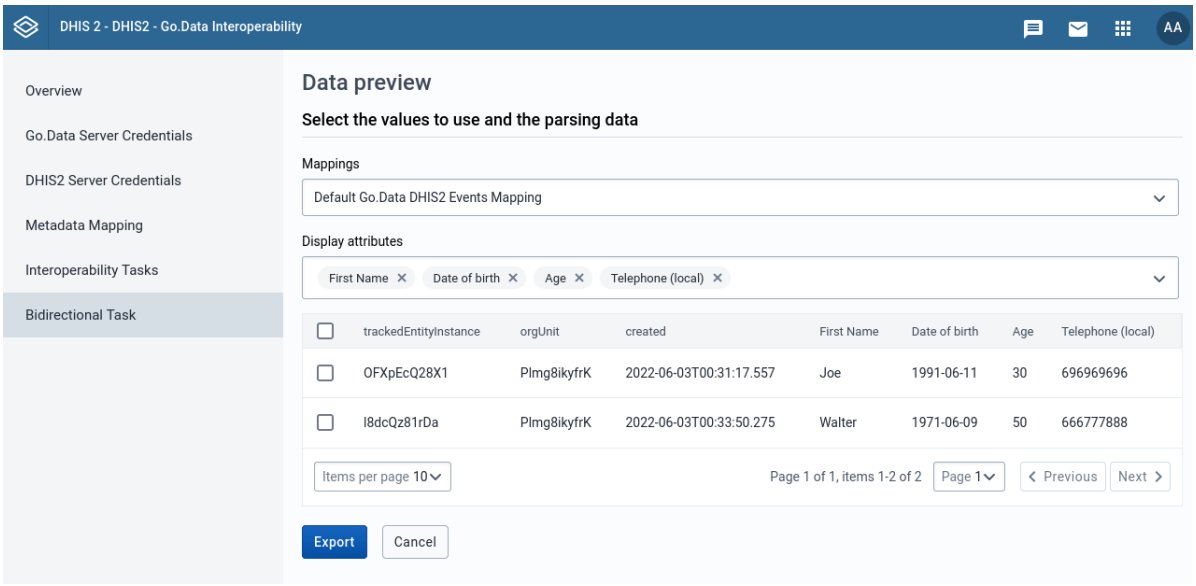

Figure C.6: Export preview. Flexible visualization. First Name, Date of Birth, Age and Telephone (local) selected.

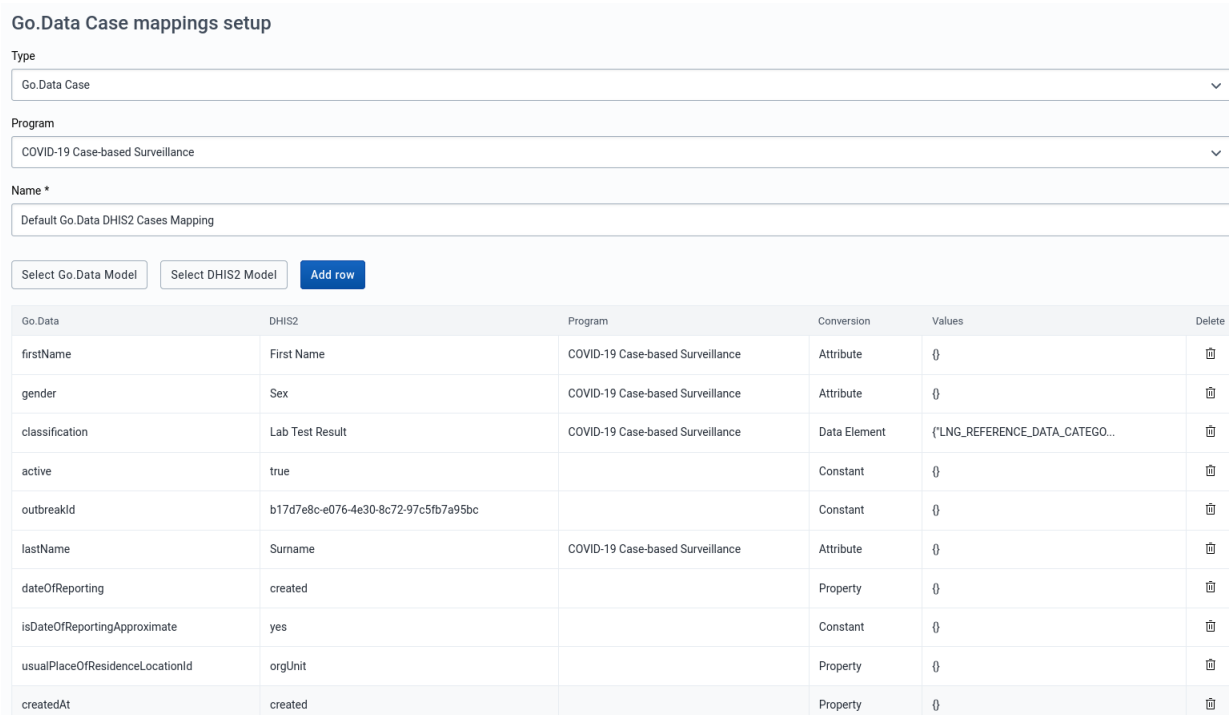

Figure C.7: Data mappings view. Case type. Autogenerated.

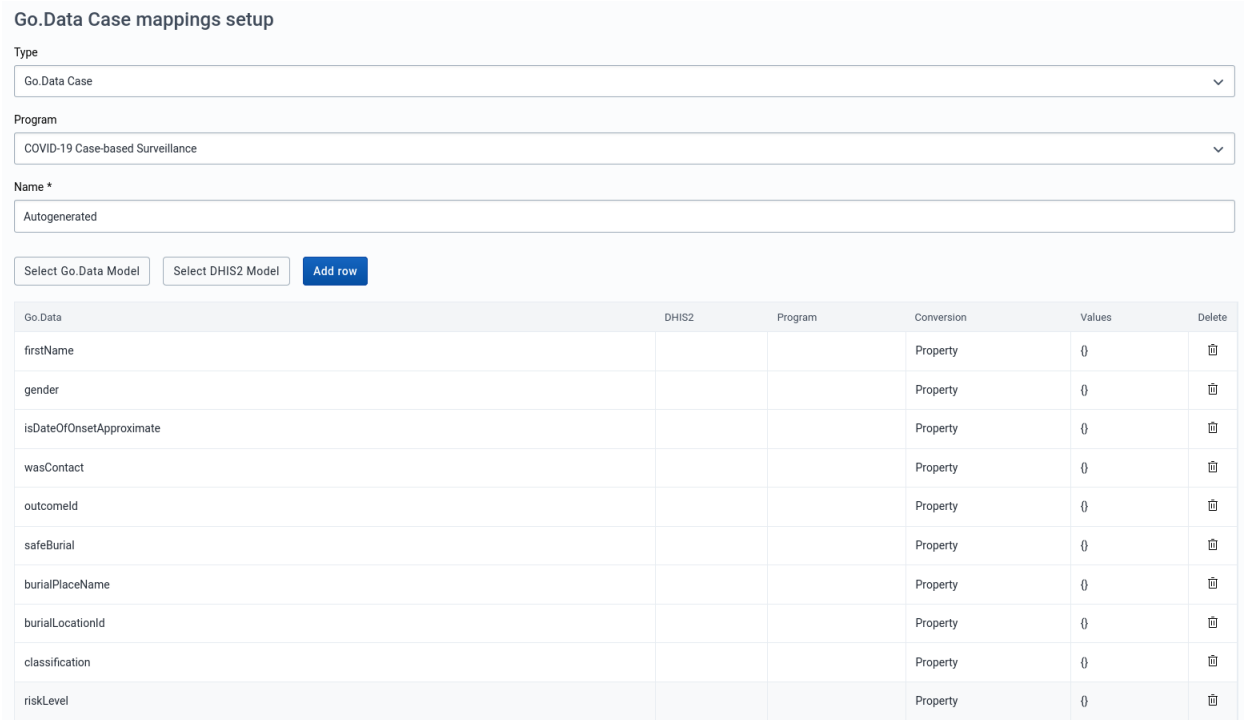

Figure C.8: Autogenerated case mapping template.

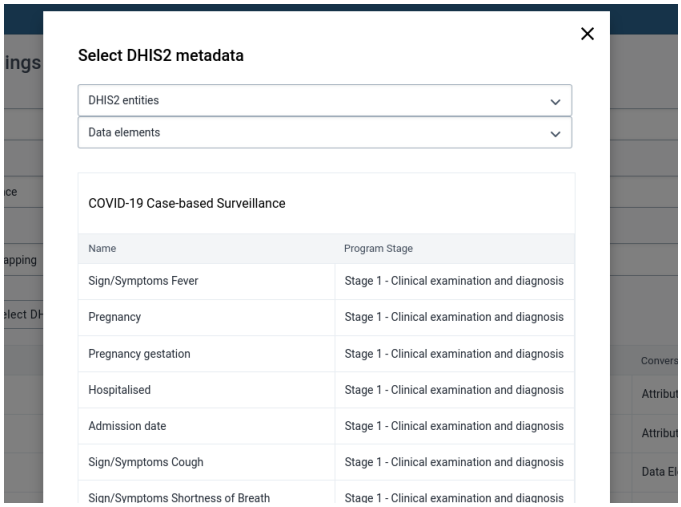

Figure C.9: Select DHIS2 metadata value modal to be mapped.

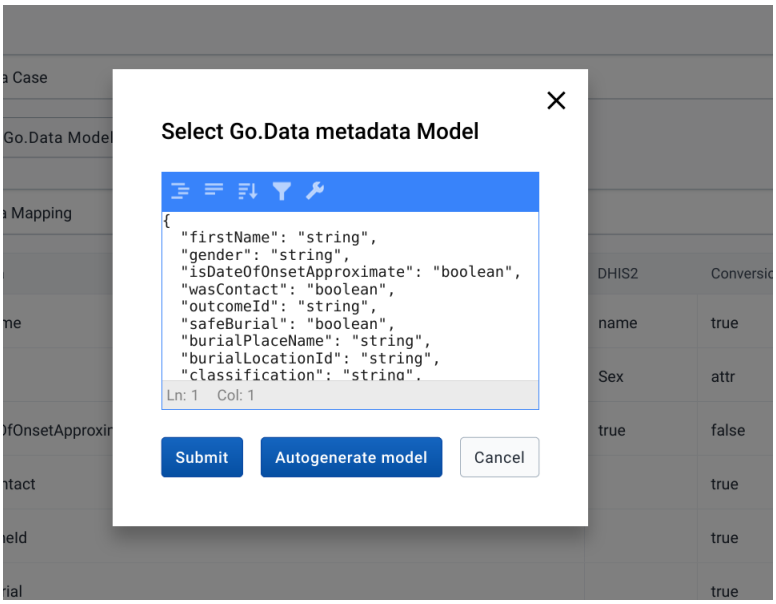

Figure C.10: Select Go.Data modal. *Submit*, *Cancel* and *Autogenerate* buttons.

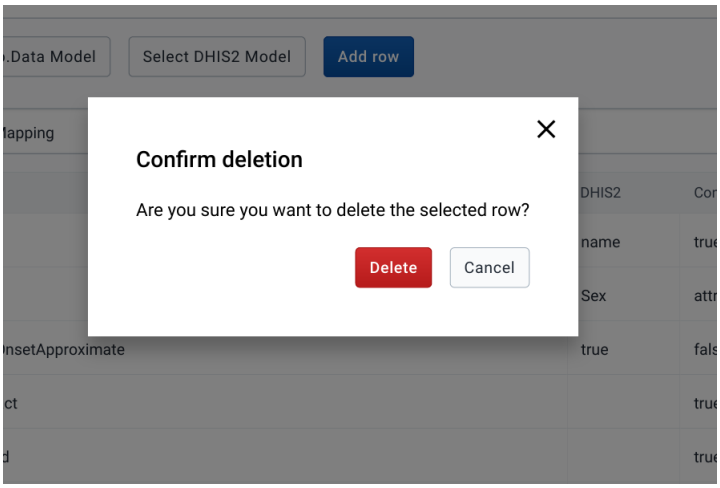

Figure C.11: Modal: *Are you sure you want to delete the selected row?*

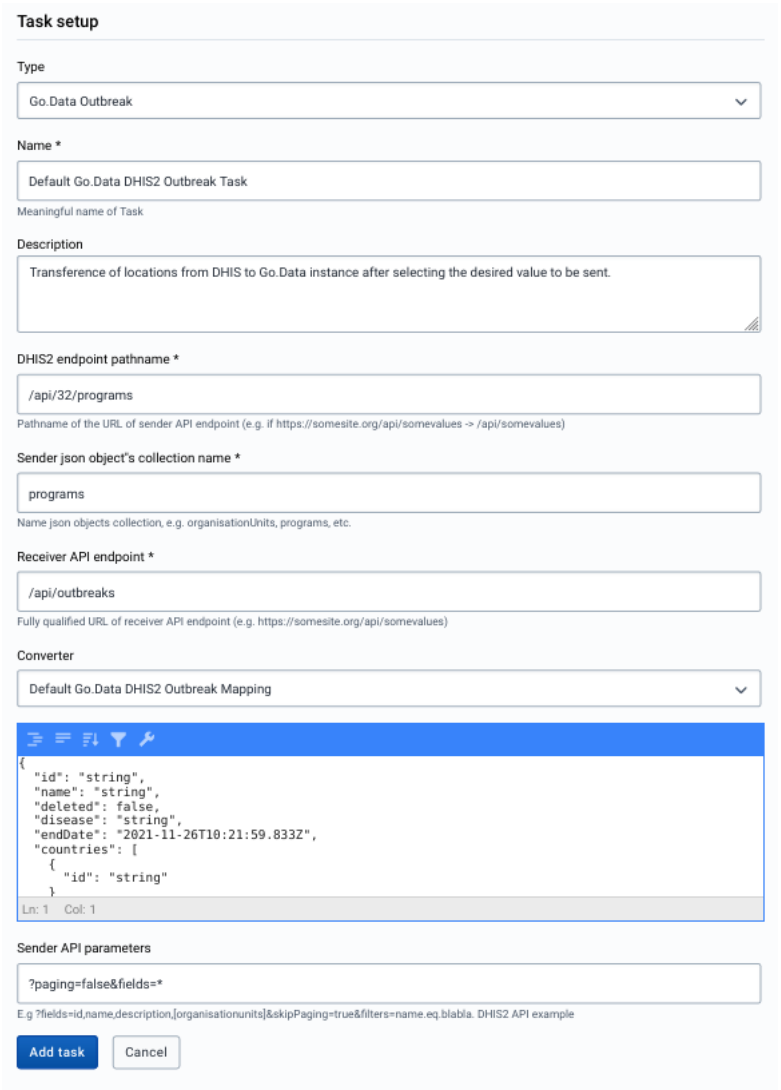

Figure C.12: Go.Data Outbreak Task Form.

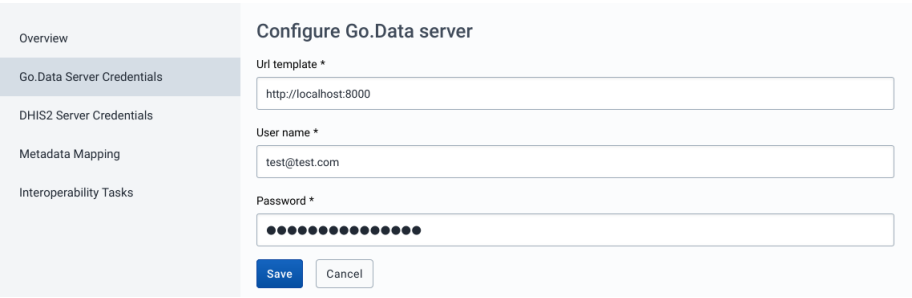

Figure C.13: Go.Data Credentials Form.

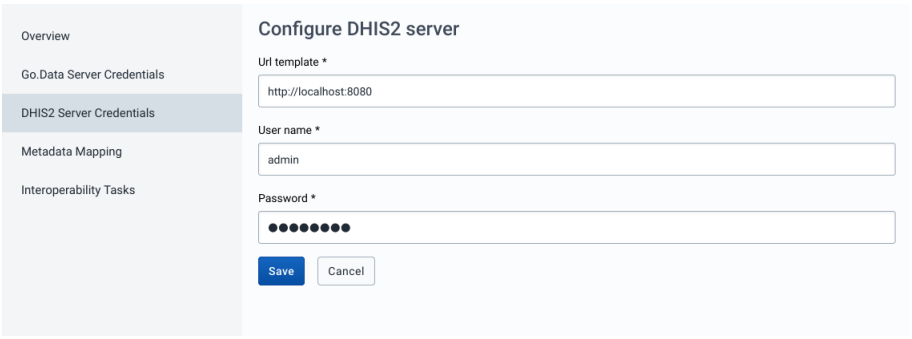

Figure C.14: DHIS2 Credentials Form.

C.1 New most relevant algorithms. Source Code.

C.1.1 Algorithm for data exporting from DHIS2

```
getDataElementsFromEnrollments = (programs) => {
  const dataElements = {};
  for (const { events } of programs)
    for (const { dataValues } of events)
      for (const { dataElement, value } of dataValues)
        dataElements[dataElement] = value;
  return dataElements;
};
applyMappingExport = ({ attributes, enrollments, ...rest }) => {
  //---------------- PART 1 -----------------
  const result = {};const dataElements =
    !enrollments || enrollments.length === 0 ? {}
    : this.getDataElementsFromEnrollments(enrollments);
  //---------------- PART 2 -----------------
  if (!!godataValue) {
    this.getMapping().forEach(
      ({ dhis2, godata, props: { values, conversion } }) => {
        if (conversion === "true")
          result[godata] = dot.pick(dhis2, rest);
        else if (conversion === "false")
          result[godata] = dhis2;
        else if (conversion === "attr")
          result[godata] = attributes.find(
                (a) => a.attribute === dhis2)?.value;
        else if (conversion === "delm")
          result[godata] = dataElements[dhis2];
        if (!!values && !!ret[godata])
          result[godata] =
            this.getFromPredefinedValuesExport(
                        values, ret[godata]);
      });
  }
  //---------------- PART 3 -----------------
  dot.object(ret);
  return ret;
};
```

Listing C.1: Export data using data mappings. Github file: *./models/mapping.js*

C.1.2 Algorithm for data importing from DHIS2

```
applyMappingImport = (initialData) => {
  //---------------- PART 1 -----------------
  const result = {
    attributes: [],
    enrollments: [],
  };
  const mapping = this.getMapping()
  const orgUnit = this.getOrgRootFromMapping();
  if (!!mapping)
    //---------------- PART 2 -----------------
    mapping.forEach(
      ({
        dhis2,
        godata,
        programStageId,
        program,
        props: { values, conversion },
      }) => {
        if (conversion === "true") {
          if (!!dhis2)
            result[dhis2] = this.getFromPredefinedValuesImport(
              values,
              dot.pick(godata, initialData)
            );
        } else if (conversion === "false") {
          result[dhis2] = this.getFromPredefinedValuesImport(
                              values, godata);
        } else if (conversion === "attr") {
          const value = dot.pick(godata, initialData);
          if (!!value)
            result.attributes.push({
              attribute: dhis2,
              value: this.getFromPredefinedValuesImport(values,
                                          value),
            });
        } else if (conversion === "delm")
            ///  CHECK THE SECOND SNIPPED TO SEE THIS PART ///
    });
  return ret;
};
```

Listing C.2: Export data using data mappings. Github file: *./models/mapping.js*

```
const value = dot.pick(godata, initialData);
if (!!value && !!dhis2) {
  const dataValue = {
    dataElement: dhis2,
    value: this.getFromPredefinedValuesImport(values, value),
  };
  let enrollment = result.enrollments.findIndex(
    (e) => e.program === program
  );
  if (enrollment < 0)
    enrollment =
      result.enrollments.push({
        orgUnit: orgUnit,
        program: program,
        events: [],
      }) - 1;

  if (!result.enrollments[enrollment].events)
    result.enrollments[enrollment].events = [];

  let event = result.enrollments[enrollment].events.findIndex(
    (e) => e.program === program
            && e.programStage === programStageId
  );
  if (event < 0)
    event =
      result.enrollments[enrollment].events.push({
        program: program,
        orgUnit: orgUnit,
        eventDate: new Date(),
        programStage: programStageId,
      }) - 1;

  if (!result.enrollments[enrollment].events[event].dataValues)
    result.enrollments[enrollment].events[event].dataValues = [];

  result.enrollments[enrollment].events[event].dataValues.push(
    dataValue
  );
}
```

Listing C.3: Second Part. Export data using data mappings. Github file: *./models/mapping.js*

Appendix D

WHO Global Consultation on Go.Data interoperability Images

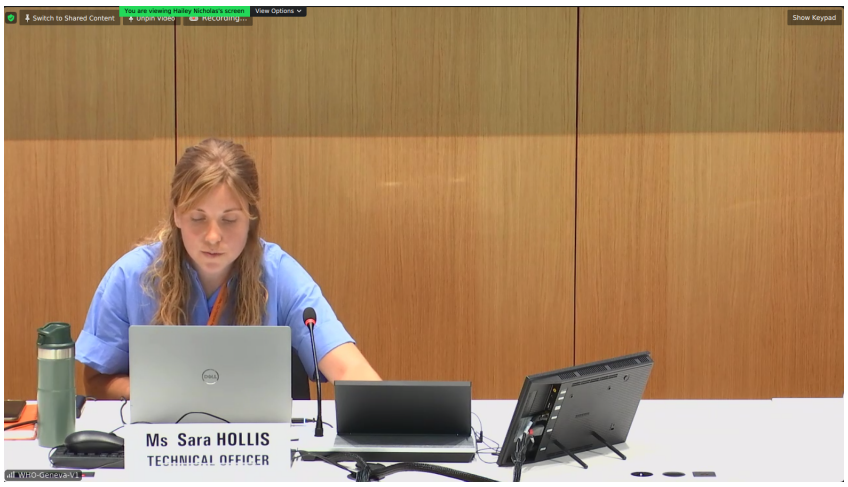

Figure D.1: Ms Sara Hollis on her speaking time.

Figure D.2: Full shot of the meeting (WHO team).

Figure D.3: Icebreaker phase. Everyone marked where he was from at the map.

Bibliography

[1] "Dhis2 data store." [Online]. Available: https://docs.dhis2.org/en/develop/using-the-api/dhis-core-version-236/data-store.html

[2] G. Basso, "Covid-19 sp: Uti v. nova cachoeirinha," May 2020. [Online]. Available: https://www.flickr.com/photos/gustavocb/49913737837/in/photostream/

[3] "Go.data official website." [Online]. Available: https://www.who.int/tools/godata

[4] "Dhis official website." [Online]. Available: https://dhis2.org/

[5] WISCENTD-UPC, "Go.data - dhis2 interoperability app github." [Online]. Available: https://github.com/WISCENTD-UPC/godata-dhis2-interoperability-app

[6] "Glassdor, sueldos en españa." [Online]. Available: https://www.glassdoor.es/index.htm

[7] W. Wyman, "The black plague," *The North American Review*, vol. 164, no. 485, pp. 441–452, 1897.

[8] "Who coronavirus (covid-19) dashboard." [Online]. Available: https://covid19.who.int/

[9] A. Ruiz de Gopegui Aramburu. [Online]. Available: http://www.exteriores.gob.es/representacionespermanentes/oficinadelasnacionesunidas/es/quees2/paginas/organismosespecializados/oms.aspx

[10] "Goarn official website." [Online]. Available: https://extranet.who.int/goarn/

[11] "Goarn partnership in outbreak response." [Online]. Available: https://www.who.int/csr/outbreaknetwork/goarnenglish.pdf

[12] P. Teixeira, *Professional Node. js: Building Javascript based scalable software.* John Wiley & Sons, 2012.

[13] C. Gackenheimer, "What is react?" in *Introduction to React*. Springer, 2015, pp. 1–20.

[14] R. Panadero, "Go.data - dhis2 interoperability app." [Online]. Available: [RemovedthepublicURLtoavoidconfusionsfortheAppusers]

[15] WorldHealthOrganization, "Godata-dhis2-interop-app github." [Online]. Available: https://github.com/WorldHealthOrganization/godata-dhis2-interop-app

[16] P. por Rosa Fernández and . m. 2022, "Electricidad: Precio medio final españa 2010-2022," Mar 2022. [Online]. Available: https://es.statista.com/estadisticas/993787/precio-medio-final-de-la-electricidad-en-espana/

[17] "Cors policy." [Online]. Available: https://cloud.google.com/apigee/docs/api-platform/reference/policies/cors-policy

[18] "Dhis2 documentation." [Online]. Available: https://docs.dhis2.org/en/

[19] "App hub submission guidelines." [Online]. Available: https://developers.dhis2.org/docs/guides/apphub-guidelines/#app-icon

[20] "react-json-view." [Online]. Available: https://www.npmjs.com/package/react-json-view