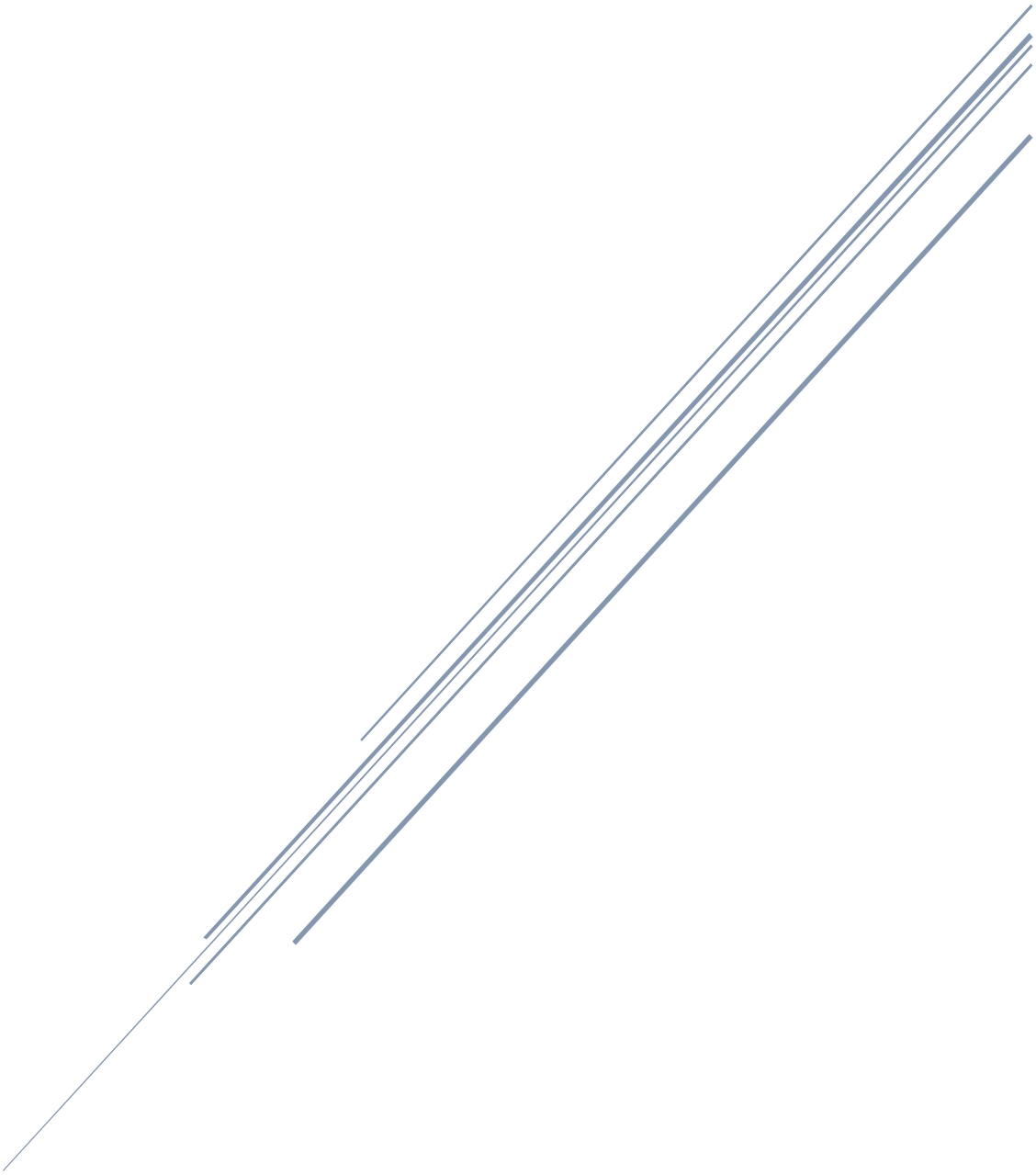
CODA

Eike Blomeier, Gil Salvans Torras



Paris Lodron University Salzburg

SDI Services Implementation (Winterterm 2020/2021)

Abstract

This project consists of the development of a Spatial Data Infrastructure (SDI) with a thematical focus on the Covid19 pandemic in three countries: Austria, Germany, and Spain. In agreement with this, the infrastructure is continuously running and its data is updated in a daily basis automatically so the user can get a general understanding of the status of the pandemic in each of the aforementioned countries with the latest data through an interactive web application with different dashboards. To achieve this, this SDI can be divided in three stages. The first one, which regards to the daily data collection and setting it up into a geospatial database. Secondly, connecting the database to a GI Server to publish all the data as standard OGC services. Finally, a retrieval of the different services is carried out by the different dashboards of the web application.

Key words: SDI, Covid19, services, Austria, Germany, Spain, dashboards

Table of Contents

Introduction

CODA, standing for Covid-19 Dashboards, aims to develop a Spatial Data Infrastructure (SDI) regarding the Covid19 pandemic development in near real time for Austria, Germany and Spain. Its information is scrapped in a daily basis from the three country data sources to serve the current situation in each of the aforementioned countries accordingly. This information is conditioned to be represented in an interactive manner and therefore, the targeting audience is the general public. With this, the user gets a quick and intuitive overview of the current pandemic situation in each specific country. Additionally, the different output dashboards will also help the users in their daily life decision making regarding their current personal health conditions and their daily routines.

In technical terms, the entire SDI architecture has been self-developed by the project authors; backend & frontend. On one hand, for the backend, the most important pillars are the data scrapping, processing and storing, and the data sharing through OGC standard services. On the other hand, the frontend is led by the web application development which obviously retrieves interactively all the previous services and data.

Once the project has been briefly and generically introduced, it is time to focus on the objectives.

Main Objectives

- Retrieve Covid-19 infection numbers and store them into a spatial database.
- Use the processed and stored data to generate a web application with interactive dashboards which display the current Covid-19 infection numbers for the different countries (and its regions) and showing different graphical data representations.
- Make all the data (published services) displayed by the dashboards publicly available to be accessed by an HTTP request from any user. Also, obviously, make the web application also universally accessible through the WWW.
- Use Geo Server technologies to carry out a map representation of the data and use it subsequently in the different dashboards.

Sub Objectives

- Develop the entire project by merely using open source technologies.
- Provide additional sources of information to the dashboard in an interactive way (e.g. Telegram / Worldwide restrictions per country).

Non fulfilled goals

- Integrate Twitter data into the different dashboards of the web app.

Reason: Due to time constraints this goal has not been achieved.

Non-goals

- Non-guaranteed data validity.
- Use of commercial software technologies.
- Provision of additional database editing functions compared to the base functions of a spatial database.

Architectural Overview

The architecture of the CODA-SDI contains many different components to build the application (Figure 1). The whole architecture again can be subdivided into four subcategories. At first, R-scripts are running once a day so scrape the latest available Covid-19 data for Austria, Germany, and Spain. Moreover, the scripts are writing the scraped data into different PostgreSQL-databases. Secondly, a GeoServer is connected to the database and generates multiple WFS per country. The WFS again, are used by two different applications. The dashboard itself. In order to display the latest developments of the pandemic to the user. And a Node-Red script which processes and sends the data via subscribable Telegram-Bots. The following chapters will explain the different parts of the architecture in more detail.

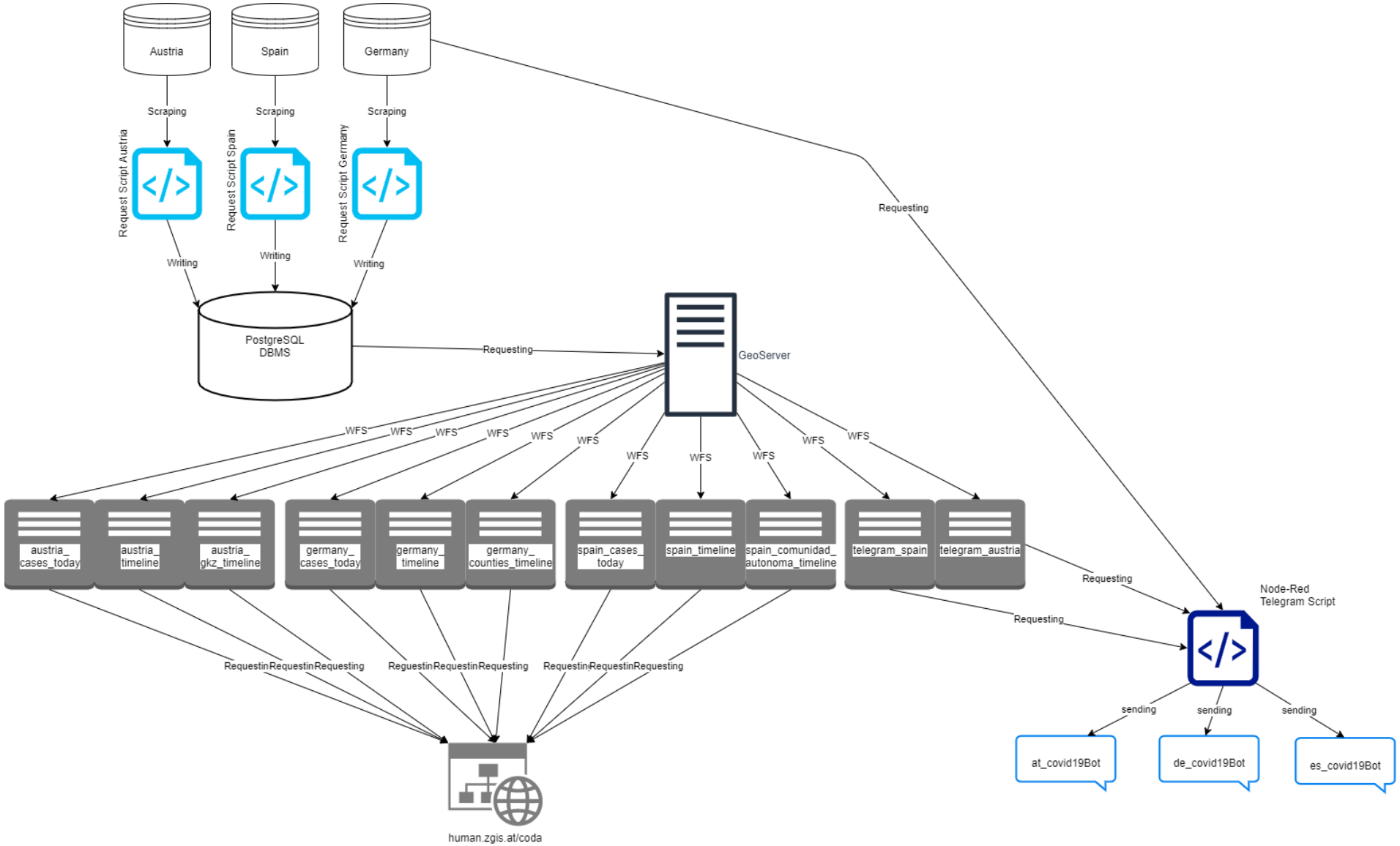


Figure 1 Architectural Diagram

Work Package 1 – Data Retrieval and Storage

Data Retrieval

R-Scripts are used to automatically scrape the latest data per country and to store the data into a PostgreSQL database. The scripts are running as a scheduled task on a Windows 2012 R2 Server once every night. Figure 2 shows the design of the scripts. The used sources for the countries are shown in the following table:

Austria	Germany	Spain
https://covid19-dashboard.ages.at/data/CovidFaelle_Timeline_GKZ.csv	https://services7.arcgis.com/mOBPykOjAyBO2ZKk/ArcGIS/rest/services/Covid19_RKI_Sums/FeatureServer/0	https://media.githubusercontent.com/media/microsoft/Bing-COVID-19-Data/master/data/Bing-COVID19-Data.csv
https://covid19-dashboard.ages.at/data/CovidFaelle_GKZ.csv	https://services7.arcgis.com/mOBPykOjAyBO2ZKk/ArcGIS/rest/services/RKI_Landkreisdaten/FeatureServer/0	

Table 1 Data sources

While the data for Austria and Germany are the official governmental numbers, the data from Spain is crowdsource and therefore less well maintained.

Before the actual scraping of the data begins, all three scripts are setting up their work-environment. Firstly, the scripts are configuring the e-mail service which is used to automatically sending e-mails to the developers informing them about the runtime result. This is done by using the *gmailr* package. Secondly, the scripts are connecting to the database using the *RPostgreSQL* and *DBI* packages. Furthermore, all three scripts are sending a notification e-mail in the end of the script that everything worked well if no errors or warnings appeared during runtime. These steps are the same in each of the scripts. In the upcoming sub-chapters, the scraping process per script is explained in more detail.

Austria

After setting up the environment for Austria, the script loads the timeline Covid-19 data for Austria on county-level. This is done in a tryCatch function to prevent the script from crashing if any warning or error appears. In the next step, the today's data for Austria is loaded on county-level. Again, the operation is performed in a tryCatch function to prevent the script from crashing if any problems are appearing while loading the data. Since, both datasets are encoded in a csv-file, the built in *read.csv* function is used to retrieve the data. After loading the data, it is stored into the *covid19_austria* table in the database. To do so, the *dbWriteTable*

function is used to write the data into the table. For both datasets, the writing process is surrounded by a tryCatch function to, again, prevent the script from failing if anything goes wrong and to have the possibility to rollback database operations in case anything went wrong while writing the data.

Germany

After setting up the environment for Germany, the script loads the latest Covid-19 data for Germany on county level. The data is served as an *ESRI Feature Service* and to query the data, a where-clause needs to be created before querying the data using the *esri2df* function from the *esri2sf* package. This is done in a tryCatch function to prevent the script from crashing if any warning or error appears. In the next step, the script writes the Covid-19 data into the *covid19_germany* table in the database. In the next step, the seven-day incidence numbers are loaded on county level from a different *ESRI Feature Service* by using the *esrid2sf* function from the *esri2sf* package. Likewise, the first service, a where-clause is needed to query the server and querying the server is performed within a tryCatch function to prevent the script from crashing in cases any warning or error shows up. Afterwards, the data is written into the *inzidenzen_germany* table. Both database writing operations are done in a tryCatch function to prevent the script from failing if anything goes wrong and to have the possibility to rollback database operations in case anything went wrong while writing the data. To write the data into the database the *dbWriteTable* function is used.

Spain

After setting up the environment for Spain, the script loads the Covid-19 data for Spain for the autonomous communities. This is done in a tryCatch function to prevent the script from crashing if any warning or error appears. Since, the dataset is encoded as a csv-file, the built in *read.csv* function is used to retrieve the data. After loading the data, it is stored into the *covid19_spain* table in the database. To do so, the *dbWriteTable* function is used to write the data into the table. The writing process is surrounded by a tryCatch function to prevent the script from failing if anything goes wrong and to have the possibility to rollback database operations in case anything went wrong while writing the data.

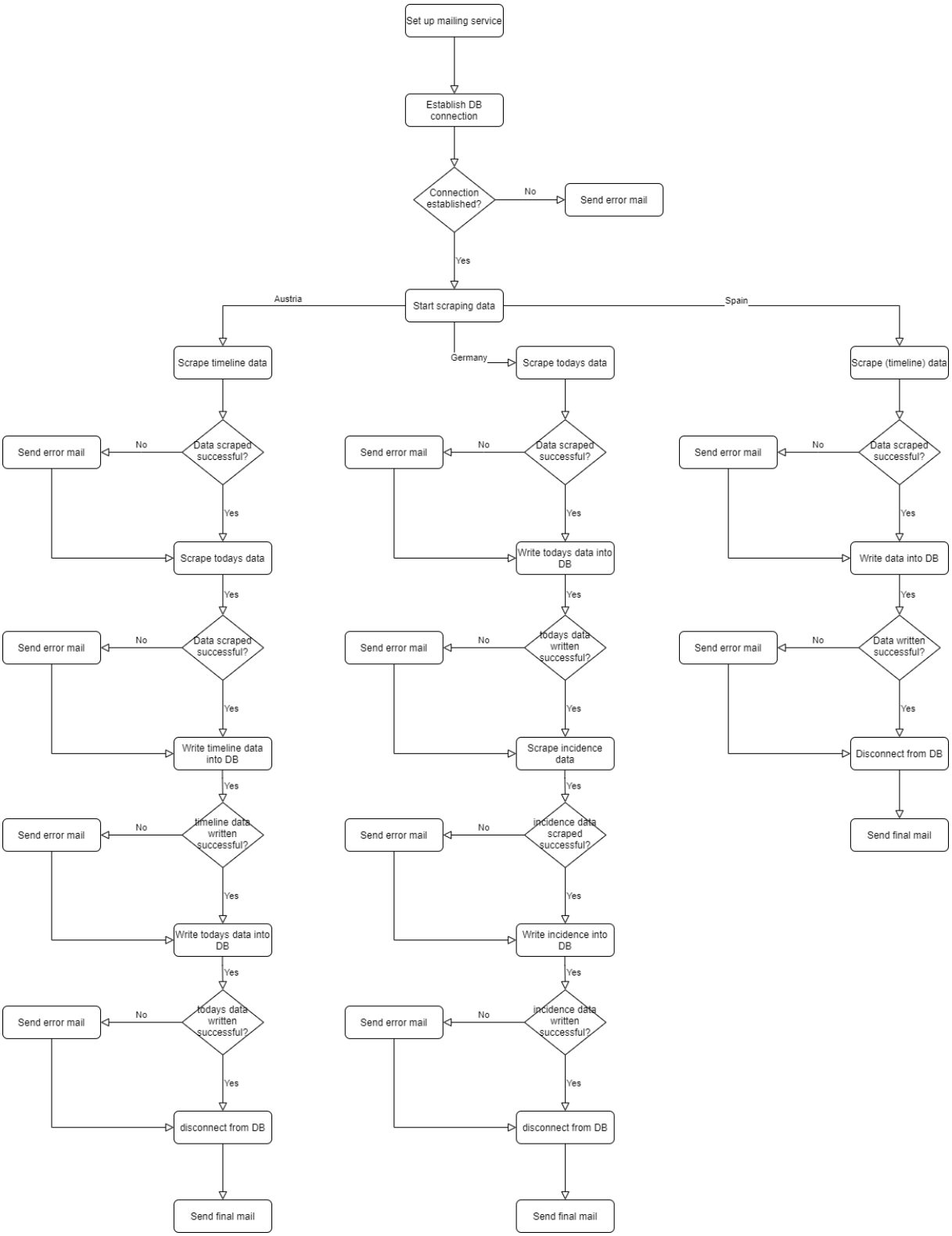


Figure 2 Scripts

Data storage

To store the data per country, a PostgreSQL with the PostGIS extension is used. For each country, the database contains a table which stores the geometries on county-level (Austria and Germany) and autonomous communities (states) level for Spain. Furthermore, the database contains tables containing the Covid-19 data and generates multiple views from the data regarding different purposes. An overview of the tables and views is displayed in Figure 3. Detailed information about the views and tables per country is provided in the following sub-chapters. Implementation details for the views can be found in the Git.

Austria

Table name	Purpose
covid19_austria	The table contains the Covid-19 data for Austria since February 2020. It contains the daily numbers per county.
austria_geoms	The table contains the geometries for each county in Austria.

Table 2 Austrian tables

View name	Composed of	Purpose
austria_cases_today	covid19_austria, austria_geoms	This view contains the geometries and the Covid-19 numbers per district for the latest available date.
austria_gkz_timeline	covid19_austria, austria_geoms	This view contains the summed-up cases per day and county.
austria_timeline	covid19_austria, austria_geoms	This view contains the summed-up cases per day for the state of Austria.
telegram_austria	Covid19_austria	This view contains the Covid-19 numbers per county prepared for the Node-Red telegram script.

Table 3 Austrian views

Germany

Table name	Purpose
covid19_germany	This table contains the Covid-19 data for Germany since January 2020. It contains the daily numbers per county.
inzidenzen_germany	This table contains the Covid-19 incidence number for Germany since January 2020. The numbers are daily and per county.
germany_geoms	This table contains the geometries for each county in Germany.

Table 4 German tables

View name	Composed of	Purpose
germany_cases_today	covid19_germany, inzidenzen_germany, Germany_geoms	This view contains the geometries and the Covid-19 numbers per county for the latest available date.
germany_counties_timeline	covid19_germany	This view contains the summed-up cases per day and county.
germany_timeline	covid19_germany	This view contains the summed-up cases per day for the state of Germany.

Table 5 German views

Spain

Table name	Purpose
covid19_spain	This table contains the Covid-19 data for Spain since the outbreak. It contains the daily numbers per autonomous community.
spain_geoms	This table contains the geometries for each autonomous community in Spain.

Table 6 Spanish tables

View name	Composed of	Purpose
spain_cases_today	covid19_spain, spain_geoms	This view contains the geometries and the Covid-19 numbers per autonomous community for the latest available date.
spain_comunidad_autonoma_timeline	covid19_spain	This view contains the summed-up cases per day and autonomous community.
spain_timeline	covid19_spain	This view contains the summed-up cases per day for the state of Spain.
telegram_spain	covid19_spain	This view contains the Covid-19 numbers per autonomous community prepared for the Node-Red telegram script.

Table 7 Spanish views

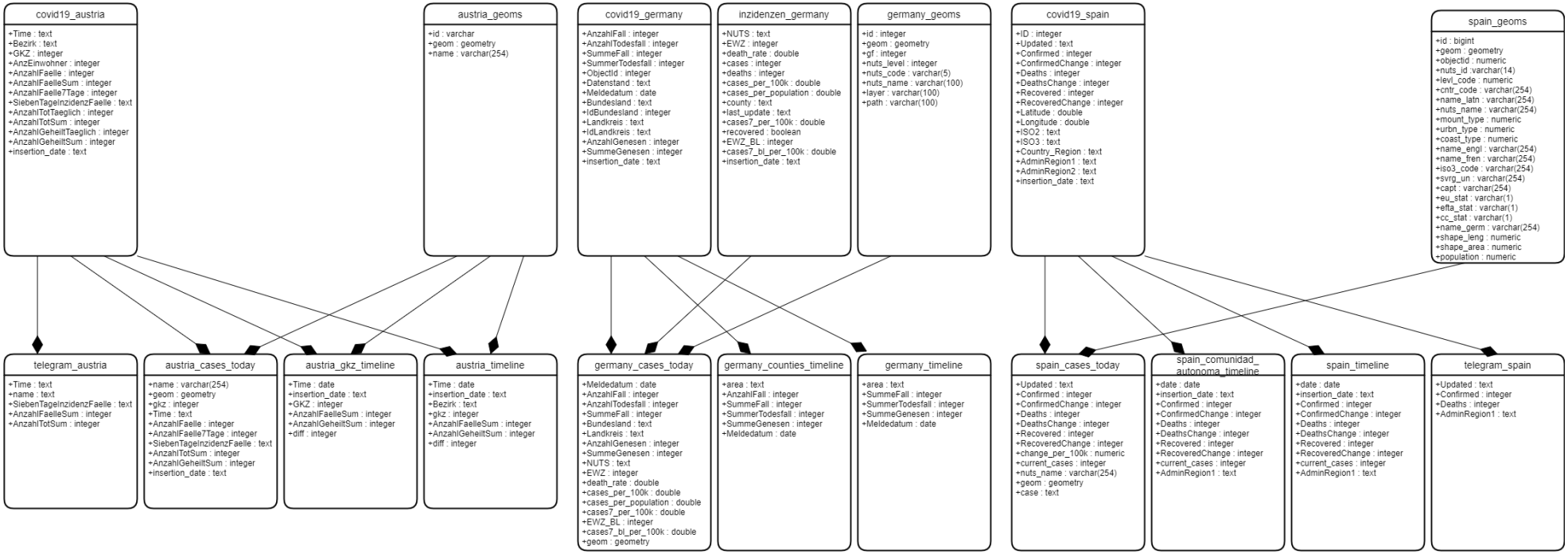


Figure 3 Database

Work Package 2 – Publishing OGC Services

To publish the data as OGC services, a GeoServer is used. The following sub-chapters providing a detailed view on the published services and the styling of the published maps. As well as, what metadata is provided for every service. Figure 4 provides a graphical overview of the shared services.

Services Austria

Service name	Data source	Purpose
austria_cases_today	austria_cases_today	This service shares a spatial WFS for Austria. The polygons are representing the geometries of the Austrian counties and their corresponding today's Covid-19 numbers.
austria_gkz_timeline	austria_gkz_timeline	This service shares a non-spatial WFS. It contains the Covid-19 timeseries data for each Austrian county since the outbreak of the pandemic until today.
austria_timeline	austria_timeline	This service shares a non-spatial WFS. It contains the Covid-19 timeseries data for the state of Austria since the outbreak of the pandemic until today.
telegram_austria	telegram_austria	This service shares a non-spatial WFS. It contains the Covid-19 numbers per Austrian county to be used by the Node-Red telegram bot.

Table 8 Austrian services

Services Germany

Service name	Data source	Purpose
germany_cases_today	germany_cases_today	This service shares a spatial WFS for Germany. The polygons are representing the geometries of the German counties and their corresponding today's Covid-19 numbers.
germany_counties_timeline	germany_counties_timeline	This service shares a non-spatial WFS. It contains the Covid-19 timeseries data for each German county since the outbreak of the pandemic until today.
germany_timeline	germany_timeline	This service shares a non-spatial WFS. It contains the Covid-19 timeseries data for the state of Germany since the outbreak of the pandemic until today.

Table 9 German services

Services Spain

Service name	Data source	Purpose
spain_cases_today	spain_cases_today	This service shares a spatial WFS for Spain. The polygons are representing the geometries of the autonomous communities and their corresponding today's Covid-19 numbers.
spain_comunidad_autonoma_timeline	spain_comunidad_autonoma_timeline	This service shares a non-spatial WFS. It contains the Covid-19 timeseries data for each Spanish autonomous community since the outbreak of the pandemic until today.
spain_timeline	spain_timeline	This service shares a non-spatial WFS. It contains the Covid-19 timeseries data for the state of Spain since the outbreak of the pandemic until today.
telegram_spain	telegram_spain	This service shares a non-spatial WFS. It contains the Covid-19 numbers per Spanish autonomous community to be used by the Node-Red telegram bot.

Table 10 Spanish services

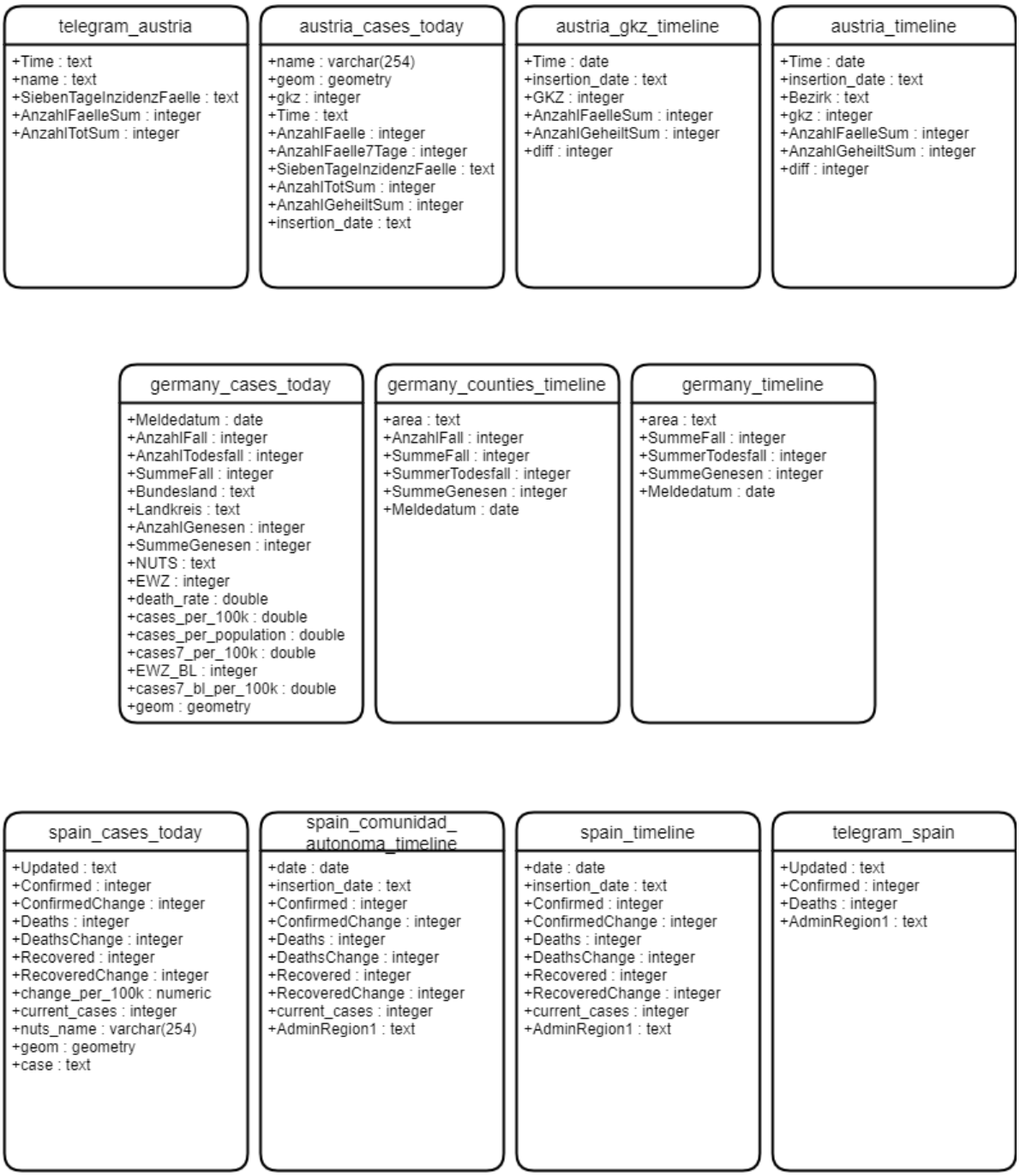


Figure 4 Shared services

Styling

Regarding styling of the shared services, most of them have been shared as WFS and therefore, the styling has been carried out in the frontend JavaScript code as functions. However, each dashboard legend has been requested using WMS requests (GetLegendGraphic) to automatically get each country's legend in a quick and efficient manner. According to this, each country's daily cases service has been styled with its own SLD (Styled Layer Descriptor – xml based) document within the GeoServer.

Figure 5 Database connection with GeoServer

Metadata sharing

The last subsection of this second work package regarding data and services sharing has been the metadata elaboration. This has been carried out using the GeoNetwork server from the university. Two metadata standards have been used: The ISO-19139 for the vector datasets and the ISO-19139/119 for the shared OGC services (WMS/WFS). In these, the user can find all the information about the data used and shared, its providers, contact, spatial extent, etc. Basically, a complete metadata catalogue per each shared dataset/service (obviously, xml-based). For more information, find the metadata on the CODA GitLab repository, in the metadata folder.

Work Package 3 – Dashboard Elaboration

This third word package encompasses all the frontend work. It consists of the development of web application which contains the different interactive dashboards per each of the countries, worldwide measures dashboard and telegram bots. To do so, each of the country dashboards contains a web map displaying the latest Covid 19 data, a table (filterable and with a search bar) and a time series plot. For Austria and Germany, the last seven days incidence is displayed on the respective maps, and for Spain –since the data quality is very variable- the current cases are displayed. Pop-up interactions are also enabled in each map and for each clicked district, the time-series chart gets adapted to it and auto-scaled automatically. For the worldwide measure's dashboard, further information can be found in the subsequent work package (Adding Extras) in the second sub part.

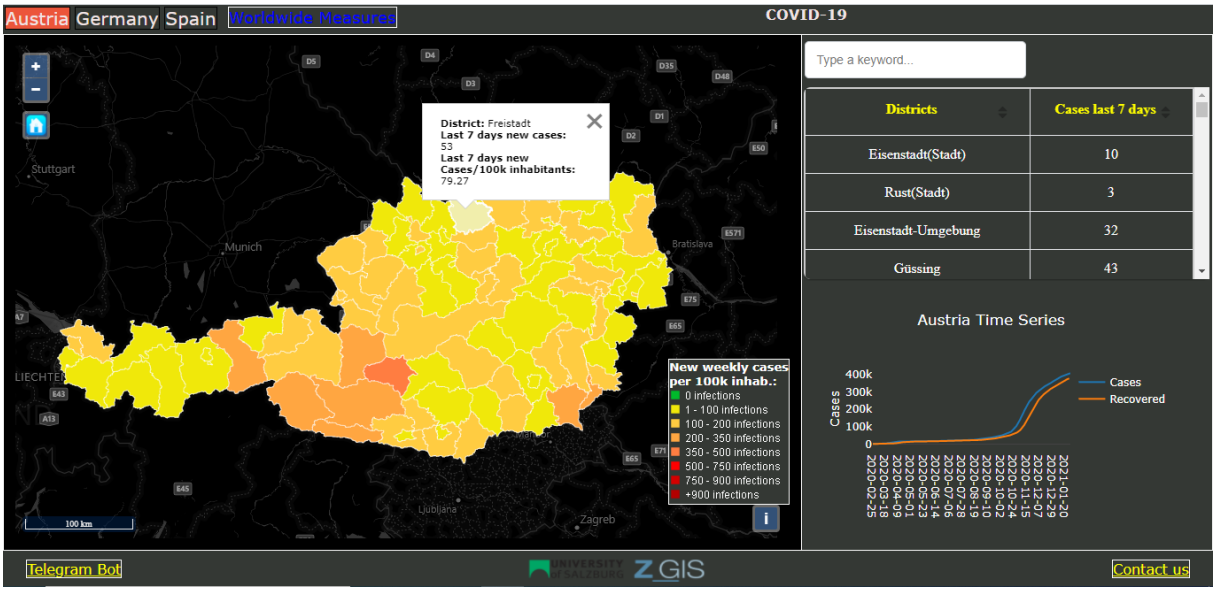


Figure 6 Web Application screenshot showing the dashboard for Austria

In technical terms, the frontend has been developed by using NodeJS (JavaScript server-side runtime environment) and Parcel to bundle the web app. Therefore, it has been deployed on a production server from the university to guarantee a universal access via internet. In agreement with this, the upcoming Table 11 specifies which libraries and node modules have been used.

Library	**Used for**
OpenLayers v.6.4.3	Web Maps development and geospatial matters
Ol-ext	Extension plugin of OpenLayers used for popups
Grid JS	Tables elaboration
Plotly JS	Time Series charting

jQuery	JSON data handling (requested using AJAX) -> for non-geospatial matters (tables & plots)

Table 11 Used JS libraries

Regarding deployment, the application has been deployed in a production server from the university as mentioned before and the web server technology used has been an Apache Tomcat 9. According to this, the application can be universally accessed through the internet using the following link: http://human.zgis.at/coda .

Work Package 4 – Adding Extras

Telegram Bot

To provide information to users with no need to open the dashboard, for each country a telegram bot was implemented. Users can subscribe to these bots to get a daily notification of the current numbers for the subscribed country.

Country	Bot name
Austria	at_covid19Bot
Germany	de_covid19Bot
Spain	es_covid19Bot

Table 12 Telegram bots

To send out messages via the bots, a Node-Red workflow is set up to retrieve latest Covid-19 information, format the information into a message and send the message via the bot for each country. The implementation can be reviewed in the project Git. Figure 7 shows the graphical overview of the Node-Red implementation workflow.

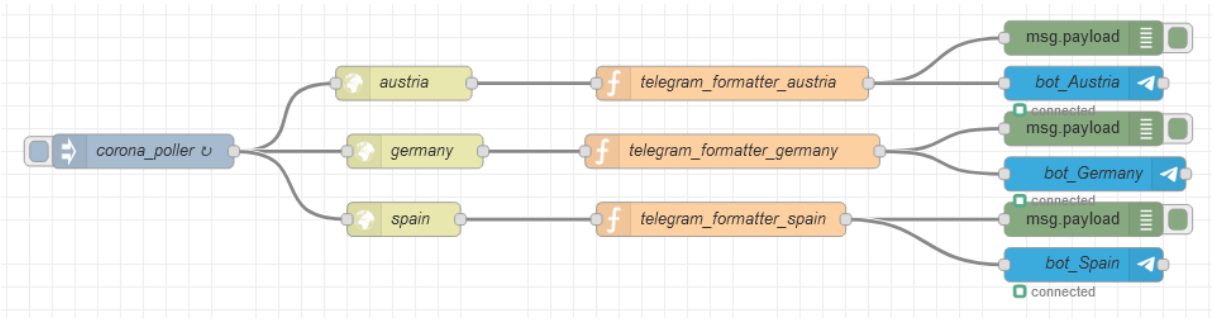


Figure 7 Node-Red Workflow

World Measures

The second section of this last work package consisted of the addition of a new dashboard within the web application. In this case, the aim was not to add any country, but to add a dashboard in a world scale with all the up to date Covid19 measures by country. According to this, a Rest Feature Service from the United Nations has been used as the backbone information source of this section. This data is basically provided within the OCHA framework; a humanitarian initiative impulsed by the UN.

Regarding interface, this additional dashboard is integrated within the web application as an additional selectable tab. Once the user selects it, there is the possibility to select any country in the world and (on the side panel) get immediately its latest Covid-19 measures and restrictions. This data service is, as mentioned before, an ArcGIS Rest Feature Service and it is asynchronously retrieved by the web application using an AJAX call in order to always display the latest information without having to refresh or do anything.

Work Packages Diagram

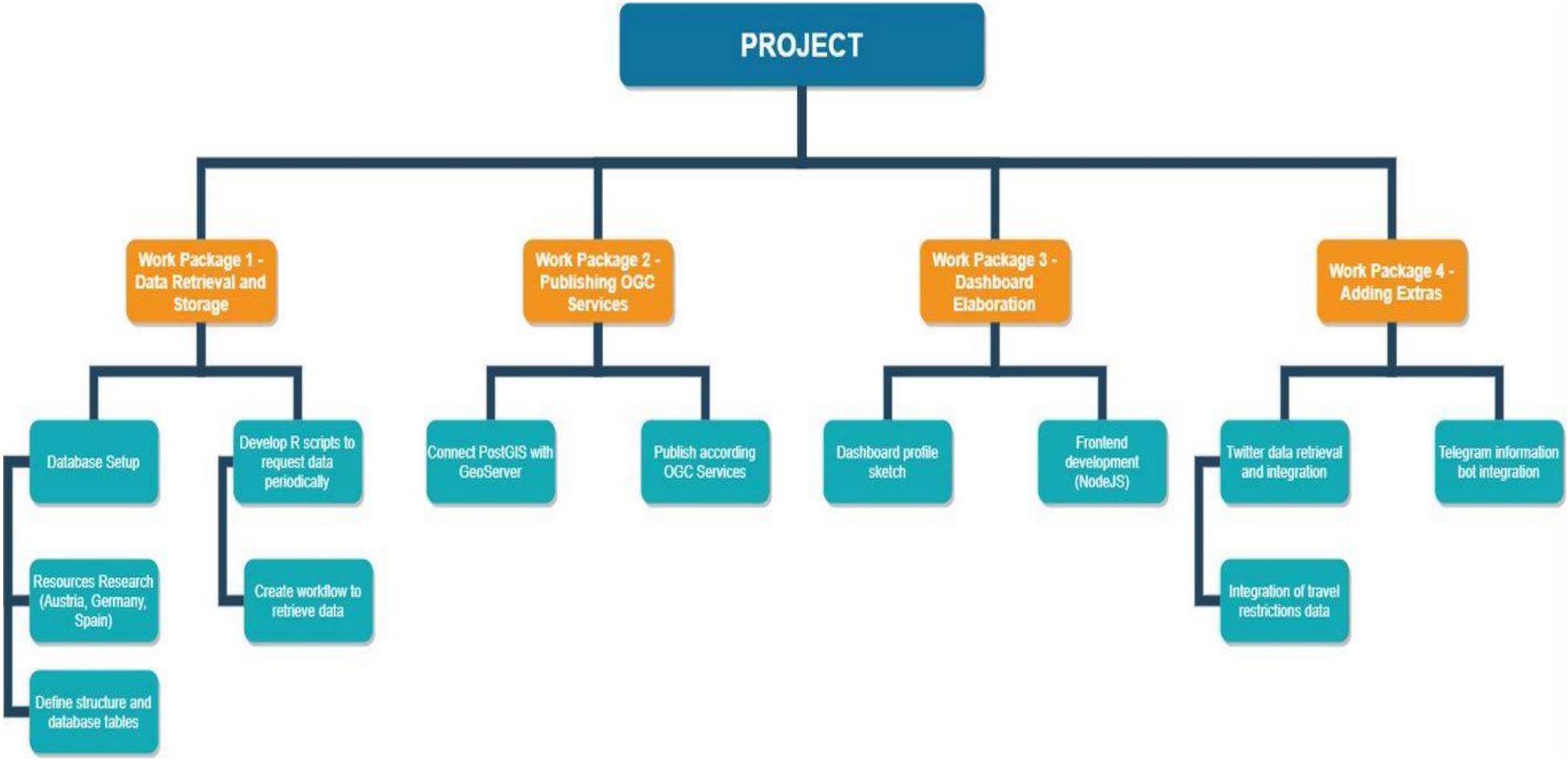


Figure 8 Work Packages

Risk Matrix

In order to find and evaluate potential risks, the upcoming risk matrix was elaborated throughout the project development.

	Risk	Mitigation Strategy	Type	Update
1	Lack of data sources	Find out sources of current Covid-19 dashboards	L	29.10.2020
2	Complex raw data structure	Have a clear data schema (regarding structure and quality) and an according enhanced retrieval process	H	29.10.2020
3	Node Red dashboard capabilities do not fit in the project purpose	Find new dashboard methodologies or even code it ourselves.	H	29.10.2020
4	Univeristy GeoServer cannot be setup	Setup our own GeoServer and host it in an external server.	L	29.10.2020

Figure 9 Risk matrix

Conclusion

Even though, not all goals could be fulfilled in the end, the project was still a success. We were able to setup our own SDI using open source technologies. We managed to implement our own website from scratch, without using pre-build templates and host it on a production server from the university. Hosting the dashboard on a university server has the advantage that the public can access the application 24/7 to get the latest information for their needs. Since, Covid-19 is an ongoing pandemic which affects people from all over the world, implementing this dashboard was not only challenging us to proof and improve our software development skills, the dashboard is also useful for society. A big challenge in the beginning was to find services which are fulfilling our needs. A big issue there was, that every country hosts its own data without any standardization. For consequence, a data harmonization between Austria, Germany and Spain was a shortcoming.

To conclude, the project was very useful because we had to face the entire stack of developing an SDI. According to the objective of the course have been accomplished.