Manual

Internet of Kegs by PUM12

May 2023

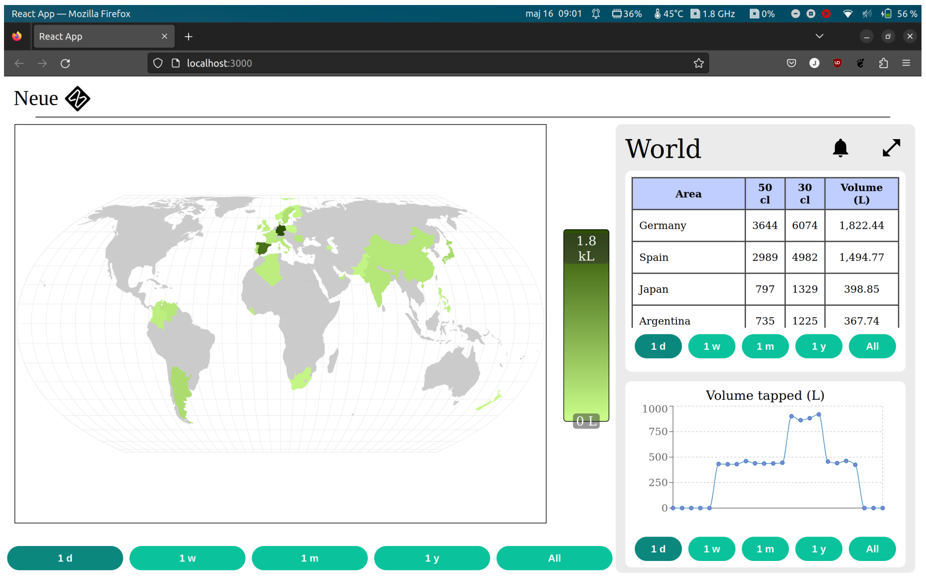
1 How to use the application as a user

In the React frontend, the user is first greeted with a map view of the world, which displays the volume of dispensed beer in each country through color coding, which can be seen in Figure 1. If there is no data for a location, it appears in gray, otherwise the location is displayed according to the color gradient. The color gradient ranges from 0 to the highest volume found on the map. Below the map, there are five buttons for a day, a week, a month, a year, and all collected data for the current view. If the user clicks on one of the buttons, the data for the map and gradient are replaced with the corresponding time interval's data. At the top left of the view, Neue's name and logo can also be seen. If the user clicks on it, regardless of the current view, the page reloads to the home view.

To the right of the map, more information about the current view can be seen. At the top, it shows which view the user is currently in. In Figure 1 the location is World, so the name “World” is displayed. In this side view, there is a table and a graph showing volume data. The time interval for both of these can be changed using the buttons underneath them, just like for the map. To the right of ”World”, two buttons can be seen: one for alarms and one to expand this detail view.

If the user presses the expand button, they will be taken to the expand view seen in Figure 2. The expand button is now replaced with a minimize button instead, which will take the user back to the previous view. Here, a more detailed view for the view on the level the user is currently on can be seen. In this example, the detail view for ”World” is shown, but if the user had navigated to Germany before pressing the expand button, Germany's detail view would have been displayed, which will be shown later.

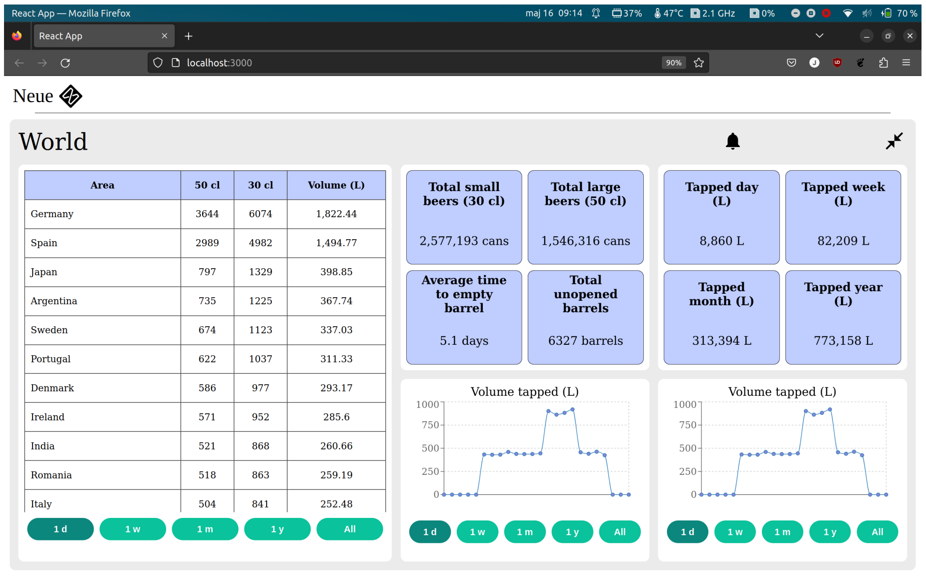
In this detailed view, the same table and graph from before are shown, along with eight overview boxes and an additional graph for comparison reasons. Once again, the time interval can be changed for the table and graphs. In the two upper left overview boxes, the user can see how much beer has been dispensed in total, represented in 30cl and 50cl glasses. In the the two lower left overview boxes, the average time to empty a keg and the number of unopened kegs can be seen. As of now these two boxes are not implemented, and the values are dummy values. In the four right overview boxes, the total amount of dispensed beer for different time intervals can be seen.



World

Area	50 cl	30 cl	Volume (L)
Germany	3644	6074	1,822.44
Spain	2989	4982	1,494.77
Japan	797	1329	398.85
Argentina	735	1225	367.74

Figure 1: Home screen



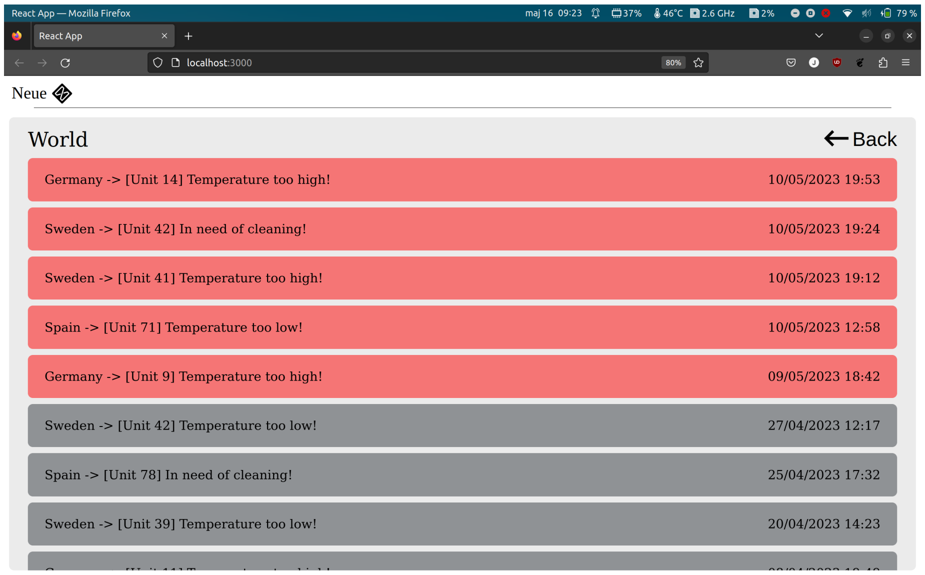
World

Area	50 cl	30 cl	Volume (L)
Germany	3644	6074	1,822.44
Spain	2989	4982	1,494.77
Japan	797	1329	398.85
Argentina	735	1225	367.74
Sweden	674	1123	337.03
Portugal	622	1037	311.33
Denmark	586	977	293.17
Ireland	571	952	285.6
India	521	868	260.66
Romania	518	863	259.19
Italy	504	841	252.48

Total small beers (30 cl) 2,577,193 cans	**Total large beers (50 cl)** 1,546,316 cans
Average time to empty barrel 5.1 days	**Total unopened barrels** 6327 barrels

Tapped day (L) 8,860 L	**Tapped week (L)** 82,209 L
Tapped month (L) 313,394 L	**Tapped year (L)** 773,158 L

Figure 2: Expand view



World ← Back

Germany -> [Unit 14] Temperature too high! 10/05/2023 19:53

Sweden -> [Unit 42] In need of cleaning! 10/05/2023 19:24

Sweden -> [Unit 41] Temperature too high! 10/05/2023 19:12

Spain -> [Unit 71] Temperature too low! 10/05/2023 12:58

Germany -> [Unit 9] Temperature too high! 09/05/2023 18:42

Sweden -> [Unit 42] Temperature too low! 27/04/2023 12:17

Spain -> [Unit 78] In need of cleaning! 25/04/2023 17:32

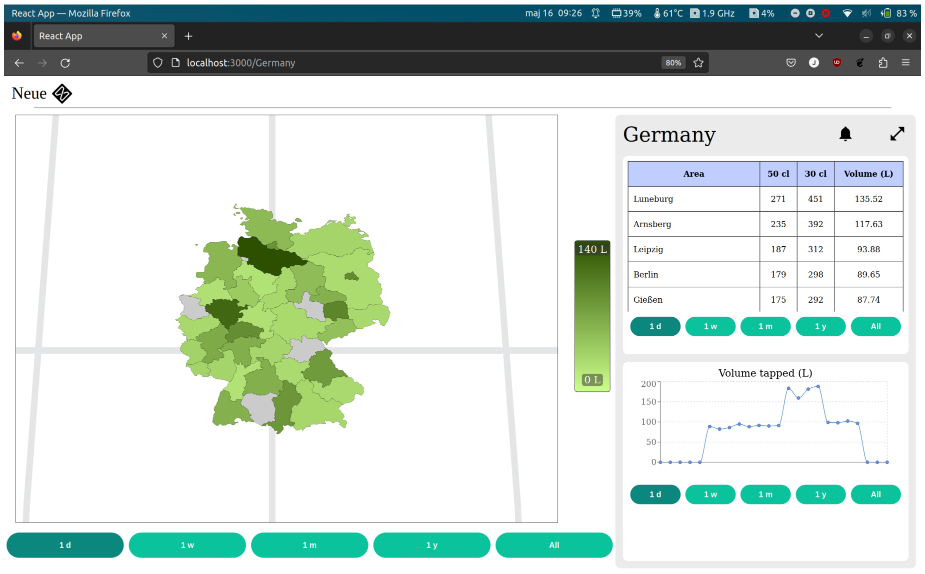
Sweden -> [Unit 39] Temperature too low! 20/04/2023 14:23

Figure 3: Alarm view

If the user presses the alarm button, they will be taken to this alarm view seen in Figure 3. This view is not completely implemented, but the general functionality is still there. Now, just like for the detail view, this view depends on the level the user was on when they clicked the button. In this example, all alarms throughout the world are shown. Active alarms are displayed in red, while handled alarms are displayed in gray. For example, it can be seen that Sweden has an active alarm on sensor unit 42, where it is in need of cleaning. To mark an alarm as handled, the user can press the alarm, and it will change from active to inactive, changing its color. Similarly, an alarm can be reactivated if it was accidentally turned off. There is also a back button at the top right of the view that takes the user back to the view they were on before.

Finally, if the user is back on the home view, they can navigate to a deeper level by clicking on a country. If the user chooses to find and click on Germany seen in Figure 4, they will be taken to this view. Here, data for Germany can be seen, and if the user chooses to press any of the alarm or expand button, only data for Germany and its deeper regions will be displayed.

As of now, this is as deep as the user can go, however more maps can be added as well as deeper map layers to display even more detailed data. A map could be added for Luneberg in Germany, for example, which displays deeper levels of administration, and so on.



Germany

Area	50 cl	30 cl	Volume (L)
Luneburg	271	451	135.52
Arnsberg	235	392	117.63
Leipzig	187	312	93.88
Berlin	179	298	89.65
Gießen	175	292	87.74

Figure 4: Germany view

2 How it works

Please read the README.md in the project folder for a more detailed view on how to install and run the project, as well as how the project is structured and how the code works.

3 How the Cloud works

In this chapter the functions used in Google Cloud will be described.

3.1 BigQuery

Most of the work in Google Cloud was performed in BigQuery. In BigQuery there is a dataset called *Testing123* where all of the tables and procedures are contained.

3.1.1 Tables

The table *RealData* contains the real time data sent from all sensors. Right now it is only filled with dummy data. It is also partitioned by unit which means querying on units will be faster and less resource intensive. Volume data is stored in millilitres, pressure in millibars and temperature in Celsius. *unit* is a sensors ID and *timestamp* is the timestamp when the measurement was made.

The tables *DataByHour*, *DataByDay*, *DataByWeek* and *DataByMonth* are supposed to contain old data aggregated from the *RealData* table. Scheduled queries are supposed to compress the data from the *RealData* table and insert it into these other tables. This is not implemented, but the idea is that old data gets moved to these tables for easier access instead of storing all data points in the *RealData* table which would otherwise get filled with millions of data points, possibly worsening performance.

The *LocationData2* table contains information about sensors geographical location. It is filled with dummy data and most of the columns are not used right now but might be useful information to know later. The *code* column contains the country code which is used in the frontend. The *layer* and *layer2* columns represent different governing layers of the country i.e. state, province, municipality, etc. Countries have a different number of layers which makes it harder to implement and right now only one layer is used in the frontend, however more may be added.

The schema used in these tables can be changed and improved to fit your future needs. If you want to insert new data into the tables you can either write a SQL query, use Pub/Sub or generate data with code and put it in a new table which can then be inserted into the desired table.

3.1.2 Stored procedures

In BigQuery there are stored procedures (routines) which are used by the fronted to fetch data from the tables. These procedures can be divided into two categories, fetching data for the tables in the frontend and data for the graphs. For each category there exists five procedures which gets data from the last 24 hours, week, month, year and all time. The procedures are linked to the time step buttons you can find in the frontend. These are the only procedures which are used but there exists some additional procedures which were used when testing ideas and might be interesting for the future.

3.1.3 Pub/Sub

A small test with Pub/Sub was set up. The topic called *RealTopic* is the one used when you want to publish data into the *RealData* table. The data flows from the topic via a subscription called *TestRealTimeData* into the *RealData* table. The Python script *Pubber.py* can be used as a publisher to publish data to the *RealTopic* topic, which then gets sent into the *RealData* table.

The schema used in the topic is called *RealSchema* which is an AVRO format of how the structure should look like. It is used when publishing to make sure the data is in the correct form.

3.1.4 Service accounts

Service accounts are used to communicate between the Cloud and the backend and frontend. The service accounts take on the roles of calling procedures and acting as publisher and subscriber. To authenticate the service accounts we used service account keys which can be found in Google Clouds IAM. These keys are not very secure and we recommend changing them often or removing them and finding a better solution.

4 Questions

If you have any questions feel free to send an email to TDDD96_2023_12@groups.liu.se and we may respond if we have the time.