The Transatlantic Scooters' Project Documentation

Laura Bakała
Wojciech Kretowicz
Karol Pysiak
Mateusz Szysz

18 October 2021

1 Deliverables

The project will focus on the relation (or lack thereof) between the weather and public transport delays. The final solution should be able to predict average delay for the following days based on the weather prognosis, traffic, and other factors. It would provide numerous advantages for both municipal administration and society. From the former's point of view, it would allow selecting an appropriate mean of transport based on predicted demand. For the latter, it would show if it's convenient to choose subway or perhaps it would be too crowded.

2 Data sources

2.1 Warsaw Open Data - online information about trams and buses

The primary data source for the project is Warsaw Open Data, an open database containing much information about Warsaw, *inter alia* public transport. It describes the position of trams and buses in real-time. This data can be found here: Otwarte dane po warszawsku.

This data can be accessed by a RESTFUL API with dedicated endpoint that returns on-line data about trams and buses. This endpoint returns a JSON file, where each record consists of following fields:

1. Lines – a line number of a given tram or a bus
2. Lon – a longitude of the position of the vehicle
3. Lat – a latitude of the position of the vehicle
4. VehicleNumer – an identification number of the vehicle
5. Brigade – an identificator of a team that operates the vehicle at given time

6. Time – a timestamp created at the moment of reading GPS data

The data is updated with the declared frequency of 1 minute. However, in practice, we observed that the data is updated with the frequency of 10 to 15 seconds. Single call to the API returns a variable number of records, up to about 300 during rush hours. Thus this data is treated by us as a stream data.

2.2 Warsaw Open Data - public transportation schedule

Moreover, we use Warsaw Open Data, available under the same link as in case of the online information about public transport (Otwarte dane po warszawsku), to obtain the schedule. This is once again a RESTFUL API. However, in this case the updates of the data are made in a much smaller frequency (schedule changes relatively rarely).

We utilize three different endpoints to collect all necessary data. All of them return a JSON file, where each record is described as a list of objects with `"key"` and `"value"` fields.

Each endpoint has a limited number of `"key"` values that can appear. These are as described below,

The endpoint with the list of bus/tram stops returns about 7000 records. The fields are as follows:

1. zespol – ID number of the complex of bus/tram stops (bus/tram stops are grouped together under one name when they are placed in the same location, usually on different sides of a road junction)
2. slupek – ID number of the exact bus/tram stop, unique for given stop complex
3. nazwa_zespolu – human-friendly name of a stop complex (e.g. “Morskie Oko”)
4. id_ulicy – ID number of the street the stop is placed on
5. szer_geo – latitude
6. dlug_geo – longitude
7. kierunek – direction the stop is facing
8. obowiazuje_od – date when the data was last changed

The endpoint with the list of bus/tram lines for given bus/tram stop returns between 0 and 20 records. It requires `zespol` and `slupek` values from the previous call. The only field appearing is `linia`, giving the line number.

The endpoint with the timetable for given bus/tram stop and line number returns up to a hundred records. It requires `zespol`, `slupek`, and `linia` values from the previous calls. The fields are as follows:

1. czas – hour of departure (i.e. hour, minute and second, but the last value is always equal to 0)
2. trasa – ID code of the travel (understood as a sequence of stops)

3. kierunek – direction of the travel

4. brygada – ID number of the team operating the vehicle

5. symbol_1, symbol_2 – additional attributes, usually empty

If it wasn't for the fact that we filter out bus stops, the volume of the data (i.e. the number of timetable entries) would be as high as a few million records. However, with filtering included, the volume of the fully processed data is about 110 000 records.

2.3 Meteorological data

The third and last data source is open meteorological data source from which we can download prognosis for the specific amount of days as well as the historical data. This is a RESTFUL API available here: Weather API.

Single call to this API returns a JSON file with following fields:

1. cod - Internal parameter

2. message - Internal parameter

3. cnt - A number of timestamps returned in the API response

4. list - List of weather predictions

    - dt - Time of data forecasted, unix, UTC
    - main
        - temp - Temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
        - feels_like - This temperature parameter accounts for the human perception of weather. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
        - temp_min - Minimum temperature at the moment of calculation. This is minimal forecasted temperature (within large megalopolises and urban areas), use this parameter optionally. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
        - temp_max - Maximum temperature at the moment of calculation. This is maximal forecasted temperature (within large megalopolises and urban areas), use this parameter optionally. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
        - pressure - Atmospheric pressure on the sea level by default, hPa
        - sea_level - Atmospheric pressure on the sea level, hPa
        - grnd_level - Atmospheric pressure on the ground level, hPa
        - humidity - Humidity, %
        - temp_kf - Internal parameter
    - weather
        - id - Weather condition id

    - main - Group of weather parameters (Rain, Snow, Extreme etc.)
    - description - Weather condition within the group. You can get the output in your language.
    - icon - Weather icon id
  - clouds
    - all - Cloudiness, %
  - wind
    - speed - Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour.
    - deg - Wind direction, degrees (meteorological)
    - gust - Wind gust. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour
  - visibility - Average visibility, metres
  - pop - Probability of precipitation
  - rain
    - 3h - Rain volume for last 3 hours, mm
  - snow
    - 3h - Snow volume for last 3 hours
  - sys
    - pod - Part of the day (n - night, d - day)
  - dt_txt - Time of data forecasted, ISO, UTC
5. city
  - id - City ID
  - name - City name
  - coord
    - lat - City geo location, latitude
    - lon - City geo location, longitude
  - country - Country code (GB, JP etc.)
  - timezone - Shift in seconds from UTC

This data is updated with the frequency of 3 hours. Single call returns 40 weather predictions. Thus it is treated by us as a batch data.

We are using the free pricing plan provided by the website. This gives us an information about the current weather conditions, a minute forecast for 1 hour, hourly forecast for 2 days, daily forecast for 7 days and historical weather for 5 days. This API is up 95% of time. We are allowed to call this API 1000 times a day or 60 times a minute.

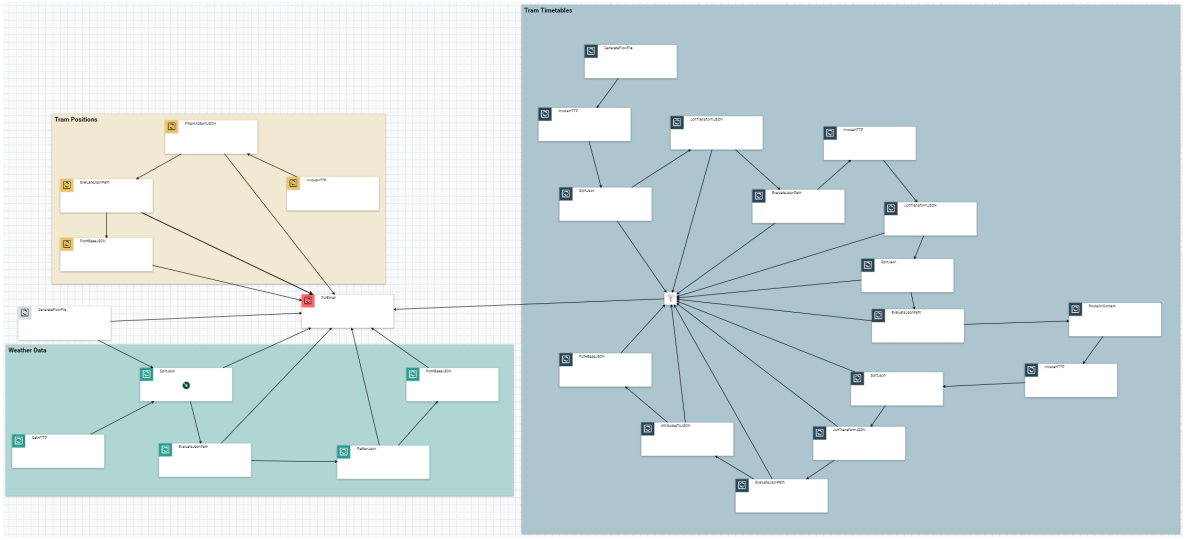

Figure 1: Schema of data processing

3 Data processing plan

The whole data processing pipeline is summarized in a figure 1.

Please, note, that a failure at any step is reported by an email.

3.1 Warsaw Open Data - online information about trams and buses

Warszawskie Dane offers access to the data in the streaming fashion. Thus, a whole pipeline of data transformation is processed as a stream.

Processors used in the data transformation are listed below:

1. InvokeHTTP – calls an API every 10 seconds in order to obtain records about trams and buses
2. FilterAndSplitJSON – filters incorrect records and splits them into separate JSON files, position that is outside of Warsaw is considered as incorrect record
3. EvaluateJSONPath – takes out required fields from each record
4. PutHBaseJSON – puts each record into HBase

This is visualized on a figure 2.

3.2 Public transport schedule

Processors used in this part of data processing have following tasks:

1. GenerateFlowFile – saves an API key to an attribute for later API calls

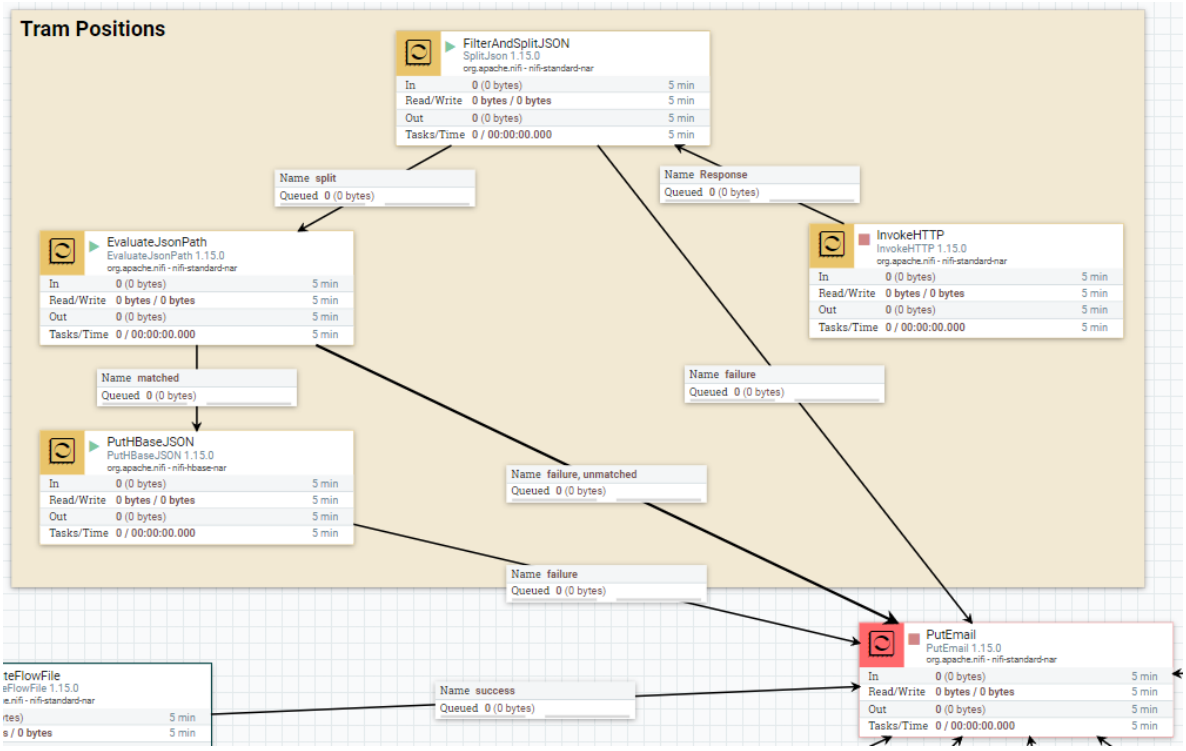

Figure 2: Schema of processing trams and buses positions

2. InvokeHTTP – calls an API once a day in order to obtain a list of bus and tram stops
3. SplitJson – splits the list into separate stops
4. JoltTransformJSON – rearranges stop data into `"key": "value"` form
5. EvaluateJsonPath – saves a majority of information into FlowFile attributes: stop id, stop number, stop name, latitude, longitude, and direction
6. InvokeHTTP – calls an API for each bus/tram stop to obtain a list of bus/tram lines that stop there
7. JoltTransformJSON – rearranges data into a list of {`"linia": "XXX"`} objects
8. SplitJson – splits the list into separate FlowFiles for each bus/tram line
9. EvaluateJsonPath – extracts line as a FlowFile attribute
10. RouteOnContent – filters tram lines, understood as one- or two-digit numbers; bus lines are usually three-digit numbers or contain letters like "N" for night lines
11. InvokeHTTP – calls an API for each tram stop and line to retrieve a timetable (a list of departure times)
12. SplitJson – splits the list into separate timetable entries

13. JoltTransformJSON – rearranges stop data into `"key": "value"` form
14. EvaluateJsonPath – extracts time as a FlowFile attribute
15. AttributesToJSON – saves data extracted as FlowFile attributes into FlowFile content in JSON format
16. PutHBaseJSON – puts each record into HBase

This is visualized on a figure 3.

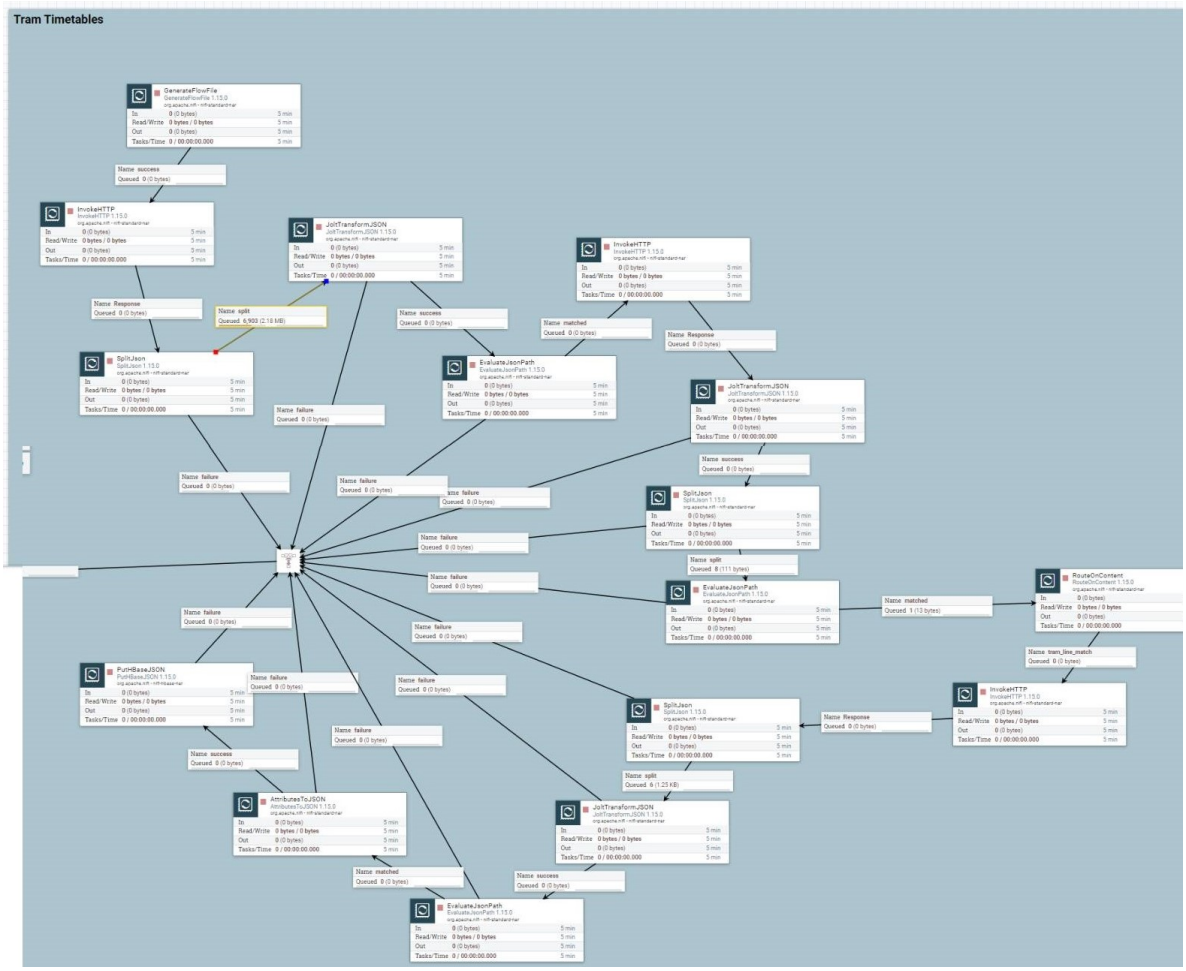

Figure 3: Schema of processing trams and buses positions

3.3 Weather data

Processors used in this pipeline have a following tasks to do:

1. GetHTTP – calls an API every 3 hours in order to download a JSON about the weather with multiple records, each call is sent just one minute after the scheduled update in the API

2. SplitJSON – splits a received JSON into multiple records
3. EvaluateJSONPath – takes out from each record needed fields
4. FlattenJSON – simplifies a structure of received JSON to make it more adequate for HBase
5. PutHBaseJSON – puts a single record into HBase

This is visualized on a figure 4.

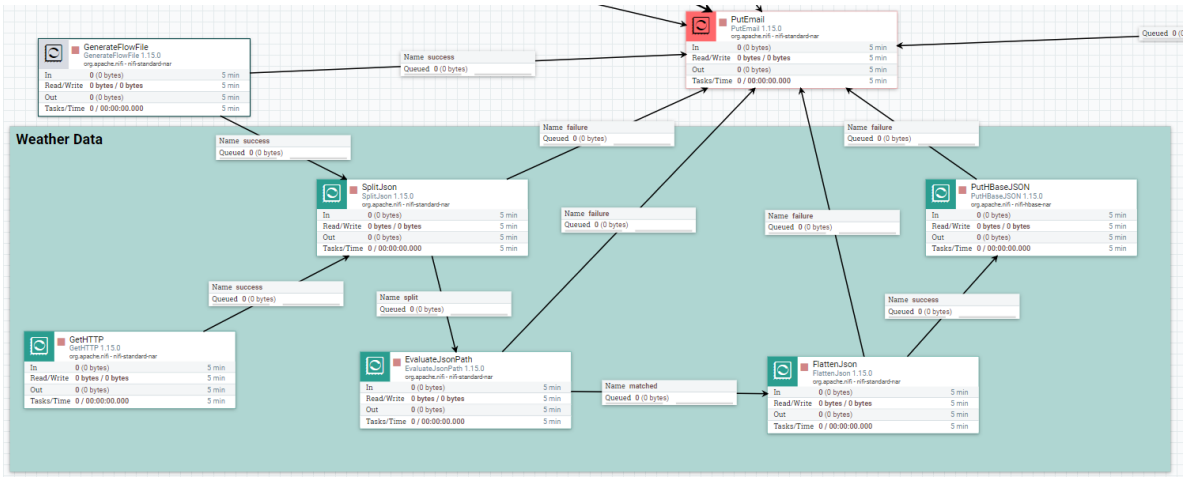

Figure 4: Schema of processing trams and buses positions

3.4 Calculating discrepancies between scheduled and actual arrival

This part is considered by us as a future work. This is a part a data processing that utilizes information from both online information about positions of buses and trams and their schedule.

We plan to calculate the difference between expected arrival and actual arrival for each tram and bus at a given stop based on their position. This requires an identification of a next stop for each vehicle at a given time and calculating the difference in a time when the vehicle eventually approaches the stop.

This data processing part will constitute a third NIFI processing group.

4 Architectural plan

Due to the expected size of the data, on-site solutions were out of question, which mean we'll be using cloud solutions. After a mildly heated discussion we agreed to try and use Google Cloud platform, as it provides the users with the most unpaid resources.

The basis for our architecture will be the "Kappa architecture" approach, since our data already is in the stream form and we wanted to develop something different than we had done previously.

As was described above, we have 3 data sources: online information about the buses and trams, public transport timetables and weather data. The data from each data source is processed and summarized on an online basis on dashboards, prompted to the predictive model and stored in an HBase.

Predictions made by the model will be stored in an HBase table and presented online on a dashboard.

Error at any step will be reported by an email.

The diagram is shown on fig. 5.

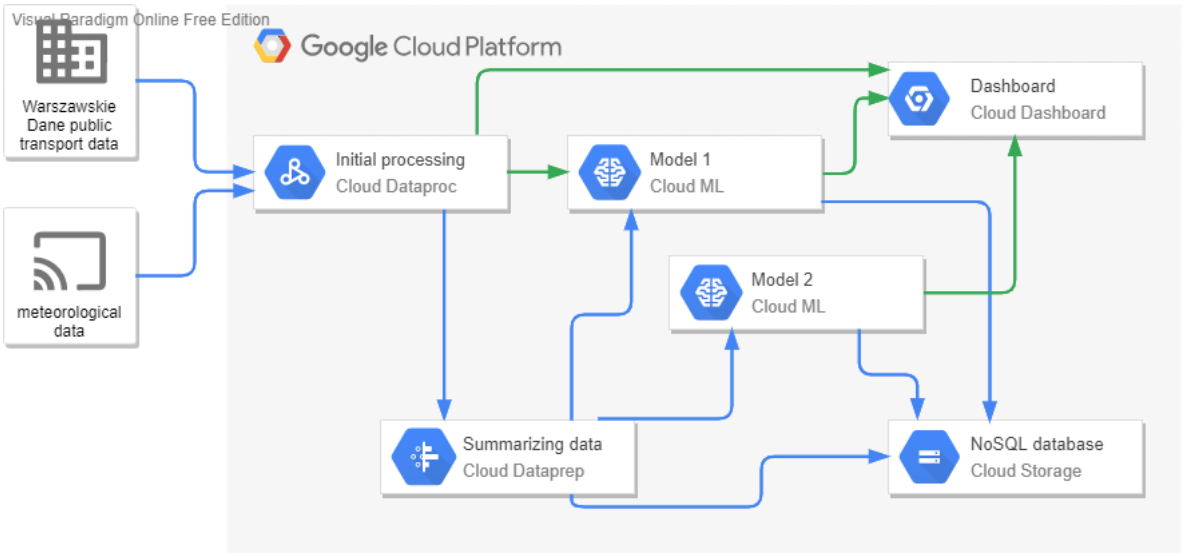

Figure 5: Simplified architectural plan

5 Technology used

In this section we describe technologies that we have already used and that we are planning to use. The whole project is implemented in the **Google Cloud**. All data transformations are handled and managed by the **NIFI** and respective NIFI processors.

We store the data in an **HBase** on a **Hadoop** cluster.

We are planning to use **PySpark** for training machine learning models. However, we will focus on this part in the future.

6 Tests

In order to check if the data are collected, processed and stored in HBase correctly, we compared the expected results with the actual ones. The table below with short descriptions of tests proves that in each case they are equivalent.

Table	Test	Expected result	Actual result
weather	exemplary record	temperature, air pressure etc. in 3 hours intervals with timestamp of collecting data	Figure 6.
trams	exemplary record	longitude and latitude of each tram in given moment of time with timestamp of collecting data	Figure 7.
timetable	number of rows	increasing number of rows	Figure 8.
timetable	exemplary record	position of the tram stop with lines and directions of trams with timestamp of collecting data	Figure 9.

```
hbase(main):001:0> scan 'weather'
ROW                                                    COLUMN+CELL
 1638025200                                            column=weather_data:clouds.all, timestamp=1638019986995, value=75
 1638025200                                            column=weather_data:dt, timestamp=1638019986995, value=1638025200
 1638025200                                            column=weather_data:dt_txt, timestamp=1638019986995, value=2021-11-27 15:00:00
 1638025200                                            column=weather_data:main.feels_like, timestamp=1638019986995, value=271.6
 1638025200                                            column=weather_data:main.grnd_level, timestamp=1638019986995, value=983
 1638025200                                            column=weather_data:main.humidity, timestamp=1638019986995, value=93
 1638025200                                            column=weather_data:main.pressure, timestamp=1638019986995, value=995
 1638025200                                            column=weather_data:main.sea_level, timestamp=1638019986995, value=995
 1638025200                                            column=weather_data:main.temp, timestamp=1638019986995, value=275.03
 1638025200                                            column=weather_data:main.temp_kf, timestamp=1638019986995, value=0.4
 1638025200                                            column=weather_data:main.temp_max, timestamp=1638019986995, value=275.03
 1638025200                                            column=weather_data:main.temp_min, timestamp=1638019986995, value=274.63
 1638025200                                            column=weather_data:pop, timestamp=1638019986995, value=0
 1638025200                                            column=weather_data:sys.pod, timestamp=1638019986995, value=n
 1638025200                                            column=weather_data:visibility, timestamp=1638019986995, value=10000
 1638025200                                            column=weather_data:weather[0].description, timestamp=1638019986995, value=broken clouds
 1638025200                                            column=weather_data:weather[0].icon, timestamp=1638019986995, value=04n
 1638025200                                            column=weather_data:weather[0].id, timestamp=1638019986995, value=803
 1638025200                                            column=weather_data:weather[0].main, timestamp=1638019986995, value=Clouds
 1638025200                                            column=weather_data:wind.deg, timestamp=1638019986995, value=242
 1638025200                                            column=weather_data:wind.gust, timestamp=1638019986995, value=7.43
 1638025200                                            column=weather_data:wind.speed, timestamp=1638019986995, value=3.39
 1638036000                                            column=weather_data:clouds.all, timestamp=1638019987161, value=83
 1638036000                                            column=weather_data:dt, timestamp=1638019987161, value=1638036000
 1638036000                                            column=weather_data:dt_txt, timestamp=1638019987161, value=2021-11-27 18:00:00
 1638036000                                            column=weather_data:main.feels_like, timestamp=1638019987161, value=272.75
 1638036000                                            column=weather_data:main.grnd_level, timestamp=1638019987161, value=984
 1638036000                                            column=weather_data:main.humidity, timestamp=1638019987161, value=88
 1638036000                                            column=weather_data:main.pressure, timestamp=1638019987161, value=996
 1638036000                                            column=weather_data:main.sea_level, timestamp=1638019987161, value=996
 1638036000                                            column=weather_data:main.temp, timestamp=1638019987161, value=275.02
 1638036000                                            column=weather_data:main.temp_kf, timestamp=1638019987161, value=0.02
 1638036000                                            column=weather_data:main.temp_max, timestamp=1638019987161, value=275.02
 1638036000                                            column=weather_data:main.temp_min, timestamp=1638019987161, value=275
 1638036000                                            column=weather_data:pop, timestamp=1638019987161, value=0
 1638036000                                            column=weather_data:sys.pod, timestamp=1638019987161, value=n
 1638036000                                            column=weather_data:visibility, timestamp=1638019987161, value=10000
 1638036000                                            column=weather_data:weather[0].description, timestamp=1638019987161, value=broken clouds
 1638036000                                            column=weather_data:weather[0].icon, timestamp=1638019987161, value=04n
 1638036000                                            column=weather_data:weather[0].id, timestamp=1638019987161, value=803
 1638036000                                            column=weather_data:weather[0].main, timestamp=1638019987161, value=Clouds
 1638036000                                            column=weather_data:wind.deg, timestamp=1638019987161, value=221
 1638036000                                            column=weather_data:wind.gust, timestamp=1638019987161, value=4.77
 1638036000                                            column=weather_data:wind.speed, timestamp=1638019987161, value=2.11
 1638046800                                            column=weather_data:clouds.all, timestamp=1638019987161, value=90
 1638046800                                            column=weather_data:dt, timestamp=1638019987161, value=1638046800
 1638046800                                            column=weather_data:dt_txt, timestamp=1638019987161, value=2021-11-27 21:00:00
 1638046800                                            column=weather_data:main.feels_like, timestamp=1638019987161, value=272.09
 1638046800                                            column=weather_data:main.grnd_level, timestamp=1638019987161, value=985
 1638046800                                            column=weather_data:main.humidity, timestamp=1638019987161, value=87
 1638046800                                            column=weather_data:main.pressure, timestamp=1638019987161, value=998
```

Figure 6: Exemplary rows of 'weather' table.

7 Analysis

7.1 Modelling

We model public transportation delays for the next few hours or the next few days. Our data processing gives information about delays in public transport and the current

```
hbase(main):004:0> scan 'trams'
ROW                                    COLUMN+CELL
 1219+1218_2021-11-27 15:19:56         column=position:Brigade, timestamp=1638022805749, value=3
 1219+1218_2021-11-27 15:19:56         column=position:Lat, timestamp=1638022805749, value=52.255444
 1219+1218_2021-11-27 15:19:56         column=position:Lines, timestamp=1638022805749, value=28
 1219+1218_2021-11-27 15:19:56         column=position:Lon, timestamp=1638022805749, value=20.98218
 1219+1218_2021-11-27 15:19:56         column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:56
 1219+1218_2021-11-27 15:19:56         column=position:VehicleNumber, timestamp=1638022805749, value=1219+1218
 1222_2021-11-27 15:19:45              column=position:Brigade, timestamp=1638022805749, value=4
 1222_2021-11-27 15:19:45              column=position:Lat, timestamp=1638022805749, value=52.260612
 1222_2021-11-27 15:19:45              column=position:Lines, timestamp=1638022805749, value=27
 1222_2021-11-27 15:19:45              column=position:Lon, timestamp=1638022805749, value=20.980764
 1222_2021-11-27 15:19:45              column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:45
 1222_2021-11-27 15:19:45              column=position:VehicleNumber, timestamp=1638022805749, value=1222
 1260_2021-11-27 15:19:53              column=position:Brigade, timestamp=1638022805749, value=5
 1260_2021-11-27 15:19:53              column=position:Lat, timestamp=1638022805749, value=52.264183
 1260_2021-11-27 15:19:53              column=position:Lines, timestamp=1638022805749, value=28
 1260_2021-11-27 15:19:53              column=position:Lon, timestamp=1638022805749, value=20.972136
 1260_2021-11-27 15:19:53              column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:53
 1260_2021-11-27 15:19:53              column=position:VehicleNumber, timestamp=1638022805749, value=1260
 1268_2021-11-27 15:19:57              column=position:Brigade, timestamp=1638022805749, value=1
 1268_2021-11-27 15:19:57              column=position:Lat, timestamp=1638022805749, value=52.25271
 1268_2021-11-27 15:19:57              column=position:Lines, timestamp=1638022805749, value=28
 1268_2021-11-27 15:19:57              column=position:Lon, timestamp=1638022805749, value=21.04988
 1268_2021-11-27 15:19:57              column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:57
 1268_2021-11-27 15:19:57              column=position:VehicleNumber, timestamp=1638022805749, value=1268
 1289+1290_2021-11-27 15:19:57         column=position:Brigade, timestamp=1638022805749, value=2
 1289+1290_2021-11-27 15:19:57         column=position:Lat, timestamp=1638022805749, value=52.22466
 1289+1290_2021-11-27 15:19:57         column=position:Lines, timestamp=1638022805749, value=27
 1289+1290_2021-11-27 15:19:57         column=position:Lon, timestamp=1638022805749, value=20.931274
 1289+1290_2021-11-27 15:19:57         column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:57
 1289+1290_2021-11-27 15:19:57         column=position:VehicleNumber, timestamp=1638022805749, value=1289+1290
 2025+2024_2021-11-27 15:19:53         column=position:Brigade, timestamp=1638022805749, value=2
 2025+2024_2021-11-27 15:19:53         column=position:Lat, timestamp=1638022805749, value=52.237843
 2025+2024_2021-11-27 15:19:53         column=position:Lines, timestamp=1638022805749, value=13
 2025+2024_2021-11-27 15:19:53         column=position:Lon, timestamp=1638022805749, value=20.982445
 2025+2024_2021-11-27 15:19:53         column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:53
 2025+2024_2021-11-27 15:19:53         column=position:VehicleNumber, timestamp=1638022805749, value=2025+2024
 2028_2021-11-27 15:19:52              column=position:Brigade, timestamp=1638022805749, value=3
 2028_2021-11-27 15:19:52              column=position:Lat, timestamp=1638022805749, value=52.184258
 2028_2021-11-27 15:19:52              column=position:Lines, timestamp=1638022805749, value=14
 2028_2021-11-27 15:19:52              column=position:Lon, timestamp=1638022805749, value=21.023954
 2028_2021-11-27 15:19:52              column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:52
 2028_2021-11-27 15:19:52              column=position:VehicleNumber, timestamp=1638022805749, value=2028
 2029_2021-11-27 15:19:56              column=position:Brigade, timestamp=1638022805749, value=6
 2029_2021-11-27 15:19:56              column=position:Lat, timestamp=1638022805749, value=52.20987
 2029_2021-11-27 15:19:56              column=position:Lines, timestamp=1638022805749, value=14
 2029_2021-11-27 15:19:56              column=position:Lon, timestamp=1638022805749, value=20.982029
 2029_2021-11-27 15:19:56              column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:56
 2029_2021-11-27 15:19:56              column=position:VehicleNumber, timestamp=1638022805749, value=2029
 2035+2034_2021-11-27 15:19:53         column=position:Brigade, timestamp=1638022805749, value=7
 2035+2034_2021-11-27 15:19:53         column=position:Lat, timestamp=1638022805749, value=52.22477
 2035+2034_2021-11-27 15:19:53         column=position:Lines, timestamp=1638022805749, value=13
 2035+2034_2021-11-27 15:19:53         column=position:Lon, timestamp=1638022805749, value=20.930975
 2035+2034_2021-11-27 15:19:53         column=position:Time, timestamp=1638022805749, value=2021-11-27 15:19:53
 2035+2034_2021-11-27 15:19:53         column=position:VehicleNumber, timestamp=1638022805749, value=2035+2034
```

Figure 7: Exemplary rows of 'trams' table.

weather conditions as well as the prognosis. We have both historical data (for a relatively short time period, because this project is still very fresh) and a current data that comes from a stream.

Thus we decided to fit a model on a batch historical data that we have already gathered to have an initial model and then model will be updated constantly with an incoming streaming data.

This part of the project is still a future work, but at this moment we have already made some plans regarding some details of a modelling. Single record / row that will be fed into a model consists of a following fields:

1. time of a day - delays may change depending on a time, for example the greatest delays may occur during morning hours and afternoon hours
2. line number - different lines may have different vulnerability for delays
3. current vehicle position - some parts of the city may have more delays than others
4. last discrepancy between expected and actual arrival - delays may accumulate over time

5. next bus stop id - the reasoning here is similar to the vehicle position
6. direction - depending on a direction, delays may vary
7. temperature - temperature (especially low, may affect driving conditions)
8. feels like temperature
9. minimum temperature
10. maximum temperature
11. pressure - pressure may affect a human capabilities of driving
12. humidity - humidity may affect a human capabilities of driving
13. cloudiness - cloudiness affects visibility
14. wind speed - perhaps wind speed affects a way vehicles are moving
15. wind direction - perhaps wind direction affects a way vehicles are moving
16. visibility - drivers drive slower when there is low visibility
17. probability of precipitation - precipitation influences delays
18. rain - driving style changes when raining
19. snow - driving style changes when snowing

7.2 Exploratory data analysis

For the users that want some more and clear insight into the data there will be dashboards containing graphs and tables summarizing the data. This part is still considered by us as a future work, but we have already made plans regarding it. We plan to include following graphs in the dashboard (please, note that this may change in the future):

1. average delays against the time of the day
2. average delays against the week day
3. average delays against the month
4. average delays against a particular line
5. heatmap of Warsaw with average delays
6. current (online) average delay
7. current (online) delays for each line
8. current (online) heatmap of Warsaw with information about delays

8 Task allocation

Note: task allocation may and will be subject to changes as the project progresses. This should be only taken as a rough guidelines to who will be held responsible for the successful implementation of certain parts of the final solution:

- **Karol**: dashboards
- **Laura**: data gathering
- **Mateusz**: data storage
- **Wojciech**: predictive model

```
Took 1.3203 seconds
hbase(main):005:0> count 'timetable'
Current count: 1000, row: 024ecf08-d7e3-46bd-80bb-9a211655cab2
Current count: 2000, row: 04b106d8-d910-4182-9bde-d6f0a3caf994
Current count: 3000, row: 06e03cb5-93d7-4173-af88-33d2edd8d51a
Current count: 4000, row: 092ad2c4-f13c-4dc9-8813-731d78c50e9a
Current count: 5000, row: 0b59e9f8-8a91-466e-a516-05f9483fbfff
Current count: 6000, row: 0db1d3c0-42b8-4b5a-a7a2-94fdfaa13662
Current count: 7000, row: 0fefd32a-a03a-41b3-933e-4a27bb1100e9
Current count: 8000, row: 122be63c-530a-4926-9896-9d713bd3ca82
Current count: 9000, row: 145f9401-bdd9-4fd4-a6a9-8d4d5c17be36
Current count: 10000, row: 16be86b6-c667-45f7-9554-d582757fa1ce
Current count: 11000, row: 18fe2b91-d13f-4774-a647-8445dd4c69f4
Current count: 12000, row: 1b5a0944-7120-4c90-a9a9-a768edc2c863
Current count: 13000, row: 1d9fe1b1-c360-4f00-98f2-4458044d3c01
Current count: 14000, row: 201756ec-9bb3-4231-adc2-c54b9cad7f67
Current count: 15000, row: 226850b6-e9d0-4b0c-af12-04040b468604
Current count: 16000, row: 24c3469b-33f2-4f61-a656-8bbb63410e7d
Current count: 17000, row: 27177c4b-7b18-4ea6-9708-f0f7ce7d24c5
Current count: 18000, row: 2957992e-91cd-42a7-89ff-7807da3e24ec
Current count: 19000, row: 2bb0768a-d252-4e7a-ac3e-41fc92be5f82
Current count: 20000, row: 2e07e68e-f2b9-4332-9854-fa676c3fe5d8
Current count: 21000, row: 306c87d9-ef65-48bd-bc05-7e45f1afc4f9
Current count: 22000, row: 32caadc7-f3e3-4a3e-991f-493da336830c
Current count: 23000, row: 350f7c02-8ea7-406c-a789-d4bec0afbdc1
Current count: 24000, row: 375756bc-b836-4170-bf74-76459d949f07
Current count: 25000, row: 39aae560-be8d-431b-b637-9801eb1f6100
Current count: 26000, row: 3bf9e3bd-cec5-4cd1-8f8d-b49960ba0d15
Current count: 27000, row: 3e512d81-a66b-4129-84d8-c14727c148f6
Current count: 28000, row: 40b7e9cd-9496-4568-a2ea-456b1de2375a
Current count: 29000, row: 431e134f-3da3-4f51-83a7-a748d96b9699
Current count: 30000, row: 456031cd-1b5e-404c-9062-ffb466a557fc
Current count: 31000, row: 47ae9716-ac00-40dc-888e-8a26771d0a27
Current count: 32000, row: 4a01d10e-b6c7-4a05-ab77-fefe901664a3
Current count: 33000, row: 4c3edecb-0e9d-4830-8ea0-cd538f79f965
Current count: 34000, row: 4e75b79d-4e8b-4a6b-bf90-6432f3991c84
Current count: 35000, row: 50b28f24-2109-4a2f-b221-ec45986f6542
Current count: 36000, row: 52fe225a-2ccc-4d8e-afb8-5f8be40d701f
Current count: 37000, row: 55530411-522d-4316-8478-e42abc7cc88e
Current count: 38000, row: 57cb3b0b-8464-41cf-ab77-7baf612963e8
Current count: 39000, row: 5a27a0de-7e8b-4367-becc-b03abb638c0f
Current count: 40000, row: 5c849a8d-d133-46fc-a16b-37c539bc71ea
Current count: 41000, row: 5ede42ba-860d-46b3-b135-3ca5e09ebb8f
Current count: 42000, row: 611ffdb4-2de7-4a0e-ac9f-dd2b3e95dc82
Current count: 43000, row: 637f690f-4023-472e-ac4e-b2968a738518
Current count: 44000, row: 65e0a012-12ce-42e9-8dd7-8fd74e966700
Current count: 45000, row: 68264e47-645a-4848-9362-3f69234d736b
Current count: 46000, row: 6a86ab9f-6011-4065-8f0d-348f6b5edb0d
Current count: 47000, row: 6ce83dd4-827d-47df-9d19-f83b88367b36
Current count: 48000, row: 6f52c885-64fe-43e5-b973-925e6498c94e
Current count: 49000, row: 71a1a514-6e19-4309-90a4-ad732890f742
Current count: 50000, row: 73ded3cd-e887-4c0f-9e2a-e01bdbbb8e94
Current count: 51000, row: 762435c7-7566-4062-9435-a13eb9cacd84
Current count: 52000, row: 7869f7e7-1dbe-4747-86dd-fc39574e6f51
Current count: 53000, row: 7aa9d127-2518-4a0c-b891-d91c9346a6f0
```

Figure 8: Number of rows in 'timetable'.

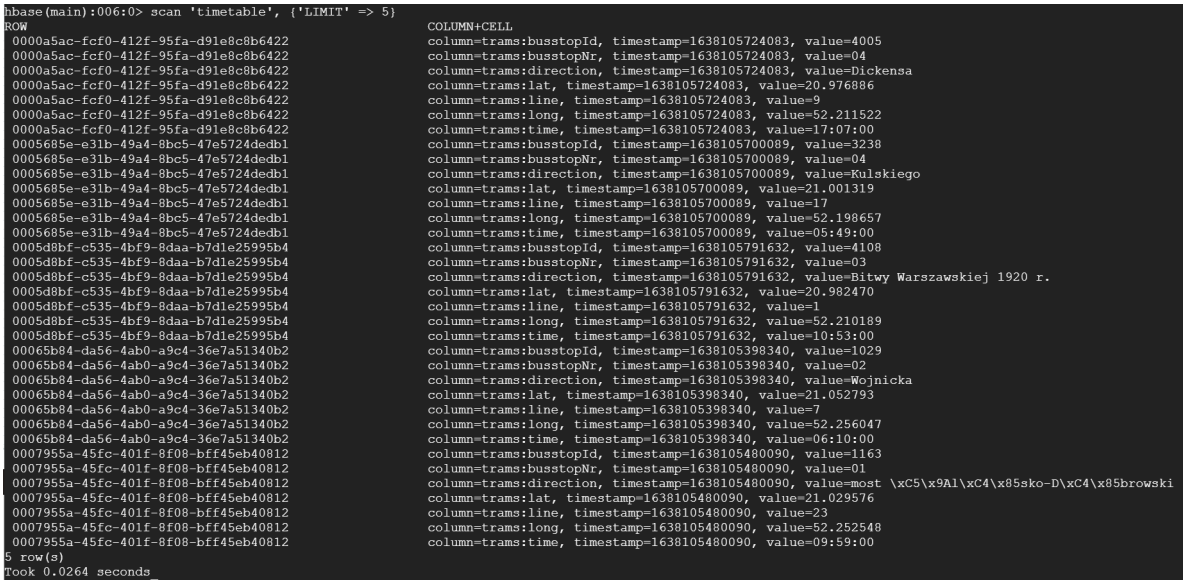

Figure 9: Exemplary rows of 'timetable'.