DATABASE DESIGN

1. **Introduction**
2. **General scheme**
3. **Data acquisition to database**
4. **Database to Interpolation**
5. **Interpolation to database**
6. **Database to MapDraw**

1.Introduction

The scope of this document is to illustrate what our database proposal is about. It will not be explained in detail all submodules between the steps, for this reason, another documents will be done.

We just show our idea of how the **dataflow** could be from the acquisition, to the final implementation.

The software used is WAMP, which gives us a local server and a graphic interface that works with mySQL.

The ID, or primary key, of each table is not important for us, thus, we just have an incremental ID every time we add another row. What we really need is to be able to split the tables due to time, position (UTM) or pollutant order, and that can be done independently of primary key.

We'll see in more detail henceforth.

2. General Scheme

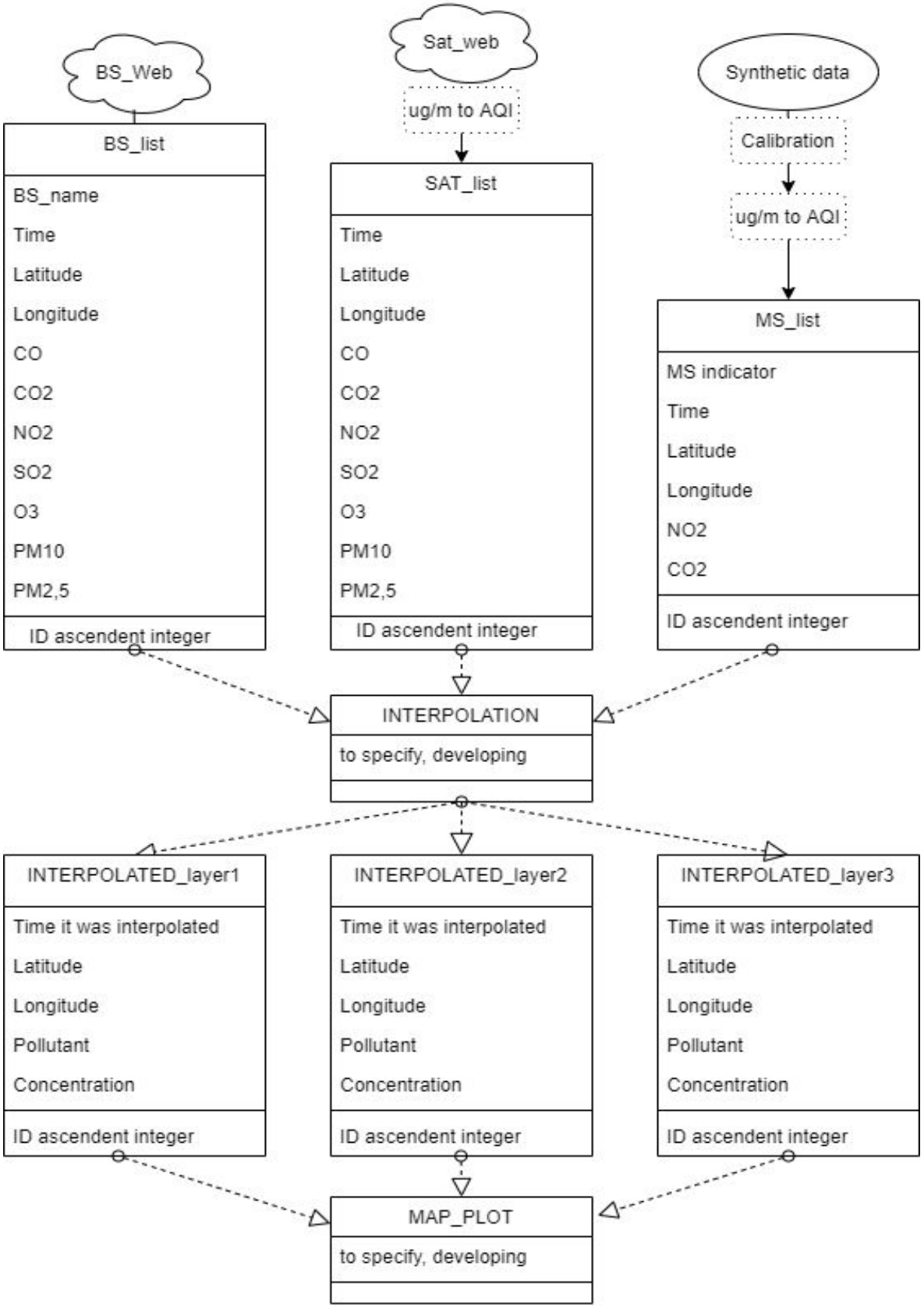

Image1: General scheme

3. Data acquisition to database

First of all, all values on database will be stored as AQI values instead of concentration. This is due to we want our map to be drawn as AQI's colored table, which identifies how painful are the different pollutants on a common scale.

Of course, even if not specified at present in the general scheme, we could add a submodule, depending on if we have stakeholders interested in concentration values, that do the inverse transformation and gives the concentration values(ug/m).

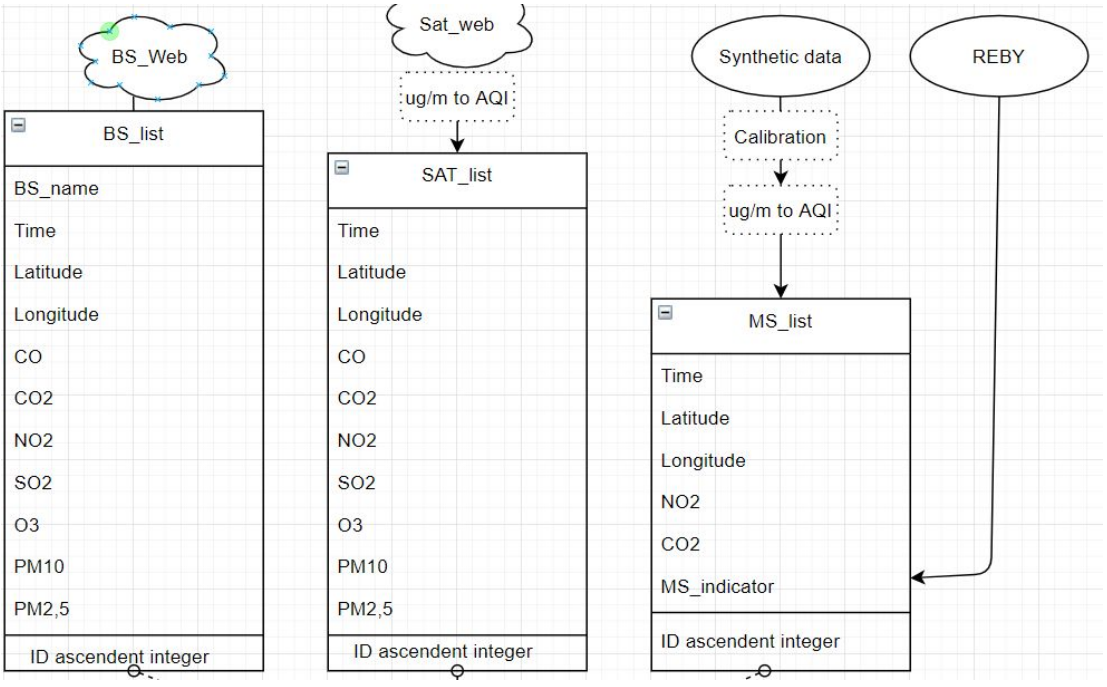

image2: Data acquisition to database scheme.

The first thing we want to clar out is that "REBY" module.

We know that mobile sensors can be tracked using an ID for each one, so every time a row is added, we know from which scooter comes from so, for example, if one of them is broken we can skip it and ignore their values. The main problem is due to at the moment, with our Synthetic data algorithm its difficult

to add a “track” parameter, so MS_indicator won’t be able at this first implementation.

Returning to the scheme, we divide the database in **3 tables**, one **for each source** because we want to have very distinguished in order to both, organisation and interpolation purposes (it will be explained in section 4). As we can see, both MS and BS list have an ID or name indicator, the idea is simple, have the **sources tracked,** so we can delete or give priority to the ones we want depending on their activity, fiability or others we could specify in the future. Also, the time registered corresponding to each measure is stored. By doing that, later we can **choose the range** of time **to work with**. Analogous, this is why longitude, latitude and pollution type is stored.

To end this section, for MS values, we want to **save the calibrated value** (will be explained in another document), to reduce the complexity of interpolation algorithm, because we want it to be as fast as possible.

4. Database to interpolation

In this section is explained the criteria of the interpolation algorithm to choose the data.

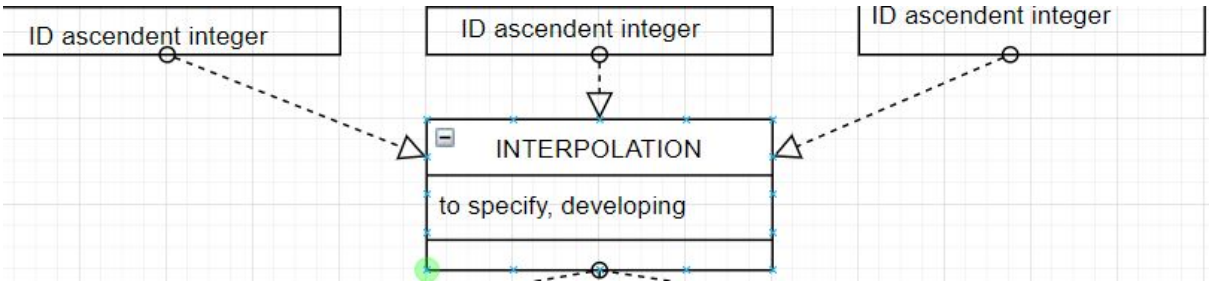

Image 3: Notice that “INTERPOLATION class” it is outside of the database.

MySQL gives us the tool of multi filtering, that is, you can choose ranges of time, position, pollution, practically any column defined in the tables. Periodically, interpolation will choose between source type (table), range of time, range of space (UTM columns), and pollutants, depending on which layer want to do the interpolation.

Not many commentaries on this little step, just that as said before, all details of module algorithms will be discussed in concrete documents.

5. Interpolation to database

Similar to first section, data is stored depending on layers, within which the time, stored as a periodical value (not according to the measure instant as data acquisition), position (for each layer will be equidistributed), pollutants and, a new column "general pollution". The idea behind is to give a general vision, choosing the most painful pollutant in each area according to AQI tables.

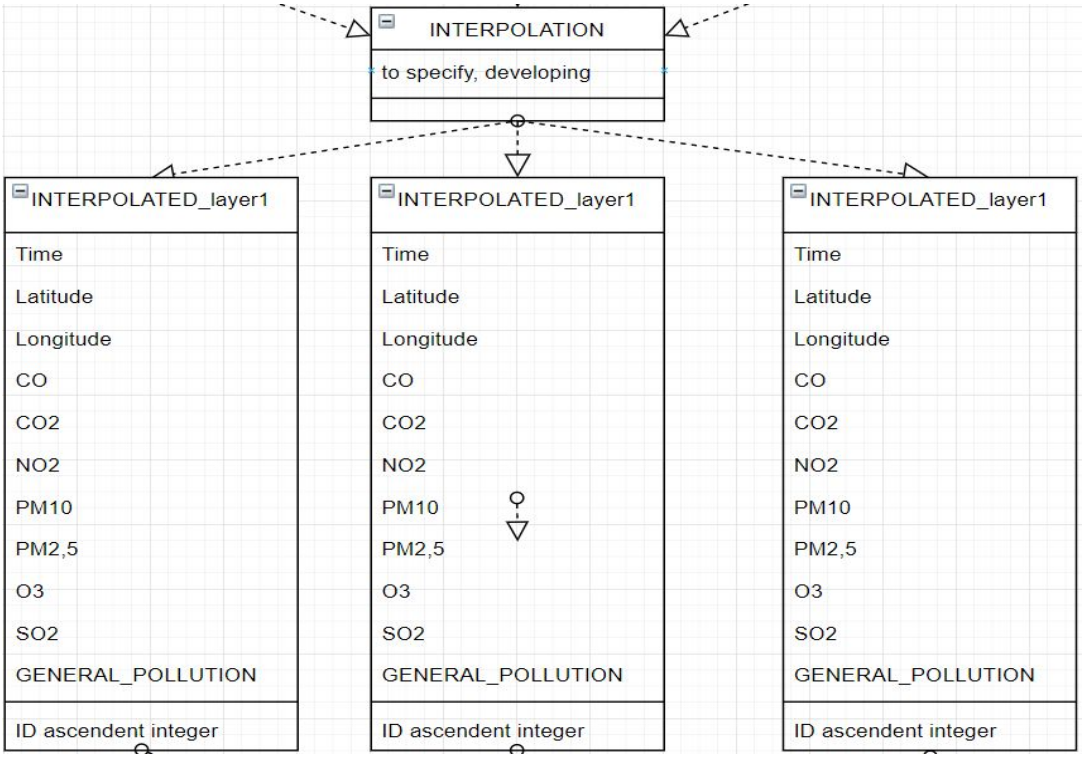

Image4: The tables will be separated from the source ones. Names are not definitive.

6. Database to map draw

Last but not least, the idea is that map draw will periodically update, at the same "frequency" as interpolation program but with a delay, corresponding to how long it takes to store interpolated values. Then the algorithm will print the values according to the layer (wait until all documentation is done).

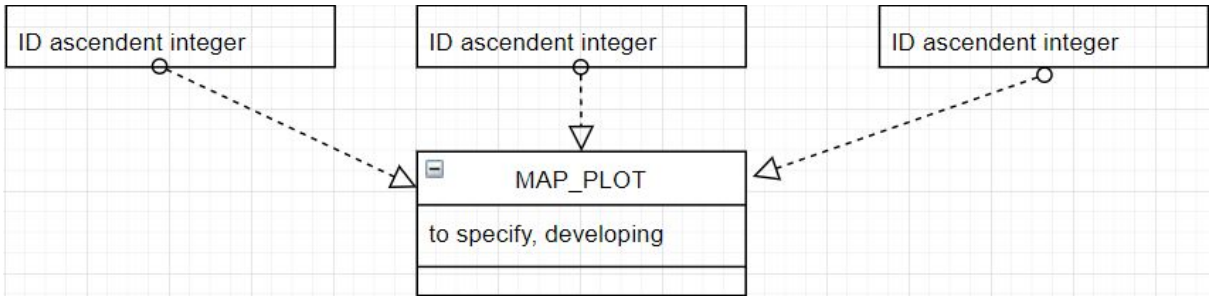

Image5: Names to be specified, ID ascendent integer is the table ID, but as explained, it is not used, at least for now.