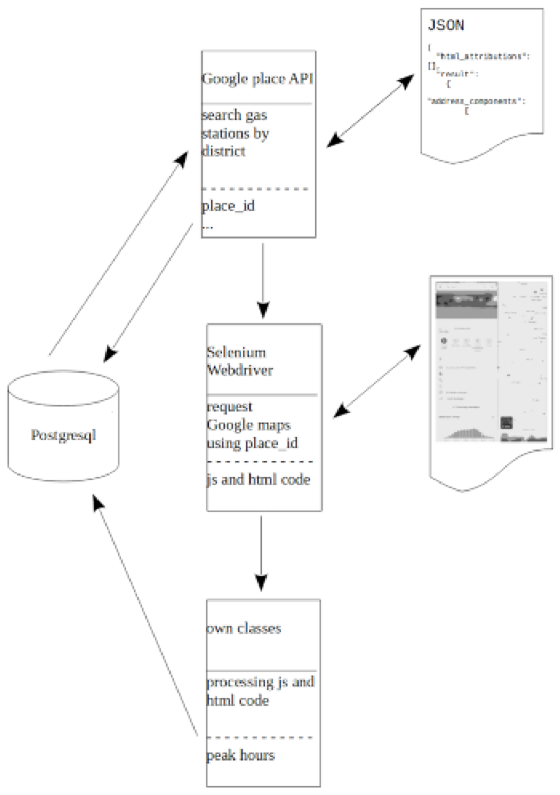
1 Part 2 . . . searching gas stations and fetching data

about them and about the peak hours.

1.1 Approach

As a first step, I use publicly available data about the capacity use of the gas stations on Google maps (peak times). To do this, I fetching all gas station place IDs (and further values) of all districts using the google place API. With this IDs, I visit the Google Maps website and read out the data for the peak hours for each gas station. I fill a Postgresql database with all the data obtained. The evaluations can then be done later via the database.



1.2 Database class

As in part one, the connection to the database is established when a database instance is called. The necessary credentials are in the file `credentials.py`. The database class is there to

1. return district names (which have not yet been entered)
2. record data from the Place API call (place id, address, name, type, geolocation data etc)
3. map address data and types to the corresponding n:m tables
4. record previously unrecorded types into the appropriate table and process their IDs for subsequent records
5. record peak times into the database.

I will not use docstrings here, as the tasks of the individual functions are easy to see.

1.3 Fetching Maps class

This class contains the functions to grab the maps web pages using Selenium webdriver.

1.4 Place class

This class contains the functions to fetch the place details. For this purpose, the google place api is used. A brief description of each function can be found in the individual docstring.

1.5 Data processing class

The functions of this class are there to extract the data from Maps web pages. On the one hand, the web page source code is parsed in such a way that the decisive places with the data are exposed. Afterwards, the data is prepared in such a way that it can be loaded into the corresponding table. In addition to this primary job, other smaller data cleaning tasks are also executed.

1.6 Main class

The main class is responsible for maintaining the order of the individual tasks. As shown in the graphic above, a search query is first executed with the given search term. The answers returned by the API (search results) are successively transferred to the database and the respective peak time data is obtained. For each individual location, the Maps page is accessed and a check is made to see whether the peak time data is available or not. If they are available, they are processed accordingly. If not, a no-data record is created for this location. As soon as all search results have been processed, the next page of the API is called up or a new search query is started. Google place API delivers a maximum of 20 search results per call. These can be extended by up to two additional "pages". For this purpose, a so-called next-page-token is supplied. However, each page cost a request credit. All places that have already been included in the database are not recorded any further. I have tried to avoid as many errors as possible; at the same time I have tried to keep the number of exceptions low in order not to store too many no-data responses.

connect database

1.7 Feed the search

1.7.1 Identify already recorded districts

1.7.2 Get not recorded district names

1.8 Start fetching data and populating DB

1.8.1 Creating search term and start searching by district

```
searching: Tankstelle in Adlershof
connected

========================= 1 ===========================
HEM Tankstelle Adlergestell 305, 12489 Berlin, Germany
```

```
------------------------------------------------------------


========================== 2 ===========================
JET Tankstelle Glienicker Weg 105, 12489 Berlin, Germany
------------------------------------------------------------


========================== 3 ===========================
Elan-Tankstelle Adlergestell 179, 12489 Berlin, Germany
------------------------------------------------------------


... etc.
```

close db connection

1.9 Conclusion

Most of the data were automatically collected by the predefined algorithm, i.e. on the basis of the district names that had already been stored in the database. In addition to the search queries by district name, manually created lists (motorway numbers and cities with more than 90,000 inhabitants) were added to the search query and processed. I did this to ensure that the desired locations on motorways and near metropolitan areas were included in every case.

Besides some error exceptions due to missing or incorrect values in the place api and some non-fetchable maps pages, the data collection went pretty well. Unfortunately, the quota for the API calls was not large enough to achieve even tighter coverage. This is also the reason for the additional search lists mentioned earlier.

1.9.1 Some personal reflexions

- ever stop the chromedriver, otherwise it leads to a buffer overflow
- next time use a pipeline model