Kirill Kuvshinov

Mobile solution for gas price collection

Based on data from a Finnish gas price website

Helsinki Metropolia University of Applied Sciences

Bachelor of science

Media engineering

Thesis

17 April 2017

Helsinki
Metropolia
University of Applied Sciences

Author Title Number of Pages Date	Kirill Kuvshinov Mobile solution for gas price collection 26 pages + 16 appendices 17 April 2017
Degree	Bachelor of science
Degree Programme	Media engineering
Specialisation option	Android development, Java, PHP
Instructor(s)	Aarne Klemetti, Researching Lecturer

This project claims to address the common problem of providing drivers with actual gas prices. No app studied seems able to retrieve today's gas data only and show them in a user-friendly fashion. Some include earlier data, others rely on community feeds; still others reveal prices only along chosen navigation routes. This paper offers no universal methods to address all the challenges. It is rather an educational attempt to solve some of them from a point of view of a student and a driver.

As the subject of this project, my own approach is described at length. Generally, it includes two Android apps facilitating each other, and both contributing to own database. Thus, only current fuel prices are shown if any, the popular locations' details are uploaded to own database, and historical data kept for future reference. Navigating the driver to the nearest gas station of choice used to be part of the project, yet recent improvements by Google Maps may have invalidated the old certificate.

In addition, the mobile solution features own price updates management (auto or manual mode) and exploits www.polttoaine.net as a solid price-quoting platform. As nothing resulted from my own attempts to contact the platform admins on trainee cooperation, scraping and parsing methods are used in this project (for educational purposes only). All trademarks discussed in this paper are property of their respective owners.

Keywords	Gas price, fuel cost, price collection, Android, Java, PHP, database, JSON, location manager, geolocation, GPS, driver app, cron, Google Maps, API, certificate

Contents

1 Introduction

The collection of current fuel prices for a specific area has always been on the radar of modern developers, no matter what programming language they use. These numerous apps claim to provide up-to-date information[1] coupled with navigation opportunities and/or distance or fuel endurance calculations along chosen routes, etc. In the multitude of choices, one runs the risk of getting lost because of the opportunities they offer.

Yet some of the apps studied and compared for this project seemed quite sophisticated to meet the simple need of a driver lost somewhere, e.g. in Finland, and in need of a reasonably priced refuel, and at a gas station as near as possible to his or her current location. The driver either has to use different apps subsequently to figure out the location and the way or rely on community-fed applications, e.g. Waze (described in 2.4), which may have no recent updates for the occasion.

The following three major reasons accounted for choosing such a topic for the final year project, namely (i) studying at Helsinki Metropolia UAS, (ii) owning a car in Finland and (iii) having a wish to engage in building an app that would help tackle this simple emergence above in a driver-friendly manner. Yielding any results from own research as such seemed a justifiable return for the time and effort invested.

Thus, the project goal was to create a price-quoting app that would highlight up-to-date prices only for the given position (identified with geolocation). This, later in the project, saw reinforcement in the form of a second app to control data uploading to own database. Therefore, the project scope grew up to two apps with different functions, thus turning into a combined mobile solution. Hence the title of the project.

2 Theoretical and practical backgrounds

What follows is an extensive account of the pre-study that took place in 2016 and gave shape to this project further into 2017.

[1] The terms "information" and "data" are used here interchangeably despite their fine differences, such as that the former is an organised form of the latter [1,188-189].

2.1 Brief introduction to some modern price-quoting apps and platforms

For this project, counterpart quoting tools were studied for similar functionality and user interface features within January-April 2016. Understandably, by now, there may have appeared quite a number of new apps, more driver-friendly and versatile; a quick observation of Google Play website [2] confirms this assumption. Therefore, this project claims no serious competitiveness to these tools and only demonstrates own methods found valuable and acquired knowledge to make a coded product by personal contribution.

Nonetheless, given this project limitation, the study included a series of short “road testing” of the following mobile apps and platforms: GasVisor[2], BenzinPrice[3], Waze[4], Google Maps / Street View, and Polttoaine[5] (as described in 2.4).

2.2 Significance of the topic

By the time this research started, Google had updated its Maps app so that it quoted fuel prices at gas stations found along navigation routes; although that option was available for US and Canada only [3;4;5]. That functionality might have seen a larger geographical coverage than before by the time this thesis is available for review.

Still, given that Google must be daily improving its map-oriented tools in terms of fuel prices, and given that more apps tend to appear on the market, the problem of quoting the real prices “here and now” remains among the highlights of today.

2.3 Own attempts to organise price collection locally

The project originally consisted in personal collection of local fuel prices based on the mentioned apps. The purpose of that practical pre-study was to make sure which app featured better updatability in the context of the emergency case described in the Introduction. That approach became the focus of this project after earlier price collection trials had failed.

---

[2] https://play.google.com/store/apps/details?id=com.ima.gasvisor
[3] www.benzin-price.ru/price.php?region_id=78
[4] https://play.google.com/store/apps/details?id=com.waze
[5] www.polttoaine.net

For brevity, those trials were semi-theoretic and saw quite different tools. E.g., hidden positioning of hunter cameras in front of a gas station price stand or even thinking of quadcopters with cameras to take pictures of the same. Obviously, financial considerations intervened among other challenges, which included having to operate the flying machines and structure their flights and control crews; or having to tell the prices on the digital camera shots, with OCR (or optical character recognition[6]) offering to switch the thesis topic. The OCR initiative never proved to be precise enough when used on colour pictures taken at dawn or dusk, esp. with prices in light-red digits vs black background (as shown in Appendix 1). All those trials combined invited to quite a different field of study and scope, irrelevant to this paper and thus were excluded.

2.4 Country-specific use of price-quoting apps

With these failures and a real life necessity to live outside of Finland within the period of dealing with this project, Saint Petersburg became the testing ground for running the apps and platform mentioned in 2.1. In addition, almost all Russian gas providers would refrain from updating their prices regularly on their websites, if any, or prefer to display average prices only (as evidenced in Appendix 2.1-2.3). On the other hand, it helped to focus more on the apps at hand.

The main idea was to see how frequently all of them received price updates in order to understand how to proceed with the project. The testing spanned sporadically within January-April 2016, between mornings to night-time.

Challenges vs favourable conditions

Below follow short analyses of pros and cons of each candidate app.

Google Maps / Street View

Initially thought as an ideal data provider for the OCR technique mentioned above, it turned out on the contrary. As per their declaration online (Appendix 3.1), the Google

[6] Tesseract for PHP (https://github.com/tesseract-ocr) was tested in parallel just to find out that it required more time and effort to introduce efficiently in this project.

Maps visual information may remain unchanged for 1-3 years. What also suggested discontinuation with this tool was that the option to see fuel costs along a chosen route in the navigation mode was still unavailable in Russia. For these two reasons, this app left the test row right from the start.

GasVisor / BenzinPrice

Since the first app here depends on the second platform's database (as mentioned at http://www.benzin-price.ru/android_gasvisor.php under *Возможности* or “App’s capabilities”), it made sense to combine them for analysis. Figure 1 shows three screenshots when used in St Petersburg.

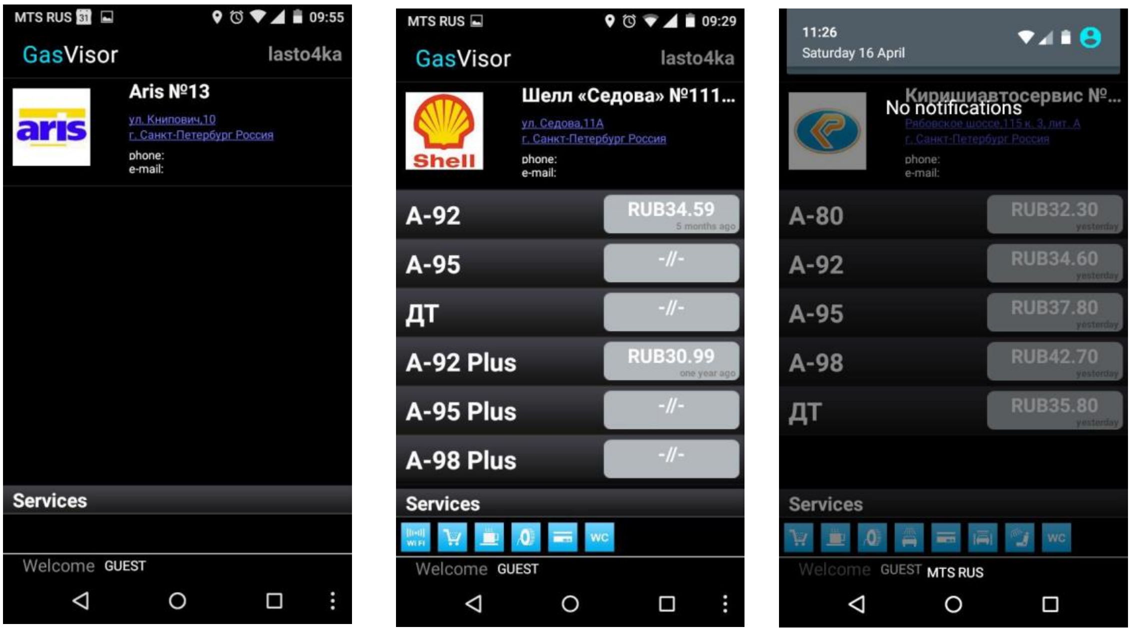
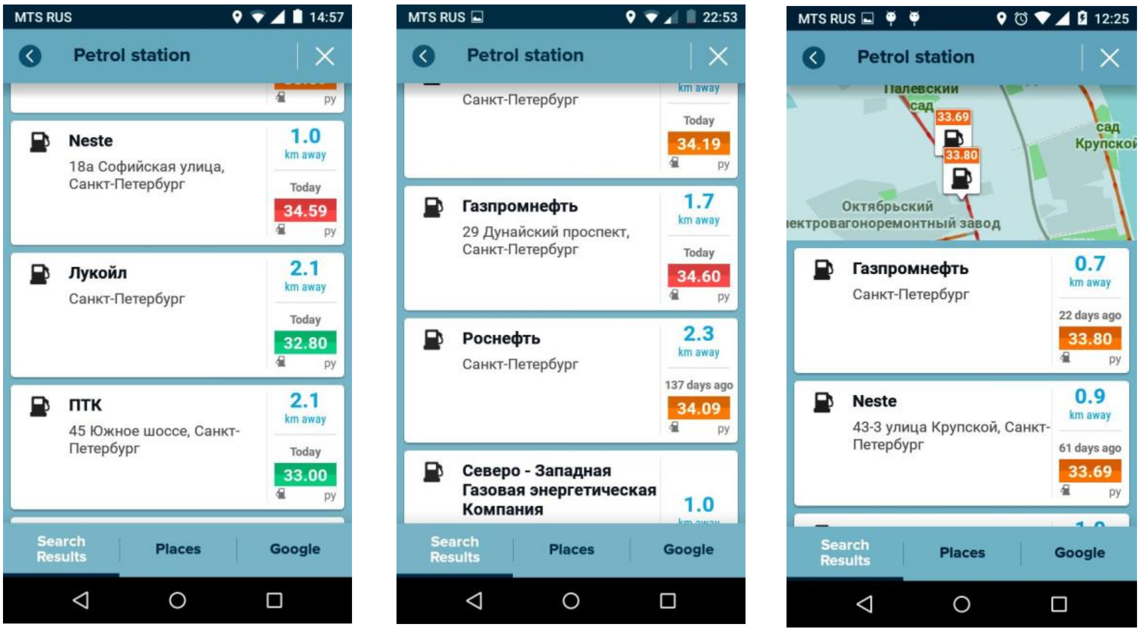


Figure 1. GasVisor at work

As a result, despite good station-related information, these driver crowdsourcing tools were usually late with updates and kept posting earlier prices instead (Appendix 3.2). In addition, *GasVisor* might at times display an empty screen or too outdated fuel prices (one year ago) as seen in Figure 1. These were the reasons to exclude the two from the research.

Waze

When compared to its counterparts, this price-quoting app, although completely dependent on the local driver community too, never exposed empty activities, and always showed better updatability (as Figure 2 demonstrates, note the coloured tags on the right



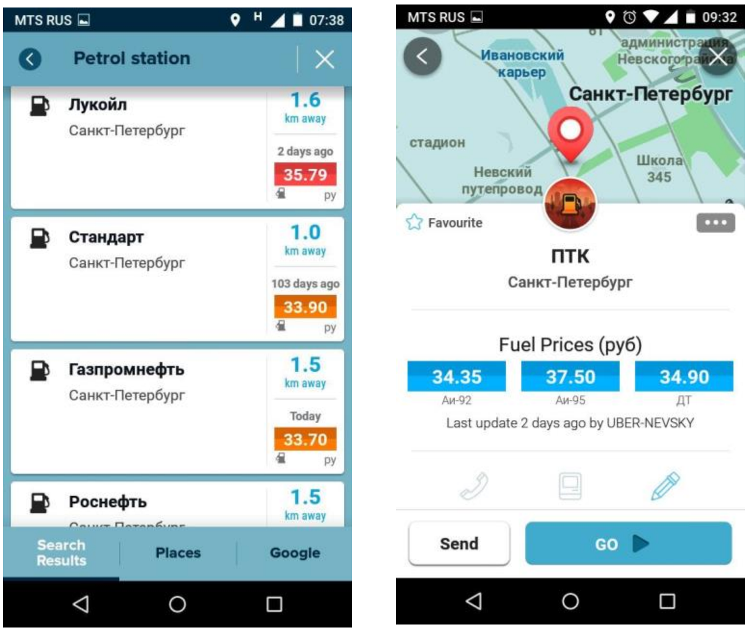


Figure 2. Waze at work

of the interface). It displayed outdated prices more seldom too.

This app featured a distance calculator, key to the comparison and the case described in the Introduction. It promoted Waze to a higher position in the comparison chart. Yet messing today's[7] prices with earlier ones spoiled the entire user experience, with the lost

[7] For the purpose of this project, "today" when used hereinafter in relation to fuel prices means prices as retrieved/seen at request/web-site visit time.

driver case in mind. And when compared with the next candidate in the test row, it demonstrated less regularity of updates.

Polttoaine

Figure 3. Polttoaine at work

This "quoter" (as sampled by Figure 3)[8], enjoyed the best updatability for its target locations. The study screenshots are given in Appendix 3.3. The data from these and a number of others were used to compare their overall updatability with Russia-based locations, in the form of comparison tables (Appendix 3.4) to define which app had the best updatability. Given the country specifics described above, Polttoaine proved to be the one.

Defining Polttoaine as the tool to focus on

Polttoaine's obvious strengths may be explained by one important difference it makes. This platform, or more precisely their database, may have good community feeds, but primarily, it appears to have stronger contributions from the local gas sellers interested in promoting their prices (see their Info tab content[9]). Otherwise, how can one account for such regular and stable updatability? Apart from its pros, Polttoaine has the following minor cons too:

- While the app launches, the emergency-scenario driver described in the Introduction, has to select his/her location manually, among the many ones hard-listed in the app; the driver may be unaware of the location;
- The app offers earlier prices that may come topmost since they are 95E-ordered by default; therefore, the driver is offered sort of overabundance of data [1,78], which is distractive;
- The app lives on officious ads (see the screen bottom of Figure 3).

These minor weaknesses were actually the launch pad for forming own attitude towards how a better app should look. I.e. it would first help the case driver out with geolocation

[8] https://play.google.com/store/apps/details?id=com.generum.polttoaine
[9] Polttoaine.netin hintatiedot päivitetään autoilijoiden ja muiden asiasta kiinnostuneiden ilmoittamien tietojen perusteella (www.polttoaine.net/index.php?cmd=info).

and, second, spare him/her time for choosing a reasonable refuel by showing only today's prices if any (with distances pre-calculated and ordered from the shortest in respect to the stop location). That scope later grew larger as another app surfaced to help build own database of fuel prices that might be valuable to market researchers if the project was to live on.

3 Building own apps and their analysis

This chapter deals mainly with own methods crystallized by the long process that the previous chapter detailed. It also describes how own apps were created to produce a combined mobile solution for gas price collection.

3.1 Forming own methods to address gas price collection

More generally, although mock-ups were at times used (see below) on every piece of paper at hand to get an idea how it would look, the main process that gave shape to the apps was via the traditional trial and error approach while coding. Other thoughts included how to make a difference, e.g. tackle the weaknesses identified in the previous chapter.

The entire project was initially about creating a single app ("User app") that would quote today's prices, if any, as displayed by Polttoaine. However, when tried for the first time in 2016, the app had to connect to a Metro MySQL table and showed only pre-inserted fake prices. The challenge was in populating that table with real Polttoaine's prices, so that the app met the project goal.

Any server-side retrieval was out of the question because of such considerations as extra load on the Metro server and its IP exposal. Exploiting proxies might be an option [1,148-149], but that would rather be outside the project scope. Therefore, data had to be collected from Polttoaine by the User app itself. That was the time when a greater part of its Java code had to be revisited to make it:

- follow a valid URL as the location request dictated (e.g. http://www.polttoaine.net/*[location]*);
- retrieve today's fuel prices if any;
- and list them to the worried driver from the emergency case.

> *DISCLAIMER:* With all the Internet ethics and local law observance due, scraping[10] and parsing techniques applied to www.polttoaine.net under this project were for educational purposes only, and with an intention to attempt creation of a better non-commercial product, given the initial emergency case for this project.

Own scrape-and-parse code (based on the *jSoup* Java library[11]) proved efficient in filtering today's prices from Polttoaine. However, the earlier prices, if any, remained intact and discarded, which looked very inefficient in terms of resource usage. It then sparked another idea: in the same code run, why not upload the historical prices to own database too?[12] Before doing this, the app would check if the location had already been requested today, thus eliminating the urge to upload same data every time a similar request was made by other users.

Clearly enough, uploading today's prices too in the same code run, before today was over, made no sense because more updates could appear on the webpage until midnight (and squeeze out earlier prices at the bottom of the webpage as seen in Appendix 4). Hence, the mobile solution at that stage needed a second app to check for the requested locations' latest prices after 00:00.

Thus, all the locations, requested today and saved in own database (via the User app) in the form of their historical prices, could now have their latest prices uploaded too. That was how the "Admin app" came into play (more detailed in the subsection *Admin app* below). It would use an alarm broadcast receiver to start a "late update" at given time (via a time picker). Once in the database, the prices would never disappear like from Polttoaine's website and might be accessed (possibly via the User app in the future), e.g. by fuel market specialists or statistics researchers.

The programming tools used for the project (chronologically since 2015):

- Eclipse IDE for Java developers;
- Android development tools for Eclipse;

---

[10] Arguably, it may be considered as simulation of the browser activity and is to become the only future technique of getting raw data from the Web [6,4;6,165-167].
[11] https://jsoup.org
[12] Of course, with all legal issues in place, i.e. once installed and before the first launch, the app should give a warning that the user agrees to share the information viewed for a specific location or any similar message. - *KK*

- Eclipse for PHP developers;
- MAMP (to run servers locally);
- phpMyAdmin (on the Metro student server);
- Android Studio;
- PhpStorm;
- yEd UML tool (http://www.yWorks.com);
- NinjaMock (https://ninjamock.com).

The testing methods included the traditional single unit runs, which if successful were integrated in the main code of the apps.

User app

With Appendix 5 showing this app’s capabilities UML scheme and activity flowchart, the following is a use case that claims to match the emergency described in the Introduction.

1. The worried driver (User) is stationary in or near a Finnish city (hopefully covered by Polttoaine’s database): the gas tank is (almost) empty. There is need for a refuel.
2. User may be very unfamiliar with the place and there is barely anyone to ask the way to the nearest gas station, leave alone the prices.
3. User may wish to consider distance-vs-price options if given any. He or she either Googles for any nearest gas station or launches the User app.
4. User has the location service checked. If disabled, the app prompts User to enable or exit.
5. User has Internet connection checked. If disabled, the app prompts User to enable or exit.
6. User has the position coordinates written to the shared preferences right on the spot. The app takes some time to confirm User’s location. Although unaware of the place, User can tell the city or municipality by the road signs. Reverse geocoding is used (coordinates to an address). If wrong, rerun or exit (use another app?).
7. User taps OK, and if the location is among the ones hard-coded in app, User sees a list of gas stations with today's prices only and other info starting from the nearest gas station: station icon, address, bonuses/comments if any, fuel prices (e95, e98, Diesel), distance.

8. If the location is being requested for the first time today, User has to agree to share the information (s)he is going to see with the local driver community (such a dialog may pop up at the very launch). User sends nothing if another driver has requested the location today.
9. This way, the app saves the location's earlier prices to own database if they are absent there.
10. User taps on an option to open Google Maps, which takes over with the navigation (pending).
11. User closes the app when convenient.
12. Compared to Polttoaine's, this one makes a difference:
    - data order-wise, no earlier prices shown (except in the future, they may be as Figure 7 shows), options start with the nearest station, the drivers sees kms too;
    - data updatability and persistence-wise, the app updates own database as described in Appendix 5, and spares the driver's mobile data traffic if other drivers have already requested the same location today or yesterday;
    - usability-wise, no ads.

With the User app tentatively titled as *GasGuide*, mock-ups further detail how the UI was originally conceived and imagined (loc service and Internet connectivity checks are skipped for brevity):

Figure 4. Confirm loc

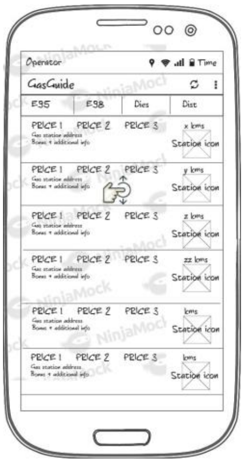


Figure 5. Choose and tap

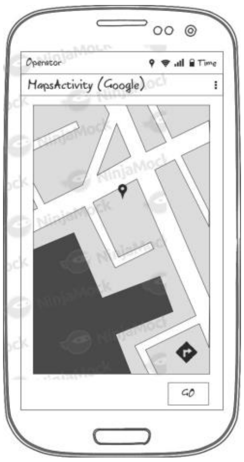


Figure 6. Navigate (pending)

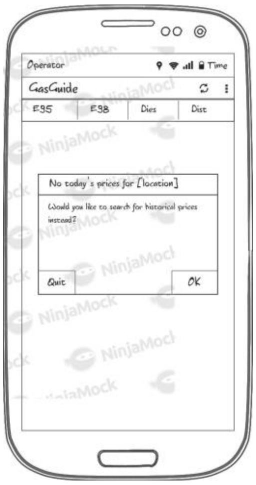


Figure 7. No-today's-prices case (pending)

As Figure 4 demonstrates, the user has three options when geocoding is over, i.e. either retry if wrong, quit the app or confirm his/her location to carry on. The next step (Figure 5) is waiting for available options to download and scrolling through them. The list features fuel prices and other data just retrieved from Polttoaine, so that they match the app header (directly under e95 and other types of fuel, with the distance column finishing the row).

The data include gas station addresses, fuel bonuses and other information the user may need. Each fuel brand has its icon to the right if available. When the user makes a decision, a tap on the choice leads to Google Maps (Figure 6).

As a location-aware app, this one had to follow the recommendations in the field, especially in terms of battery consumption [7]. For that reason, e.g. the LocationManager() method used in the app was first tried on its own in a short series of unit tests, with toasting and logging available with Android Studio that helped add a few adjustments to provide for better fix correction and control over the requestLocationUpdates() known for shortening battery life. The following code excerpt demonstrates the final version based on the quoted resource.

```
@Override
public void onLocationChanged(Location location) {
     // run a loc fix accuracy/update check here before writing to shared prefs
     // using the source's recommendation to maintain a current best estimate
     final int TWO_MINUTES = 1000 * 60 * 2;
     String latitude, longitude, accuracy, timestamp, provider;
     // as read from shared prefs
     mSharedPref = mCntx.getSharedPreferences(SHARED_PREF_LOC_LAT_LNG,
     Context.MODE_PRIVATE);
     latitude = mSharedPref.getString("Lat", "");
     longitude = mSharedPref.getString("Lng", "");
     accuracy = mSharedPref.getString("Acc", "");
     timestamp = mSharedPref.getString("Ts", "");
     provider = mSharedPref.getString("By", "");
     if (!latitude.equals("") && !longitude.equals("") && !accuracy.equals("")
          && !timestamp.equals("") && !provider.equals("")) {
          // Check whether the new location fix is newer or older
          long timeDelta = location.getTime() - Long.parseLong(timestamp);
          boolean isSignificantlyNewer = timeDelta > TWO_MINUTES;
          boolean isSignificantlyOlder = timeDelta < -TWO_MINUTES;
          boolean isNewer = timeDelta > 0;
          // If it's been more than two minutes since the current location, use
             the new location
          // because the user has likely moved
          if (isSignificantlyNewer) {
               // Toast.makeText(mCntx,
               // "Writing loc to SharedPrefs since it's significantly newer",
               // Toast.LENGTH_LONG).show();
```

```
        Log.d("Provider ", "Writing loc to SharedPrefs->significantly newer");
            writeLocToSharedPrefs(location);
            return; // we just write to shared prefs and quit the method
        } else if (isSignificantlyOlder) {
            return;
        }
        // otherwise
        // Check whether the new location fix is more or less accurate
        Float accuracyDelta = (location.getAccuracy() -
        Float.parseFloat(accuracy));
        boolean isLessAccurate = accuracyDelta > 0;
        boolean isMoreAccurate = accuracyDelta < 0;
        boolean isSignificantlyLessAccurate = accuracyDelta > 200;
        // Check if the old and new location are from the same mProvider
        boolean isFromSameProvider = isSameProvider(location.getProvider(),
        provider);
        // Determine location quality using a combination of timeliness and
        accuracy
        if (isMoreAccurate) {
            Log.d("Provider ", "Writing loc to SharedPrefs->more accurate: "
        + location.getAccuracy() + " vs " + accuracy);
            writeLocToSharedPrefs(location);
        } else if (isNewer && !isLessAccurate) {
            Log.d("Provider ", "Writing loc to SharedPrefs->newer&more
        accurate: " + location.getAccuracy() + " vs " + accuracy);
            writeLocToSharedPrefs(location);
        } else if (isNewer && !isSignificantlyLessAccurate &&
        isFromSameProvider) {
            Log.d("Provider ", "Writing loc to SharedPrefs->
        newer&!isSignificantlyLessAccurate&isFromSameProvider");
            writeLocToSharedPrefs(location);
        }
    } else {
        // i.e. the app just got installed => no shared prefs available, at
        least for lat/lng
        writeLocToSharedPrefs(location);
    }
}
```

```
private void writeLocToSharedPrefs(Location l) {
    //        Toast.makeText(mCntx, "Saving loc",
    //                Toast.LENGTH_SHORT).show();
    String lat = "" + l.getLatitude();
    String lng = "" + l.getLongitude();
    String acc = "" + l.getAccuracy();
    String ts = "" + l.getTime();
    String by = l.getProvider();
    mSharedPref = mCntx.getSharedPreferences(SHARED_PREF_LOC_LAT_LNG,
    Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = mSharedPref.edit();
    editor.putString("Lat", lat);
    editor.putString("Lng", lng);
    editor.putString("Acc", acc);
    editor.putString("Ts", ts);
    editor.putString("By", by);
    editor.apply();
}
```

In line with the flowchart introduced in Appendix 5 for the User app, writing the coordinates thus received from the device location tools (GPS, network or Wi-Fi or all combined) to the shared preferences starts separately, under an async task [8]. This helps keep the UI thread busy with other things in parallel (popping dialogues, toasting warnings and prompts to the user, etc).

Similarly, other pieces of code, as recommended excerpts were first tried and tuned outside the main code of the app; then, if successful, they would be included in the app and checked again for any runtime errors. At this stage, the entire app would be saved under the next number version and so on, so that there always was an opportunity to roll back to the previous stable version.

Admin app

The reason to create a second app as part of the combined mobile solution hereunder was briefly introduced in the beginning of this chapter. Still, a sample use case is required to give a detailed view of what this app can offer to an admin responsible for keeping an eye on own database of fuel prices, provided that the User app does it regular job as the popular locations identifier in the combined mobile solution scheme.

1. Admin knows that the User app has added more locations, requested today, to own database. It means that their historical prices are already tabled in own database, but their today's prices are not, since this is the logic of the Uses app: "today is not over yet, and more price updates may be appearing on Polttoaine's webpage. Why save them in the middle of today? Wait until it is over", etc.
2. What will happen to them by the end of today? They will stay displayed on Polttoaine well into tomorrow, just to be placed under or written over by more updates to come (i.e. the three-day approach suggested in Appendix 4).
3. In order to upload the latest prices of today (of the locations requested today via the User app), Admin either schedules an automatic upload (via a timed alarm trigger) or waits till the end of the day, then uploads these locations' latest prices manually.
4. The auto mode implements Wi-Fi (un)locking to save the battery.
5. UI-wise, Admin can check the next alarm time, i.e. when the auto mode is scheduled. (If there is a dash instead, no alarm is set.)
6. Admin can use the default time picker to schedule/start an alarm (any time).
7. Admin has various options available:
    - Simulate the User app, esp. when physically outside of Finland
    - Cancel timed alarms

- See an activity report (auto/manual mode)
- Clear the report

8. Admin taps on "Manual" to activate the manual mode, where Admin can:
    - See locations requested yesterday (if any)
    - See requested two days ago (if any).
    - Use either to update own database (JSON format, if data available) from the device, the server being spared extra loads. The data are saved to the shared preferences for persistency.
    - See the previous session upload data (JSON format, in a scrollable window at the screen bottom)
9. Admin taps back to return to the main activity (alarm-based, auto mode).
10. Admin closes the app when necessary.

If one evaluates the entire two-app approach, it can be seen that, database-wise, the User app saves the locations requested today in a separate table, then after 00:00, a server cron [9] moves them, say, to a yesterday locations table, in order to keep both the former empty and ready for the new day, and the latter populated for roll-back cases (as demonstrated in Figure 43, Appendix 5). The Admin app then steps in and checks on the recent locations by the end of today, in that second table. Now that today is over, the Admin app downloads each location’s details from Polttoaine and saves them as recent prices to the respective tables in own database.

This way, one can be sure that no more today’s prices for these locations will appear as they used to (see Appendix 4 for increments in today’s prices) on Polttoaine’s webpage because today has gone, and tomorrow has come. And that way, one can be sure that all popular locations’ prices stay in own database. That was the logic behind introducing the Admin app to reinforce the mobile solution for gas price collection under this project. In addition, yesterday's locations migrated from the first table are purged by a second cron on the server and saved into a JSON file for the sake of item 7 above (being able to upload from two days ago, should anything go wrong in the auto mode).

In conclusion to this theoretical part, a third cron would be necessary to purge own database of outdated table content, whose duration would be up to the admin to decide on. For the sake of this project, any locations with prices dating two weeks back or earlier, are removed to ZZ_[location] archive tables; tackling their growth remains outside the scope hereof.

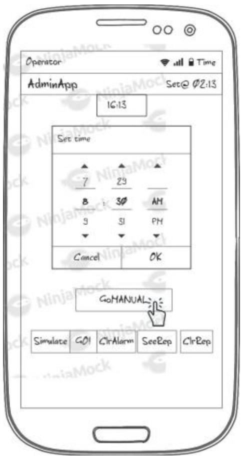


Figure 8. Set alarm (auto mode)

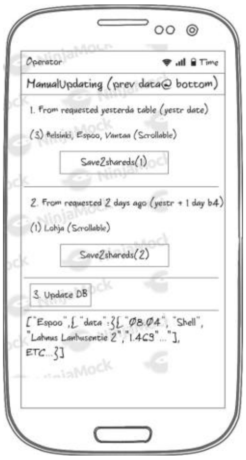


Figure 9. Update from SharedPrefs (1 or 2)

As to the look and feel of the Admin app, the mock-ups in Figures 8 and 9 demonstrate the initial idea (two separate activities with various options).

Designing the Admin app was in parallel with developing database interactions based on the tree-day approach highlighted in Appendix 4, its upload functionality primarily borrowed from the User app.

3.2 Limitation of the project

With the User app completed by February of 2016, and the Admin app surfacing in late 2016, developing them to stable versions has taken almost all the remainder of the time stipulated for the 2017 spring graduation. Due to a part- and then full-time job with a Russian company in St Petersburg, all Metropolia courses being completed, the greater part of the project was carried out and tested in St Petersburg, rather than any major city in Finland. It presupposes that some reverse geocoding limitations may also ensue in the combined mobile solution scheme, given that, as experience has shown, Google may at times choke on municipal areas of different countries, especially when there are many subdivisions: the adrress.getLocality() vs the address.getSubAdminArea() [10].

The Google over-query limit (the User app)

At the time of experimenting with the Google Maps API for distance calculations for this project, only 50 requests per second were allowed for no-licence uses. Presently, the quota may have changed; still the same limit is in force [11]. For this reason, the User app has to force delays in calculating distances between the available gas stations' and the user's addresses.

As tests show, if there are up to 50 locations to process in this regard, and if the calculations are limited to two distances per second, there is usually only 1% of failures. However, in this case the app takes much time in listing the results (the distances are integral to the UI). Moreover, for five distance calculations per second, the malfunction is diverse, e.g. 29 : 19, 14 : 34 or 19 : 29 (i.e. go vs no-go results); yet the time spent is noticeably shorter.

A major recent problem that may have related to the same, is the User app's inability to activate its former Google Maps part, which may be due to certificate alterations in the provider's policy.

The community approach (the User app)

Despite feeding on Polttoaine's data, the User app would perform better if it allowed for driver community input. For the time being, the app's UI has no feature as such.

On the other hand, the User app is indeed dependent on the local community requests. I.e. the more frequently drivers ask for today's prices, the more up-to-date is own database.

Archive tables growing in volume (the Admin app)

As the tabled data in own database become two-week old, they are removed into their respective archive tables, named ZZ_[location], by a server cron programmed to fulfil this on a regular basis. It helps to keep actual information within limits, so that requests are served faster; still the ZZ tables may become overpopulated with time. So far, the mobile solution under this project has no means to tackle this problem.

4 Results

With the project limitations taken into account, the combined mobile solution does almost everything as it should, except for the former navigation. It definitely needs more attention if it is to claim to become a remote competitor of its counterpart app from Polttoaine.

4.1 Overall performance of the two-app approach

In order to demonstrate how this approach works in real life, following are mobile screenshots of both the apps in action, in the same order as mocked up in the previous chapter. For the database approach, more screenshots are added after those of the apps.

User app

If the app's mock-ups are compared to the actual look and feel (Figures 10, 11), there seems to be only colour, the pending no-today's (Figure 13) and Google Maps scenarios that differentiate them. Still, most of the functionality declared is in place, except the Google Maps case (detailed in 4.2). For easier comparison of the User app vs its Polttoaine's counterpart, one of the latter's screen shots is added, too (as shown in Figure 12).

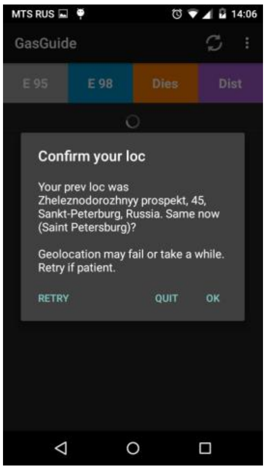


Figure 8. Confirm loc

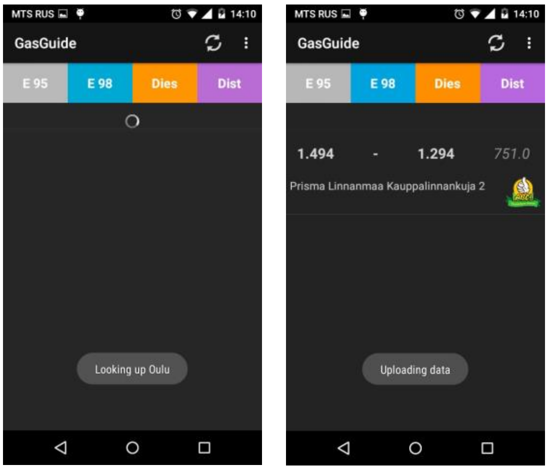


Figure 9 (two). Choose and tap

Figure 10. Polttoaine's app

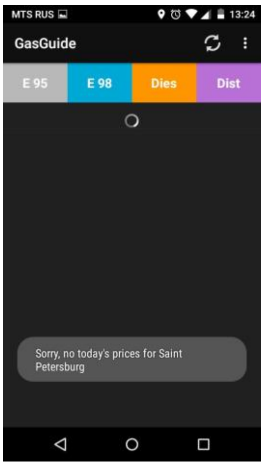


Figure 11. No-today's-prices case (pending)

As Figure 10 confirms, the test location was St Petersburg, yet to simulate a Finnish location, Oulu was chosen and hard-coded into the User app to see the actual prices come into light. And both the apps showed first 1:2 and then 2:3 options (see Figures 11, 12 and 14) in favour of the Polttoaine's app.

Figure 14 demonstrates also that data retrieved almost at the same time by the two apps, may be slightly different (compare the official Polttoaine webpage with the output of both the apps; as well as note the different distance values).

Tapping on either option offered by the Use app (Figure 14) failed to open the Google Maps activity for unknown reasons so far.

Finally, before the usual run of the code was altered for Oulu, the no-today's prices modelled option (Figure 13) was obviously handled correctly, since this location was not among the Finnish ones, hence no data for it.

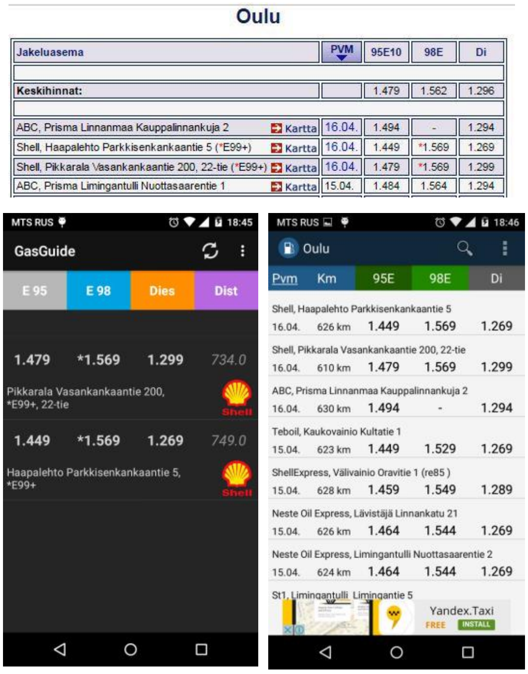


Figure 12. Own app's input appears to match the webpage

Admin app

Similarly, the other app belonging to the combined mobile solution is visualised below. Its provisional title is *jSouper DailyUpdateLauncher*; it was named so after a library with a resembling name (footnoted in 3.1) was experimented with and lay as the foundation. Manual mode examples (auto mode excluded for brevity except its report) include:

Figure 15. Set alarm (auto mode)

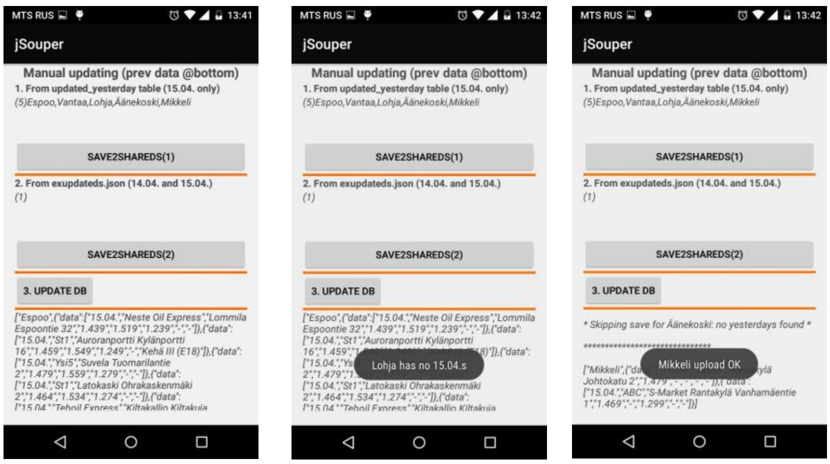


Figure 16 (three). Update from SharedPrefs (1) and reporting option available too

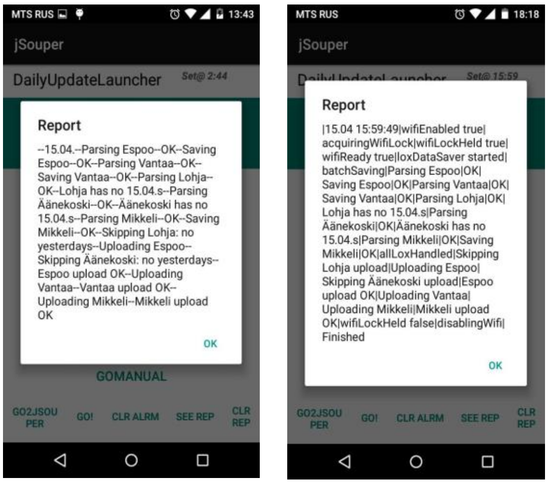


Figure 17 (two). Reporting options (auto "|" vs manual "--"). Note the battery saving technique used in the auto mode (Wi-Fi (un)locking).

As Figures 15 and 16 demonstrate, the update process was via the manual mode, after the data had been saved to the shared preferences of the device. The uploading was toasted and duly reported as it should be (Figures 16, 17). Note the minor difference in the way the auto and manual modes showed their reports separators ("|" for auto vs "--" for manual).

Database

Since these two apps are integral to the combined mobile solution for gas price collection, its overall performance will be incomplete without demonstrating the back-end process, i.e. what happens in the database. The following figures are meant to give an idea in this regard.

If the way the previous screenshots appear is viewed in chronological sequence, it can be noted that:

1. The test user is in Oulu requesting fuel prices for today via the User app. It then sends that information to the database; since it shows that it is uploading data, the location is being requested for the first time. It means that Oulu should appear in the today's locations table (as Figure 18 shows).

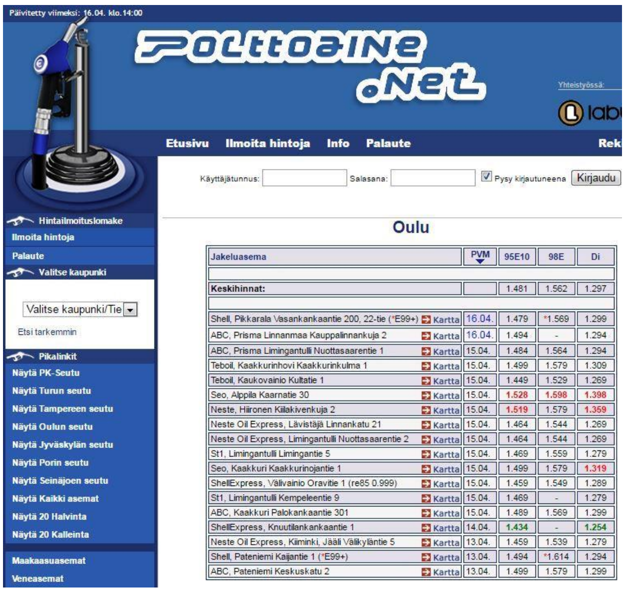


Figure 19. Oulu data (official)

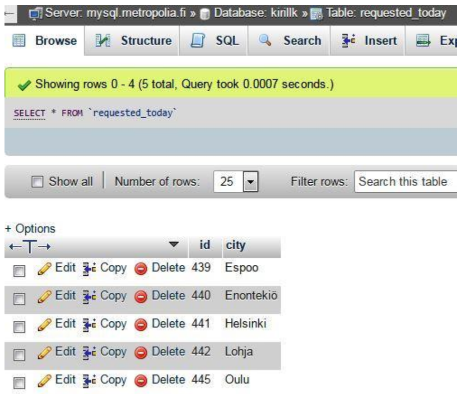
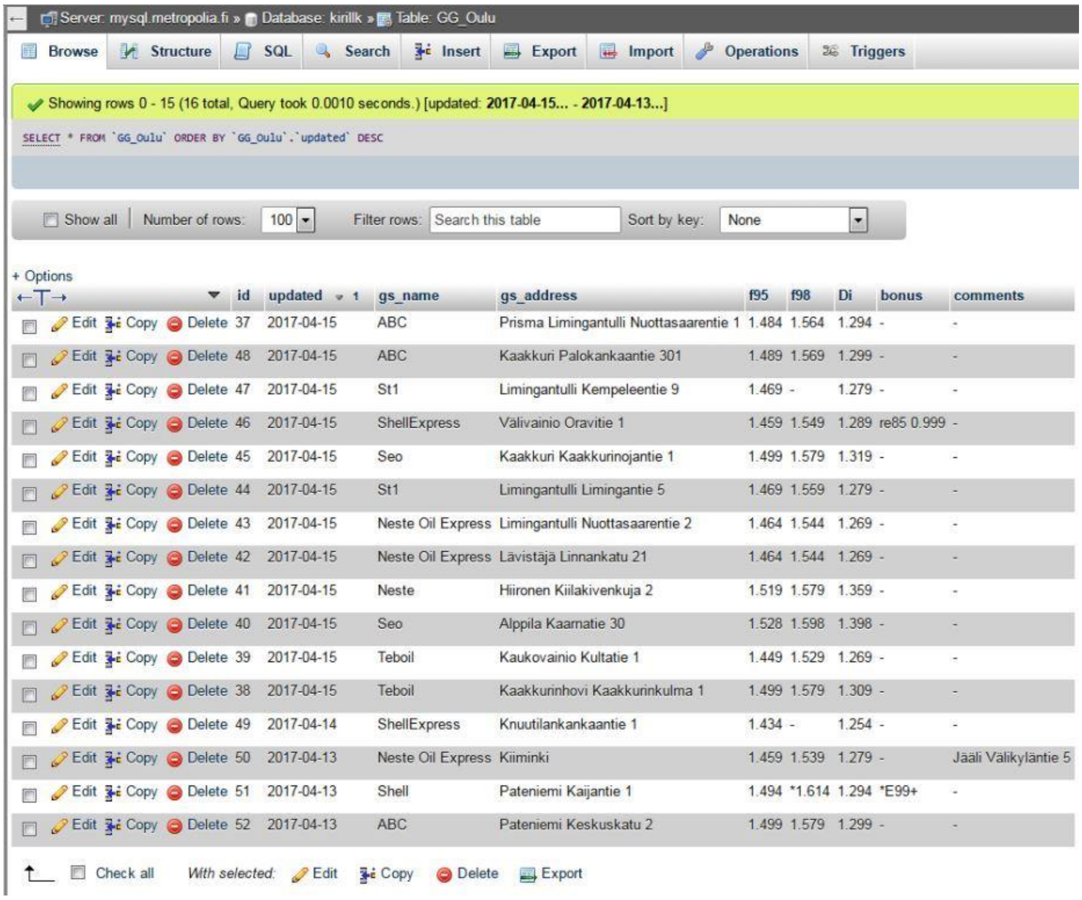


Figure 18. Oulu is in the requested_today table

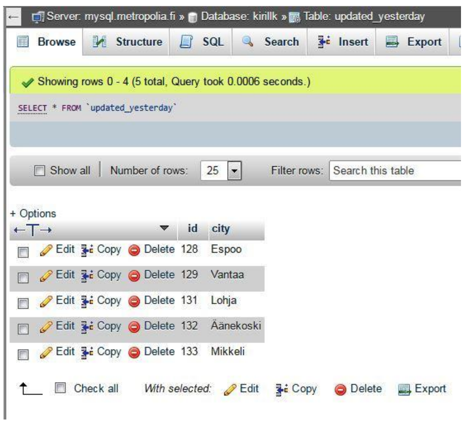
2. All but today's (16.04) prices from Oulu (Figure 19) are supposed to have been uploaded to own Oulu's table in the database. Figure 20 confirms it.



id	updated	gs_name	gs_address	f95	f98	Di	bonus	comments
37	2017-04-15	ABC	Prisma Limingantulli Nuottasaarentie 1	1.484	1.564	1.294	-	-
48	2017-04-15	ABC	Kaakkuri Palokankaantie 301	1.489	1.569	1.299	-	-
47	2017-04-15	St1	Limingantulli Kempeleentie 9	1.469	-	1.279	-	-
46	2017-04-15	ShellExpress	Välivainio Oravitie 1	1.459	1.549	1.289	re85 0.999	-
45	2017-04-15	Seo	Kaakkuri Kaakkurinojantie 1	1.499	1.579	1.319	-	-
44	2017-04-15	St1	Limingantulli Limingantie 5	1.469	1.559	1.279	-	-
43	2017-04-15	Neste Oil Express	Limingantulli Nuottasaarentie 2	1.464	1.544	1.269	-	-
42	2017-04-15	Neste Oil Express	Lävistäjä Linnankatu 21	1.464	1.544	1.269	-	-
41	2017-04-15	Neste	Hiironen Kiilakivenkuja 2	1.519	1.579	1.359	-	-
40	2017-04-15	Seo	Alppila Kaarnatie 30	1.528	1.598	1.398	-	-
39	2017-04-15	Teboil	Kaukovainio Kultatie 1	1.449	1.529	1.269	-	-
38	2017-04-15	Teboil	Kaakkurinhovi Kaakkurinkulma 1	1.499	1.579	1.309	-	-
49	2017-04-14	ShellExpress	Knuutilankankaantie 1	1.434	-	1.254	-	-
50	2017-04-13	Neste Oil Express	Kiiminki	1.459	1.539	1.279	-	Jääli Välikyläntie 5
51	2017-04-13	Shell	Pateniemi Kaijantie 1	1.494	*1.614	1.294	*E99+	-
52	2017-04-13	ABC	Pateniemi Keskuskatu 2	1.499	1.579	1.299	-	-

Figure 20. Oulu data (retrieved into own database)

3. Moreover, the Admin app revealed the locations requested yesterday (15.04) were Espoo, Vantaa, Lohja, Äänekoski, and Mikkeli (Figure 16). It means that their data should be in their respective tables of own database (see the thesis last page figure too). This presumption is confirmed by Figure 21.



id	city
128	Espoo
129	Vantaa
131	Lohja
132	Äänekoski
133	Mikkeli

Figure 21. Locations requested yesterday are store in a separate table

4. Given that a manual upload of their prices was attempted (as indicated in Figures 16, 17), all of them except today's (16.04) should be in the respective tables of the database. For brevity, only the first three locations vs their Polttoaine's counterparts are shown in Figures 22-24.

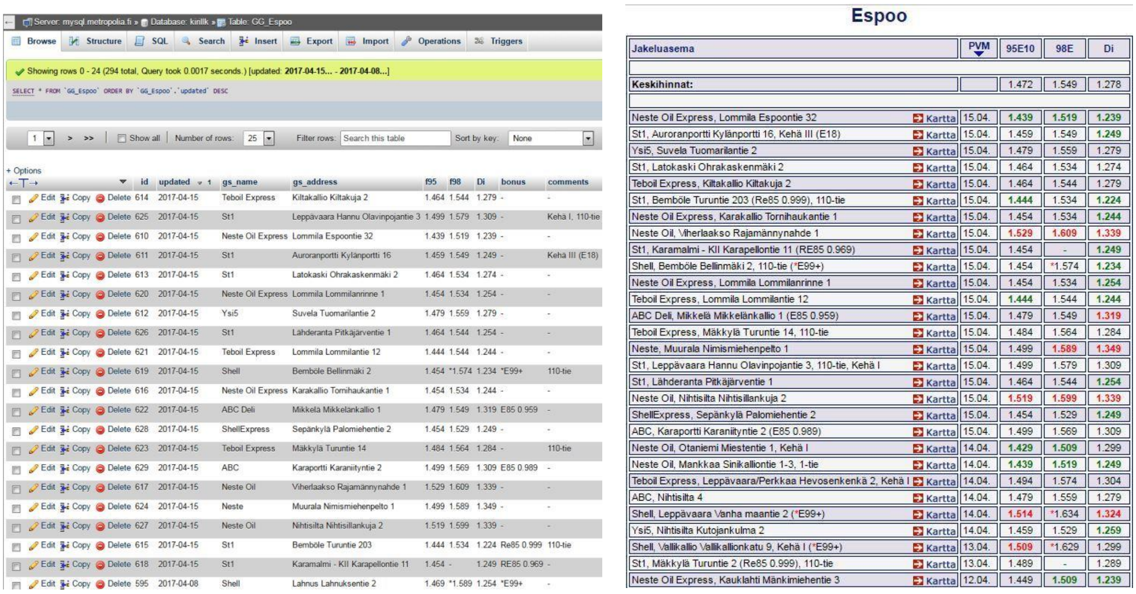


Figure 22. Espoo handled

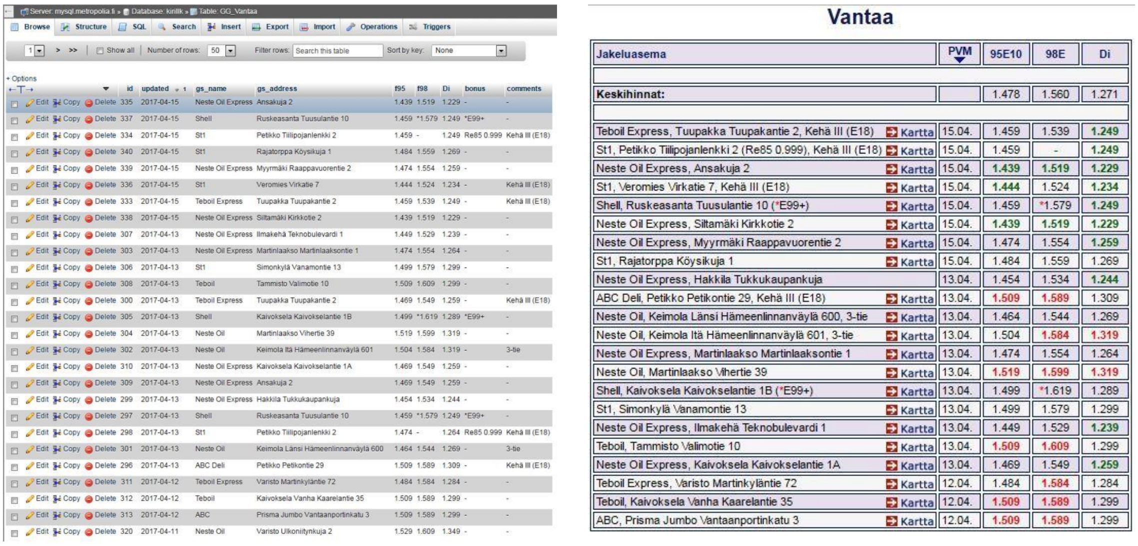


Figure 23. Vantaa processed

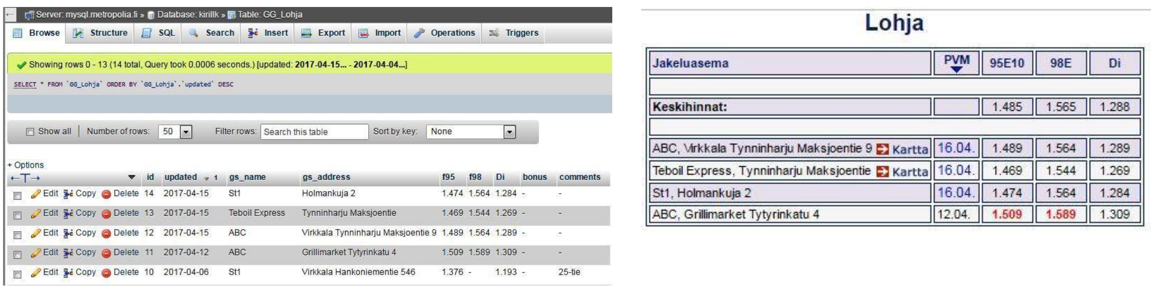


Figure 24. Lohja inserted too

For space and brevity, no cron job screenshots are furnished for this paper.

4.2 Errors explained

Unfortunately, the major problem of the combined mobile solution is the sudden malfunction of the Google Maps activity as shown in Figure 25. Own approaches to address this error included online searches, yet so far no valid solution has been found, except redesigning the entire activity and test-run it from scratch.

```
04-16 18:48:40.452 8056-12233/com.gg.yeah W/DynamiteModule: Local module
  descriptor class for com.google.android.gms.googlecertificates not found.
04-16 18:48:40.453 8056-12233/com.gg.yeah W/DynamiteModule: Failed to load
  module via V2: java.lang.ClassNotFoundException: Didn't find class
  "com.google.android.gms.dynamite.DynamiteModule$DynamiteLoaderClassLoader"
  on path: DexPathList[[zip file "/data/app/com.gg.yeah-1/base.apk", zip file
  "/data/app/com.gg.yeah-1/split_lib_dependencies_apk.apk", zip file
  "/data/app/com.gg.yeah-1/split_lib_slice_0_apk.apk", zip file "/data/app/com
  .gg.yeah-1/split_lib_slice_1_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_2_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_3_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_4_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_5_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_6_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_7_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_8_apk.apk", zip file "/data/app/com.gg
  .yeah-1/split_lib_slice_9_apk.apk"],nativeLibraryDirectories=[/vendor/lib,
  /system/lib]]
04-16 18:48:40.470 8056-12233/com.gg.yeah I/DynamiteModule: Considering local
  module com.google.android.gms.googlecertificates:0 and remote module
  com.google.android.gms.googlecertificates:2
04-16 18:48:40.470 8056-12233/com.gg.yeah I/DynamiteModule: Selected remote
  version of com.google.android.gms.googlecertificates, version >= 2
```

Figure 25. Unhandled error in calling the Google Maps activity within the User app

Among other problems encountered in this project was the formation of valid links to reach a target location. E.g. http://www.polttoaine.net/Espoo were easily followed by the code, whereas such as http://www.polttoaine.net/ /Keha_%20III%20(E18), i.e. Kehä III (E18), or http://www.polttoaine.net/Pyha_ja_rvi (i.e. Pyhäjärvi) were not.

Own methods were introduced to overcome the latter. More specifically, for umlauts and the space substitution in the links:

```
String legacy_coder(String location) {
    // forming again the ugly url string
    StringBuilder sb = new StringBuilder("");
    for (int index = 0; index < location.length(); index++) {
        char aChar = location.charAt(index);
        String s = "";
        if (aChar == 'ä') {
            aChar = Character.MIN_VALUE;
            s = "a_";
        }
        if (aChar == 'Ä') {
            aChar = Character.MIN_VALUE;
            s = "A_";
        }
        if (aChar == 'ö') {
            aChar = Character.MIN_VALUE;
            s = "o_";
        }
        if (aChar == 'Ö') {
            aChar = Character.MIN_VALUE;
            s = "O_";
        }
        s += aChar;
        sb.append(s);
    }
    return sb.toString().replaceAll("\\p{C}", "");
}
```

```
…
this.keyword = keyword.replaceAll(" ", "%20").trim();
…
```

After running these, links to the Polttoaine's 150-location webpage were followed by the code and opened gates to their html code without any problem.

5 Evaluation of the results

What follows is an attempted objective evaluation of the performance of own combined mobile solution for gas price collection, including comparisons with Polttoaine's app. Overall, but for the single major drawback represented by the Google Maps activity malfunction, the solution works, provided the local drivers use it and agree to share the information they get from Polttoaine.

5.1 Efficiency of own methods

Despite the need for further code rearrangement and more effective methods, own mobile approach has shown that:

1. The test locations are recognised correctly, which is much help to the worried driver case described in the Introduction.
2. Gas station options, though limited by the Google over-query limit (chapter 3.2), are true to those displayed on the gas prices provider's website.
3. Own database gets populated with only most requested location's data and can be developed further to give access to interested parties.
4. Own database uploading can be handled by an admin in either the auto or manual mode, with reporting and toasting available.
5. No ads are shown in the process (no distraction/disturbance).
6. Failures include the map activity and the pending no-today's prices case, thus historical prices are unavailable at present. In addition, no own app's tools have been designed to counter the archive tables overpopulation mentioned in chapter 3.2.

Next subsection deals with the key feature comparison to the Polttoaine app to see benefits and drawbacks of own approach.

5.2 Benefits and drawbacks vs studied tools

Although claiming no rivalry with the counterpart app, own apps offer the following pros and cons when compared with the Polttoaine app (in the same order as 5.1 above):

1. Unlike the comparison app, own one recognises current location.
2. Polttoaine's app may at times deviate from its cousin webpage, whereas own app scrapes them as they are (see Figure 14).
3. No official ways are known to access the Polttoaine database historical prices, whereas own database features a fair collection of them, provided local drivers are motivated to use the mobile solution from now on.
4. Unlike the comparison app, own one offers some database update management tools.
5. Own apps do not feed on ads.
6. Drawbacks include no own app's option to use local driver community input (unlike Polttoaine's app that has it); no navigation to the gas station chosen (Polttoaine has it too); no tools to counter the archive tables' growth in own database; the tables there are numerous, unrelated, and with repeated structures.

As a major side note, back to Figure 14, the distances calculated by both the apps do differ. Therefore, this distance was resized on the Google Maps webpage to show the following:

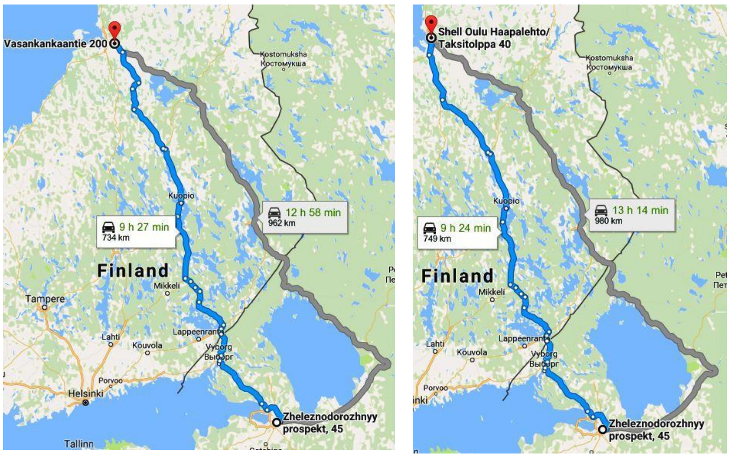


Figure 26. Deviation in distance calculations by the two apps

Comparing the results reveals that own app's length is closer to the resized one in both cases. This speaks for one major benefit in favour of the User app.

With these finishing this sub-section, it can be concluded that keeping the comparison in mind will help both parties in improving their apps in future and implementing better usability scenarios.

6 Conclusion

In conclusion, it should suffice to state that the initial emergency pictured in the Introduction, with the worried driver, unaware of the location (s)he is at and with an emptying gas tank, may have received a new solution in the form of the two apps and database detailed in this project. Although for the time being, the Google Maps activity lacks operation readiness, the system seems to qualify generally to proceed with further development as a promising, possibly, more driver-friendly tool.

As for the Admin app, its tasks may as well be server-run, yet the mobile solution in question seems to provide for more tangible control over the process right on the go.

It goes without delving into much detail that the system will depend on the local driver community too, since it is their location requests that are the prime mover for this project, and the latter was not meant to scrape and parse all the locations Polttoaine covers daily. That would seem to offer another topic for a thesis of a web security or anti-bot student.

The Google over-query, as a limitation for this project, may be surpassed by fine-tuning the process milliseconds suitable for the occasion. On the other hand, the emergency driver from the Introduction may feel frustrated at having to list through multiple options if there are so many as 20 or 50, and may as well be content with choosing among just 5-8 stations instead.

The combined mobile solution for gas price collection introduced in this project could be more efficient if the following requirements were met:

- The Google Maps activity is back and operational;
- The over-query limit is curbed via caching or other means;
- The single geolocation mode is reinforced by a manual input alternative, should the driver ignore the need for position recognition;
- The show-earlier prices case is introduced and tested for whether it is actually necessary;
- The archive tables' growth is checked by a fourth cron that stores outdated information in JSON-formatted files;
- The repeated structure of the database tables is substituted by more efficient tools in order to save space and time and eliminate the need for archiving.

Further study may be recommended to build on the project by introducing non-MySQL methods too. Additionally, other collection methods than getting updates from gas providers or driver communities, as described in Appendix 1, may encourage the continuation of this project.

References

1 Chauhan S, Panda N.K. Hacking Web Intelligence. Open Source Intelligence and Web Reconnaissance Concepts and Techniques. Waltham, MA: Syngress (Elsevier); 2015.

2 Google. Google Play [online].
URL: https://play.google.com/store/search?q=fuel%20prices%20app&c=apps.
Accessed 12 April 2017.

3 Dhanrajani A. Google Maps making stressful times easier [online]. 2015.
URL: https://maps.googleblog.com/2015/10/google-maps-making-stressful-times.html.
Accessed 12 April 2017.

4 Alvarez E. Google Maps for iOS displays gas prices along your route [online]. 2015.
URL: https://www.engadget.com/2015/12/14/google-maps-ios-gas-prices.
Accessed 12 April 2017.

5 Kriel C. Google Maps for iOS catches up to Android, gains offline navigation, up-to-date gas prices, and more [onlime]. 2015.
URL: http://siliconangle.com/blog/2015/12/15/google-maps-for-ios-catches-up-to-android-gains-offline-navigation-up-to-date-gas-prices-and-more.
Accessed 12 April 2017.

6 Turland M. php|architect's Guide to Web Scraping with PHP. Toronto, ON M3H 1B7: Marco Tabini & Associates, Inc., 2010

7 Google. Developers. Location Strategies [online].
URL: https://developer.android.com/guide/topics/location/strategies.html.
Accessed 17 December 2016.

8 Google. Developers. AsyncTask [online].
URL: https://developer.android.com/reference/android/os/AsyncTask.html.
Accessed 15 February 2016).

9 Linux. crontab(5) - Linux man page [online].
URL: https://linux.die.net/man/5/crontab.
Accessed 16 April 2017.

10 StackOverflow. Android geocoder reverse location - reliable way to get the city? [online].
URL: http://stackoverflow.com/questions/21921460/android-geocoder-reverse-

location-reliable-way-to-get-the-city.
Accessed 5 January 2017.

11 Google. Google Maps API. Premium Plan Usage Rates and Limits [online].
URL: https://developers.google.com/maps/premium/usage-limits.
Accessed 16 April 2017.

Appendix 1. Challenges of OCR

The images on this page originate from Google Street View, own photo camera, and own dashcam respectively to the order they appear here, unnumbered. When trying to recognise these via the mentioned Tesseract-PHP script, the results were discouraging, since the colour digits were never read correctly.

Own attempts to change these colour images into black and white gave no positive results: when taking pictures, one had to make sure the price tag was ideally centred. That, however, never helped even after treatment by various methods (as self-explanatory Figures 27-31 demonstrate).

Figure 27. Cropped

Figure 28. Cropped, negated

Figure 29. Cropped, negated, greyscale (CNG)

Figure 30. CNG+lit+contrasted

Figure 31. CNG+lit

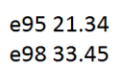
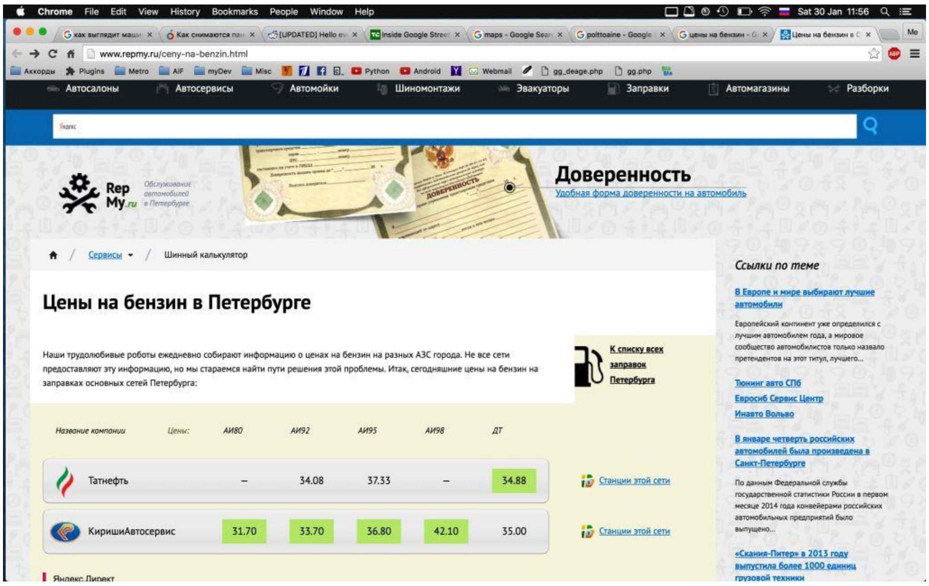

e95 21.34
e98 33.45

Figure 32. A .doc file screen shot was read like "e95 21.34e98 33.45"

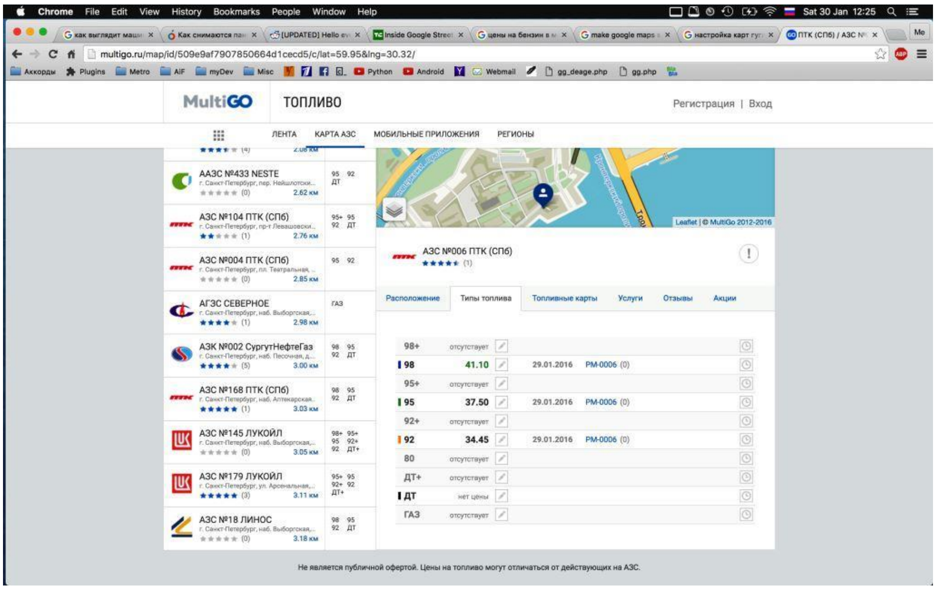
Unlike colour images, BW screenshots, as simple as black text vs white background (Figure 32), were read without any problem. Those attempts clearly explained, carrying on with two separate topics would take more time than allocated for writing one thesis.

Appendix 2. Retrieving country-specific fuel data

1. For this item, see below a screenshot of a website specialised in quoting only average fuel prices in St Petersburg. The text in fine print underneath the title (*Gas prices in St Petersburg*) reads that their hard-working bots are daily busy collecting fuel data from various gas stations of the city. Not every station is ready to provide this information, so today's major choices are the following. And only two options follow.



2. Next image demonstrates that prices shown are for evaluation only and subject to change by gas stations:



3. An excerpt from a correspondence history between me and a local Neste station (translated by me):

From: DS Spb <ds.spb@neste.com>

Subject: Re: prices available at your website

Date: 8 February 2016 at 10:09:25 GMT+3

To: Kirill Kuvshinov <Kirill.Kuvshinov@metropolia.fi>

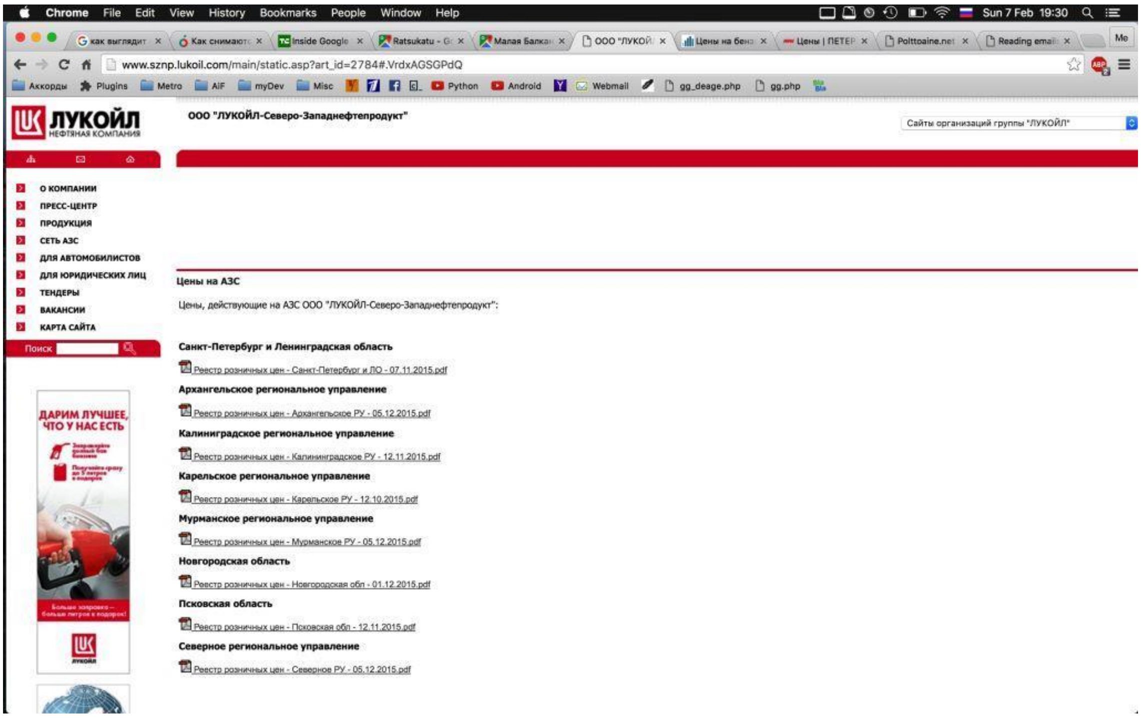
Good morning Kirill,

If you are interested in our gas station prices, the site cannot provide this information. You may want to visit any of our stations individually to learn about current price. Fuel delivery can be calculated by a special calculator if you pls follow this link.

http://www.nesteoil.ru/ru/content/calculator

> 7 February 2016, 19:05 Kirill Kuvshinov <Kirill.Kuvshinov@metropolia.fi> wrote:
>
> Good evening,
>
> Kindly advise where I can see fuel prices on your website.
>
> Best regards,
>
> Kirill
>
> --
>
> Best regards,
>
> Corporate sales office
>
> Neste Saint Petersburg, OOO
>
> +7 (812) 703 06 15
>
> ds.spb@neste.com
>
> www.neste.ru

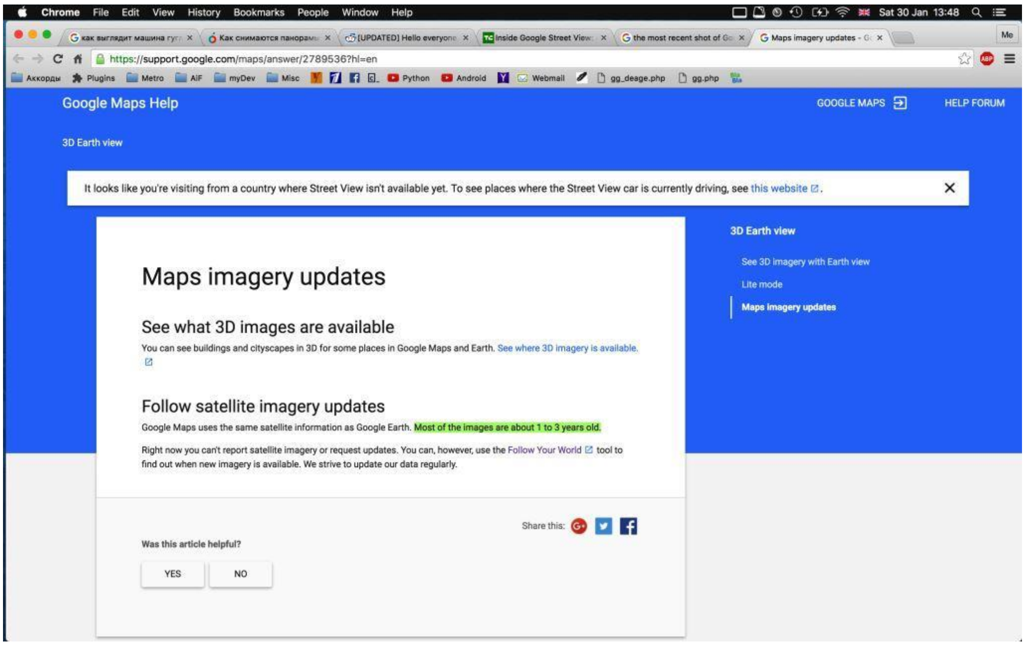
4. However, some local gas providers do have their prices published (see the excerpt below from Lukoil).



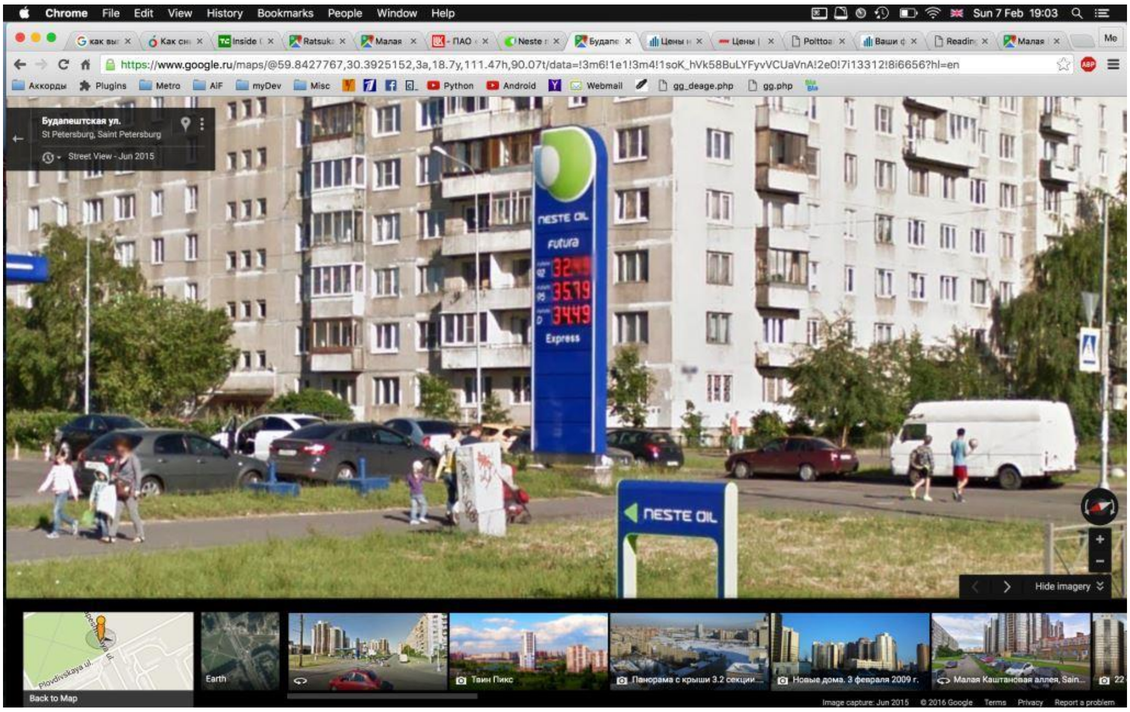
Compare the screenshot date and time with the price lists available at request time to see they are outdated as of 7 February 2016.

Appendix 3. Visual comparisons of test apps output

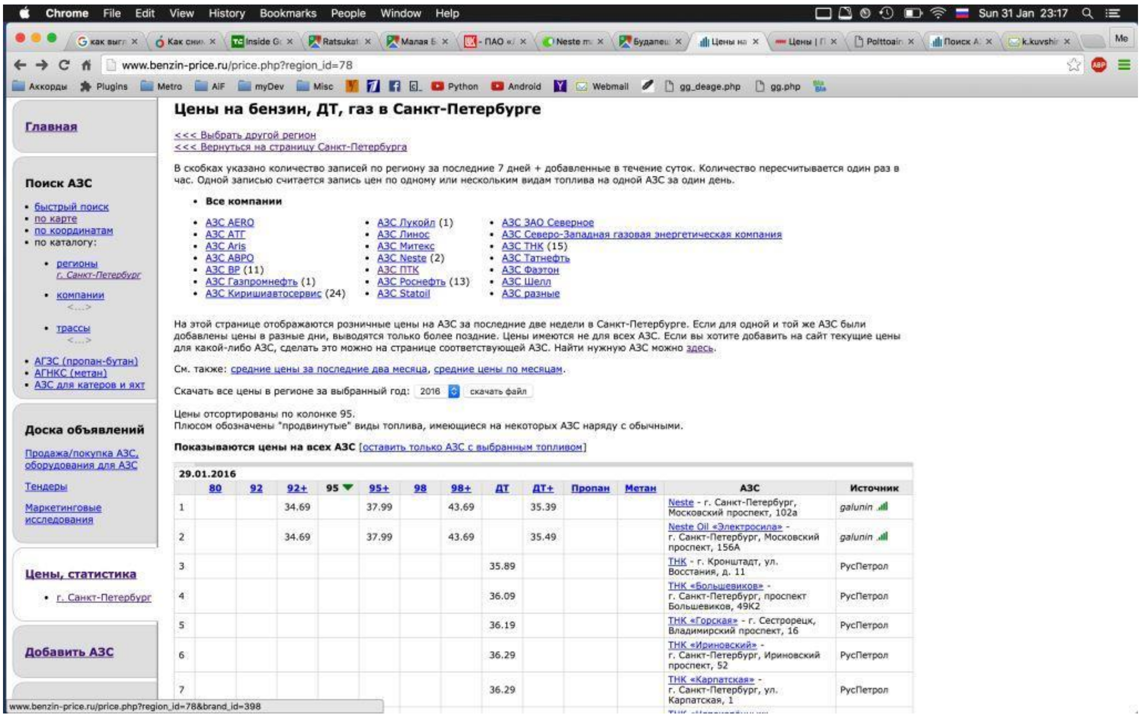
1. Google Maps image policy allows for 1-3 years of “stale” imagery.

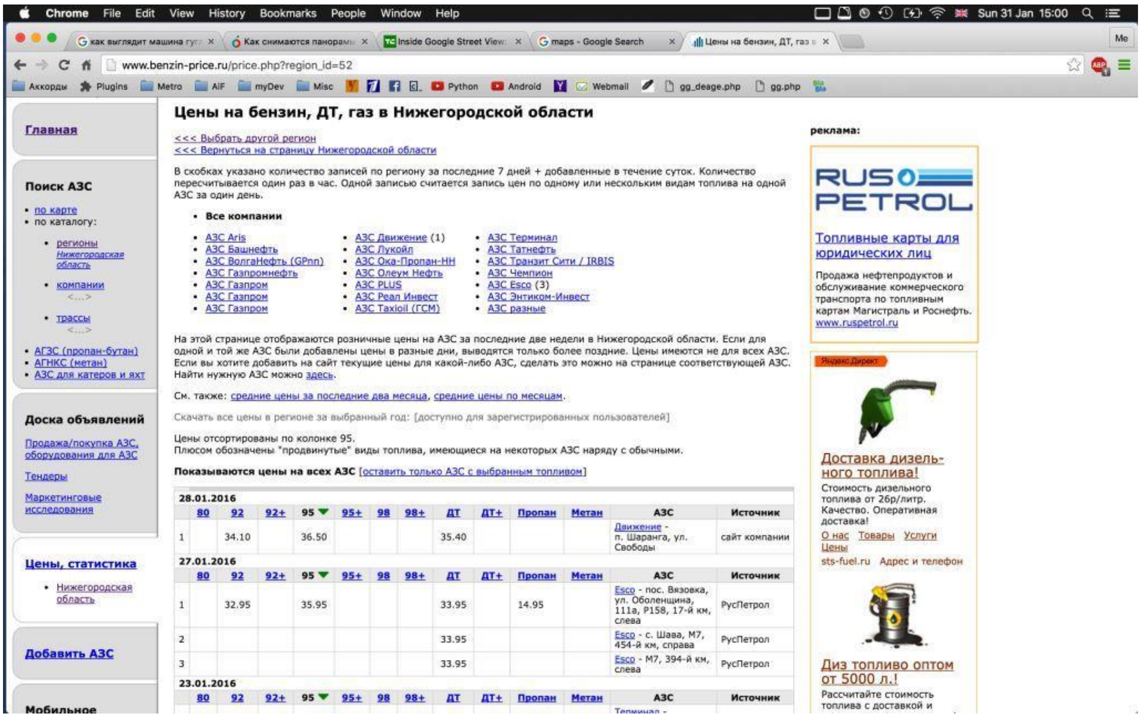


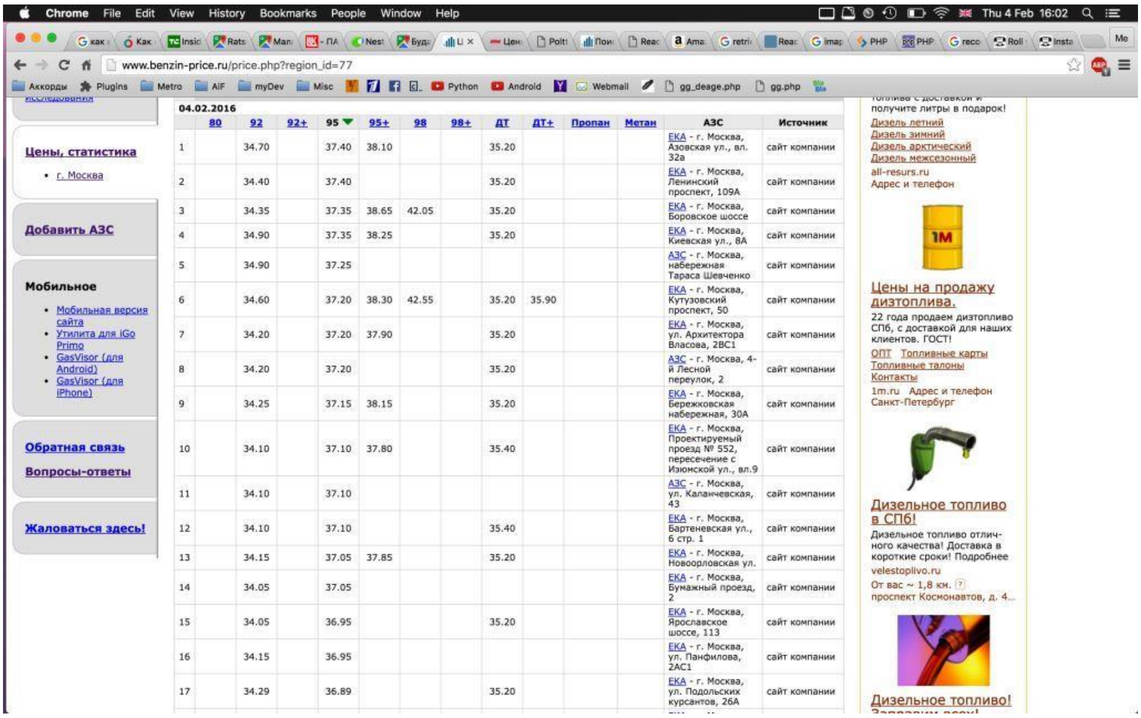
Indeed, a 2016 shot below reveals that the image capture occurred a year ago (see the right bottom part vs the right top one).



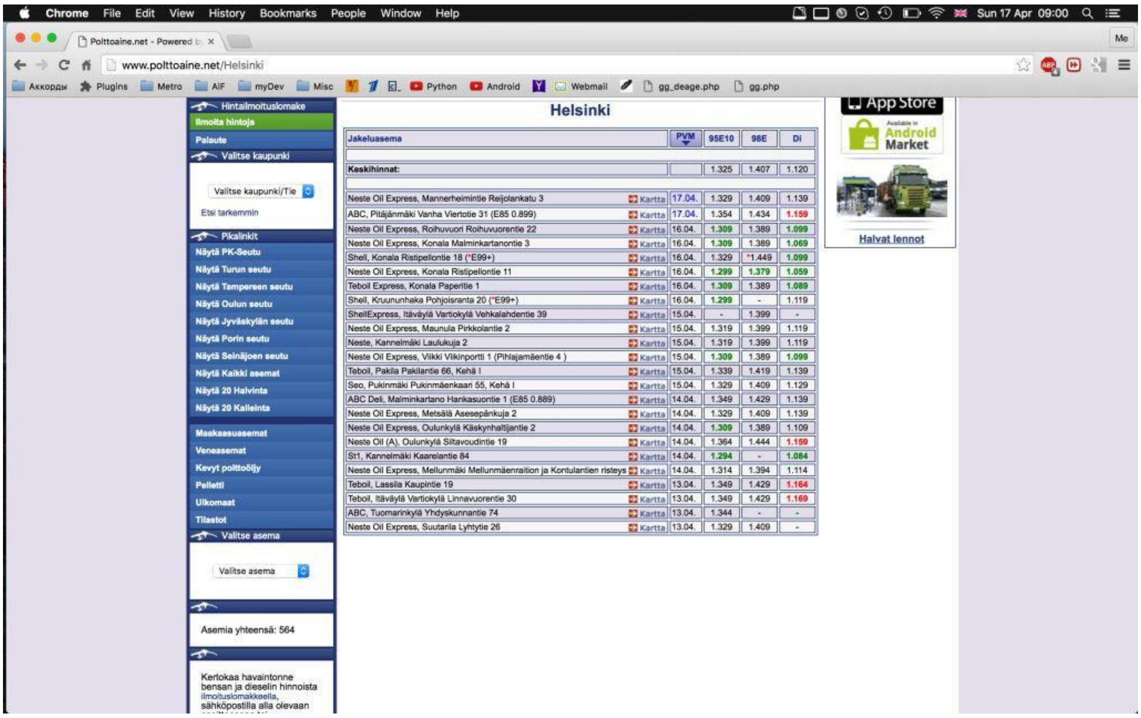
2. Moscow and Nizhniy Novgorod reinforced St Petersburg as test locations to see their overall updatability. *BenzinPrice* seldom showed actual prices at request time. Find below, in the three subsequent images, examples for St Petersburg (digits 78 in the link), Nizhniy Novgorod (52), and Moscow (77), these numbers representing the traffic police area codes. With the shooting dates and time compared to the tabled data, chronological gaps become apparent. More (screen)shots are available on request.

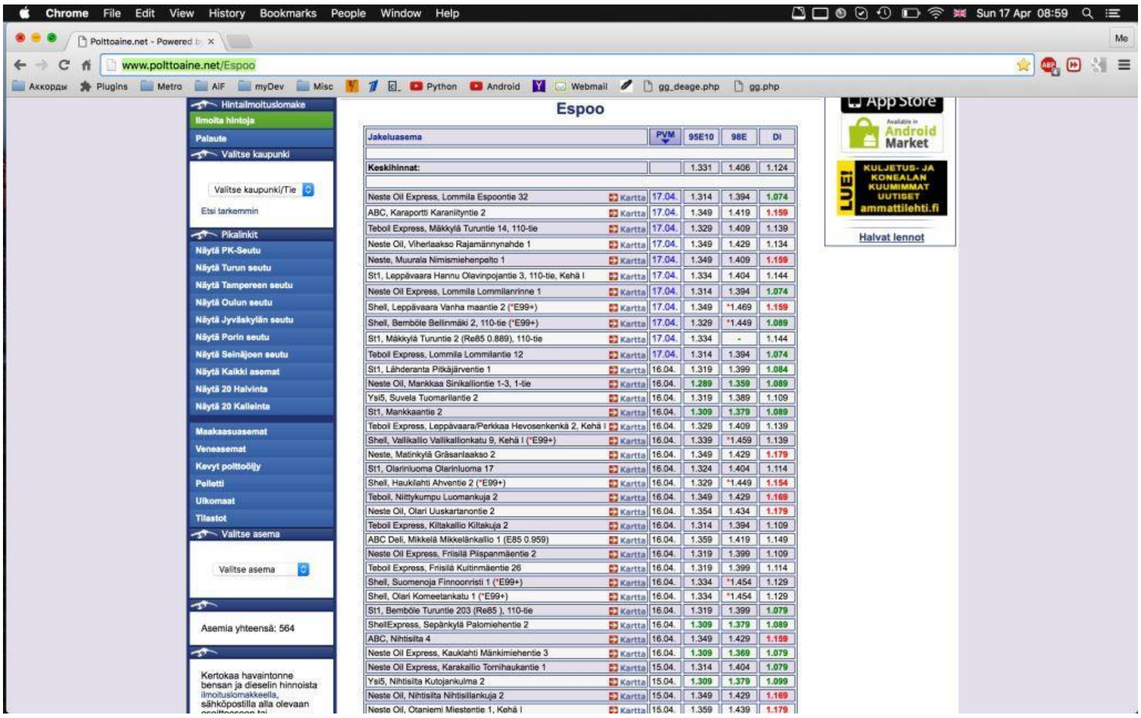






3. Unlike the other candidate tools, Polttoaine usually showed updates already in the morning (see Helsinki and Espoo as the most updated locations in Finland, in the two images below).





Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.331	1.406	1.124
Neste Oil Express, Lommila Espoontie 32	Kartta	17.04.	1.314	1.394	1.074
ABC, Karaportti Karaniityntie 2	Kartta	17.04.	1.349	1.419	1.159
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	17.04.	1.329	1.409	1.139
Neste Oil, Viherlaakso Rajamännynahde 1	Kartta	17.04.	1.349	1.429	1.134
Neste, Muurala Nimismiehenpelto 1	Kartta	17.04.	1.349	1.409	1.159
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	17.04.	1.334	1.404	1.144
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	17.04.	1.314	1.394	1.074
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	17.04.	1.349	*1.469	1.159
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	17.04.	1.329	*1.449	1.089
St1, Mäkkylä Turuntie 2 (Re85 0.889), 110-tie	Kartta	17.04.	1.334	-	1.144
Teboil Express, Lommila Lommilantie 12	Kartta	17.04.	1.314	1.394	1.074
St1, Lähderanta Pitkäjärventie 1	Kartta	16.04.	1.319	1.399	1.084
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	16.04.	1.289	1.359	1.089
Ysi5, Suvela Tuomarilantie 2	Kartta	16.04.	1.319	1.389	1.109
St1, Mankkaantie 2	Kartta	16.04.	1.309	1.379	1.089
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	16.04.	1.329	1.409	1.139
Shell, Vallikallio Vallikallionkatu 9, Kehä I (*E99+)	Kartta	16.04.	1.339	*1.459	1.139
Neste, Matinkylä Gräsanlaakso 2	Kartta	16.04.	1.349	1.429	1.179
St1, Olariniuoma Olarinluoma 17	Kartta	16.04.	1.324	1.404	1.114
Shell, Haukilahti Ahventie 2 (*E99+)	Kartta	16.04.	1.329	*1.449	1.154
Teboil, Niittykumpu Luomankuja 2	Kartta	16.04.	1.349	1.429	1.169
Neste Oil, Olari Uuskartanontie 2	Kartta	16.04.	1.354	1.434	1.179
Teboil Express, Kiltakallio Kiltakuja 2	Kartta	16.04.	1.314	1.394	1.109
ABC Deli, Mikkelä Mikkelänkallio 1 (E85 0.959)	Kartta	16.04.	1.359	1.419	1.149
Neste Oil Express, Friisilä Piispanmäentie 2	Kartta	16.04.	1.319	1.399	1.109
Teboil Express, Friisilä Kuitinmäentie 26	Kartta	16.04.	1.319	1.399	1.114
Shell, Suomenoja Finnoonristi 1 (*E99+)	Kartta	16.04.	1.334	*1.454	1.129
Shell, Olari Komeetankatu 1 (*E99+)	Kartta	16.04.	1.334	*1.454	1.129
St1, Bemböle Turuntie 203 (Re85), 110-tie	Kartta	16.04.	1.319	1.399	1.079
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	16.04.	1.309	1.379	1.089
ABC, Nihtisilta 4	Kartta	16.04.	1.349	1.429	1.159
Neste Oil Express, Kauklahti Mänkimiehentie 3	Kartta	16.04.	1.309	1.369	1.079
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	15.04.	1.314	1.404	1.079
Ysi5, Nihtisilta Kutojankulma 2	Kartta	15.04.	1.309	1.379	1.099
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	15.04.	1.349	1.429	1.169
Neste Oil, Otaniemi Miestentie 1, Kehä I	Kartta	15.04.	1.359	1.439	1.179

Compare the screenshot date and time of 2016 in the upper right-hand corner with the tabled data. Only morning shots are used to show already the difference vs other apps and tools. More screenshots for the period of January-April 2016 are available on request.

(Rest of space left blank intentionally)

4. As the research continued, it became obvious that the tabled results (exampled in the two tables below) confirmed the idea that Polttoaine was the most updated app among the other candidates. Note the digits under the parts of the day; they explain how frequent the updates were during the observation day.

Date	City	Morning	Midday vs day	Evening/night	Time
30.1.16	Moscow	0		0	11:59; 21:15.
30.1.16	N.Novgorod		0	0	12:01; 21:16.
30.1.16	St.Pete	0	0	0	11:59; 21:14.
31.1.16	Moscow		0	8	15:00; 23:18.
31.1.16	N.Novgorod		0	0	15:00; 23:19.
31.1.16	St.Pete		0	0	14:58; 23:17.
1.2.16	Moscow			2	23:01.
1.2.16	N.Novgorod			0	23:02.
1.2.16	St.Pete			1	23:00.
2.2.16	Moscow		0	7	12:35; 21:03.
2.2.16	N.Novgorod		0	0	12:37; 21:04.
2.2.16	St.Pete		0	1	12:36; 21:04.
3.2.16	Moscow		16	16	15:45; 21:29.
3.2.16	N.Novgorod		13	13	15:46; 21:30.
3.2.16	St.Pete		1	1	15:46; 21:31.
4.2.16	Moscow	0	77	80	09:48; 16:02; 23:38.
4.2.16	N.Novgorod	0	0	1	09:49; 16:05; 23:38.
4.2.16	St.Pete	0	1	1	09:50; 16:06; 23:39.
7.4.16	Moscow			1	
7.4.16	N.Novgorod			0	
7.4.16	St.Pete			2	
9.4.16	Moscow			1	
9.4.16	N.Novgorod			0	
9.4.16	St.Pete			0	
10.4.16	Moscow	0	0	5	
10.4.16	N.Novgorod	0	0	0	
10.4.16	St.Pete	0	0	0	
11.4.16	Moscow	0	0	0	
11.4.16	N.Novgorod	0	0	0	
11.4.16	St.Pete	0	0	0	
12.4.16	Moscow	1	1 vs 2	2	10:48; 12:11 15:28; 22:22.
12.4.16	N.Novgorod	0	0 vs 0	0	10:50; 12:12 15:30; 22:23.
12.4.16	St.Pete	0	2 vs 2	2	10:46; 12:07 15:25; 22:20.
13.4.16	Moscow	0	71	71	09:14; 13:27; 21:10.
13.4.16	N.Novgorod	0	0	2	09:15; 13:25; 21:10.
13.4.16	St.Pete	0	0 vs 3	25	09:13; 13:29 13:38; 21:08.
14.4.16	Moscow	0	0	16	09:03; 12:05 ; 21:11.
14.4.16	N.Novgorod	0	0	0	09:04; 12:07; 21:12.
14.4.16	St.Pete	0	1	+ 2 by Waze	09:03; 12:04; 21:10 21:47
15.4.16	Moscow	0	0	0	07:27; 14:04; 21:29.
15.4.16	N.Novgorod	0	0	0	07:30; 14:05; 21:30.
15.4.16	St.Pete	0	2	2	07:26; 14:04; 21:29.
16.4.16	Moscow	0	0	0	11:15; 16: 22; 21:28.
16.4.16	N.Novgorod	0	0	0	11:16; 16:23; 21:29.
16.4.16	St.Pete	0 +2 by W	ditto	ditto	11:12 11:28; 16:21 16:34
17.4.16	Moscow	1	1	8	08:56; 12:06; 22:43.
17.4.16	N.Novgorod	0	0	0	08:56; 12:07; 22:47.
17.4.16	St.Pete	0 +2 by W	0 +2 by Waze	0 +3 by Waze	08:57 09:09; 12:06 12:16;
18.4.16	Moscow		0 vs 0	19	12:45 14:51; 22:33.
18.4.16	N.Novgorod		0 vs 0	0	12:46 14:53; 22:34.
18.4.16	St.Pete		0 +3 by Waze	1	12:44 12:59; 14:50 14:57;
19.4.16	Moscow	0	0		09:23; 13:02
19.4.16	N.Novgorod	0	0		09:24; 13:02
19.4.16	St.Pete	0	0		09:22; 13:00

Date	City	Morning	Midday vs day	Evening/night	Time
30.1.16	Espoo	1		39	11:50; 21:23.
30.1.16	Helsinki	1		13	11:52; 21:22.
31.1.16	Espoo		13	14	15:36; 23:23.
31.1.16	Helsinki		20	24	15:38; 23:22.
1.2.16	Helsinki			15	23:07.
1.2.16	Espoo			15	23:08.
2.2.16	Helsinki		7	11	12:43; 21:07.
2.2.16	Espoo		2	15	12:43; 21:07.
3.2.16	Helsinki		0	29	15:49; 21:33.
3.2.16	Espoo		1	17	15:49; 21:34.
4.2.16	Espoo	2	7	18	09:54; 16:08; 23:36.
4.2.16	Helsinki	7	12	22	09:53; 16:08; 23:36.
7.4.16	Espoo		4	8	
7.4.16	Helsinki		5	11	
9.4.16	Espoo		0	43	
9.4.16	Helsinki		0	7	
10.4.16	Espoo		0	14	
10.4.16	Helsinki		0	19	
11.4.16	Espoo	4	4	12	
11.4.16	Helsinki	8	9	20	
12.4.16	Espoo	1	1 vs 2	15	10:42; 12:04 15:11; 2
12.4.16	Helsinki	1	3 vs 3	7	10:43; 12:05 15:24; 2
13.4.16	Espoo	0	3	15	09:10; 13:30; 21:05.
13.4.16	Helsinki	0	1	5	09:12; 13:32; 21:07.
14.4.16	Espoo	2	2	14	09:59; 12:01; 21:13.
14.4.16	Helsinki	1	3	10	09:59; 12:03; 21:15.
15.4.16	Espoo	0	9	18	07:30; 14:03; 21:26.
15.4.16	Helsinki	0	2	8	07:31; 14:01;
16.4.16	Espoo	0	27	30	11:17; 16:19; 21:25.
16.4.16	Helsinki	0	0		11:18; 16:20; 21:26.
17.4.16	Espoo	11	12	19	08:59; 12:04; 22:38.
17.4.16	Helsinki	2	6	17	09:00; 12:05; 22:39.
18.4.16	Espoo		6 vs 6	12	12:47 14:49; 22:35.
18.4.16	Helsinki		5 vs 5	17	12:48 14:48; 22:36.
19.4.16	Espoo	3	4		09:25; 13:03
19.4.16	Helsinki	0	0		09:26; 13:04

Clearly enough, the tables suggest that Polttoaine revealed better regularity of updates within the pre-study period.

Appendix 4. Polttoaine's fuel display policy visualised

According to their policy, or their Info tab, "Hintatiedot pysyvät näkyvillä 5 vuorokautta ilman uutta päivitystä" (link given in 2.4's footnotes), the website quotes prices for the last 5 days. When displayed on a single page, historical data tend to disappear as updates arrive. In order to see clearly how they work, a series of Espoo screenshots[13] follow to show the process chronologically, within a sample two-day span limit:

Espoo

Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.482	1.555	1.349
Neste Oil Express, Lommila Espoontie 32	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	22.02.	1.429	1.509	1.299
Teboil Express, Lommila Lommilantie 12	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	21.02.	1.454	1.534	1.309
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	21.02.	1.449	1.519	1.299
Teboil, Kivenlahti Vanha Jorvaksentie 1	Kartta	21.02.	1.519	1.599	1.369
Teboil, Niittykumpu Luomankuja 2	Kartta	21.02.	1.519	1.599	1.369
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	21.02.	1.449	1.519	1.319
ABC, Nihtisilta 4	Kartta	21.02.	1.469	1.539	1.309
St1, Karamalmi - KII Karapellontie 11 (RE85 0.899)	Kartta	21.02.	1.464	-	1.319
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	21.02.	1.514	1.594	1.399
St1, Lähderanta Pitkäjärventie 1	Kartta	21.02.	1.464	1.544	1.319
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	21.02.	1.484	1.564	1.339
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	21.02.	1.504	1.584	1.359
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	21.02.	1.479	1.559	1.334
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	21.02.	1.499	*1.629	1.354
ABC, Juvankartanontie 33 (E85)	Kartta	21.02.	1.439	1.519	1.309
St1, Bemböle Turuntie 203 (Re85 0.919), 110-tie	Kartta	21.02.	1.469	1.549	1.319
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	21.02.	1.479	*1.599	1.329
Neste Oil, Viherlaakso Rajamännynahde 1	Kartta	20.02.	1.519	1.599	1.384
Ysi5, Suvela Tuomarilantie 2	Kartta	20.02.	1.479	1.559	1.329
St1, Mäkkylä Turuntie 2 (Re85 0.999), 110-tie	Kartta	20.02.	1.514	-	1.369
Ysi5, Nihtisilta Kutojankulma 2	Kartta	20.02.	1.469	1.539	1.319
ABC, Karaportti Karaniityntie 2	Kartta	20.02.	1.509	1.579	1.364
ABC, Kivenlahti Kivenlahdentie 7	Kartta	20.02.	1.459	1.539	1.349

Figure 33. A random day (this Feb 22-23), midday

Espoo

Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.482	1.554	1.349
Neste Oil, Otaniemi Miestentie 1, Kehä I	Kartta	22.02.	1.449	1.499	1.359
ABC, Otaniemi Miestentie 1	Kartta	22.02.	1.469	1.539	1.369
St1, Mäkkylä Turuntie 2 (Re85 0.999), 110-tie	Kartta	22.02.	1.514	-	1.364
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	22.02.	1.479	1.559	1.334
Shell, Vallikallio Vallikallionkatu 9, Kehä I (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	22.02.	1.504	1.584	1.359
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	22.02.	1.474	1.554	1.329
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Lommila Espoontie 32	Kartta	22.02.	1.429	1.509	1.299
Teboil Express, Lommila Lommilantie 12	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	21.02.	1.454	1.534	1.309
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	21.02.	1.449	1.519	1.299
Teboil, Niittykumpu Luomankuja 2	Kartta	21.02.	1.519	1.599	1.369
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	21.02.	1.449	1.519	1.319
Teboil, Kivenlahti Vanha Jorvaksentie 1	Kartta	21.02.	1.519	1.599	1.369
St1, Karamalmi - KII Karapellontie 11 (RE85 0.899)	Kartta	21.02.	1.464	-	1.319
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	21.02.	1.514	1.594	1.399
St1, Lähderanta Pitkäjärventie 1	Kartta	21.02.	1.464	1.544	1.319
ABC, Nihtisilta 4	Kartta	21.02.	1.469	1.539	1.309
ABC, Juvankartanontie 33 (E85)	Kartta	21.02.	1.439	1.519	1.309
St1, Bemböle Turuntie 203 (Re85 0.919), 110-tie	Kartta	21.02.	1.469	1.549	1.319
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	21.02.	1.479	*1.599	1.329
Neste Oil, Viherlaakso Rajamännynahde 1	Kartta	20.02.	1.519	1.599	1.384

Figure 34. Same day's evening

Espoo

Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.473	1.546	1.336
Ysi5, Suvela Tuomarilantie 2	Kartta	23.02.	1.429	1.509	1.309
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	22.02.	1.439	1.519	1.279
ABC, Nihtisilta 4	Kartta	22.02.	1.469	1.539	1.309
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	22.02.	1.469	1.549	1.299
ABC, Karaportti Karaniityntie 2	Kartta	22.02.	1.509	1.579	1.359
ABC, Otaniemi Miestentie 1	Kartta	22.02.	1.469	1.539	1.369
Neste Oil, Otaniemi Miestentie 1, Kehä I	Kartta	22.02.	1.449	1.499	1.359
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	22.02.	1.474	1.554	1.329
Shell, Vallikallio Vallikallionkatu 9, Kehä I (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
St1, Mäkkylä Turuntie 2 (Re85 0.999), 110-tie	Kartta	22.02.	1.514	-	1.364
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	22.02.	1.504	1.584	1.359
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	22.02.	1.479	1.559	1.334
Neste Oil Express, Lommila Espoontie 32	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	22.02.	1.429	1.509	1.299
Teboil Express, Lommila Lommilantie 12	Kartta	22.02.	1.429	1.509	1.299
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	21.02.	1.449	1.519	1.319
Teboil, Kivenlahti Vanha Jorvaksentie 1	Kartta	21.02.	1.519	1.599	1.369
Teboil, Niittykumpu Luomankuja 2	Kartta	21.02.	1.519	1.599	1.369
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	21.02.	1.514	1.594	1.399
St1, Karamalmi - KII Karapellontie 11 (RE85 0.899)	Kartta	21.02.	1.464	-	1.319
St1, Lähderanta Pitkäjärventie 1	Kartta	21.02.	1.464	1.544	1.319
ABC, Juvankartanontie 33 (E85)	Kartta	21.02.	1.439	1.519	1.309
St1, Bemböle Turuntie 203 (Re85 0.919), 110-tie	Kartta	21.02.	1.469	1.549	1.319
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	21.02.	1.479	*1.599	1.329

Figure 35. This Feb 23 afternoon

Espoo

Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.472	1.545	1.333
ABC, Nihtisilta 4	Kartta	23.02.	1.489	1.559	1.309
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	23.02.	1.449	1.519	1.319
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	23.02.	1.459	1.539	1.289
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	23.02.	1.469	1.549	1.319
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	23.02.	1.514	1.594	1.389
St1, Lähderanta Pitkäjärventie 1	Kartta	23.02.	1.469	1.549	1.299
St1, Karamalmi - KII Karapellontie 11 (RE85 0.899)	Kartta	23.02.	1.469	-	1.299
ABC, Karaportti Karaniityntie 2	Kartta	23.02.	1.504	1.574	1.354
ABC, Kivenlahti Kivenlahdentie 7	Kartta	23.02.	1.489	1.569	1.339
Neste Oil, Espoonlahti Kivenlahdentie 5	Kartta	23.02.	1.429	1.509	1.339
Neste Oil Express, Kivenlahti Maininkitie 11	Kartta	23.02.	1.439	1.509	1.309
Teboil, Espoonlahti Kivenlahdentie 2	Kartta	23.02.	1.439	1.519	1.319
St1, Mäkkylä Turuntie 2 (Re85 0.999), 110-tie	Kartta	23.02.	1.499	-	1.349
Ysi5, Suvela Tuomarilantie 2	Kartta	23.02.	1.429	1.509	1.309
ABC, Otaniemi Miestentie 1	Kartta	22.02.	1.469	1.539	1.369
Neste Oil, Otaniemi Miestentie 1, Kehä I	Kartta	22.02.	1.449	1.499	1.359
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	22.02.	1.504	1.584	1.359
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	22.02.	1.474	1.554	1.329
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	22.02.	1.479	1.559	1.334
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Shell, Vallikallio Vallikallionkatu 9, Kehä I (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Neste Oil Express, Lommila Espoontie 32	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	22.02.	1.429	1.509	1.299
Teboil Express, Lommila Lommilantie 12	Kartta	22.02.	1.429	1.509	1.299
Teboil, Kivenlahti Vanha Jorvaksentie 1	Kartta	21.02.	1.519	1.599	1.369
Teboil, Niittykumpu Luomankuja 2	Kartta	21.02.	1.519	1.599	1.369
ABC, Juvankartanontie 33 (E85)	Kartta	21.02.	1.439	1.519	1.309
St1, Bemböle Turuntie 203 (Re85 0.919), 110-tie	Kartta	21.02.	1.469	1.549	1.319
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	21.02.	1.479	*1.599	1.329
Neste Oil, Viherlaakso Rajamännynahde 1	Kartta	20.02.	1.519	1.599	1.384
Ysi5, Nihtisilta Kutojankulma 2	Kartta	20.02.	1.469	1.539	1.319
ABC Deli, Tapiola Kalevalantie 3 (E85 0.959), Kehä I	Kartta	19.02.	1.479	1.549	1.379

Figure 36. Same day's 17:30 (local time)

[13] As taken from http://www.polttoaine.net/Espoo.

Usually, the page has a limited number of lines. As updates appear on top of the list (blue-, green- and orange-bordered rectangles in Figures 33-38), the previous prices tend to disappear. In addition, it may also occur that all of today's prices take up the entirety of the page, thus leaving no space for historical data.

Espoo

Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.471	1.544	1.331
Neste Oil, Viherlaakso Rajamännynahde 1	Kartta	23.02.	1.499	1.579	1.369
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	23.02.	1.479	1.559	1.334
ABC, Karaportti Karaniityntie 2	Kartta	23.02.	1.504	1.574	1.354
St1, Karamalmi - KII Karapellontie 11 (RE85 0.899)	Kartta	23.02.	1.469	-	1.299
St1, Lähderanta Pitkäjärventie 1	Kartta	23.02.	1.469	1.549	1.299
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	23.02.	1.514	1.594	1.389
ABC, Nihtisilta 4	Kartta	23.02.	1.489	1.559	1.309
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	23.02.	1.459	1.539	1.289
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	23.02.	1.449	1.519	1.319
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	23.02.	1.469	1.549	1.319
ABC, Kivenlahti Kivenlahdentie 7	Kartta	23.02.	1.489	1.569	1.339
Teboil, Espoonlahti Kivenlahdentie 2	Kartta	23.02.	1.439	1.519	1.319
Neste Oil Express, Kivenlahti Maininkitie 11	Kartta	23.02.	1.439	1.509	1.309
Neste Oil, Espoonlahti Kivenlahdentie 5	Kartta	23.02.	1.429	1.509	1.339
St1, Mäkkylä Turuntie 2 (Re85 0.999), 110-tie	Kartta	23.02.	1.499	-	1.349
Ysi5, Suvela Tuomarilantie 2	Kartta	23.02.	1.429	1.509	1.309
ABC, Otaniemi Miestentie 1	Kartta	22.02.	1.469	1.539	1.369
Neste Oil, Otaniemi Miestentie 1, Kehä I	Kartta	22.02.	1.449	1.499	1.359
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	22.02.	1.504	1.584	1.359
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	22.02.	1.474	1.554	1.329
Shell, Vallikallio Vallikallionkatu 9, Kehä I (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Neste Oil Express, Lommila Espoontie 32	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	22.02.	1.429	1.509	1.299
Teboil Express, Lommila Lommilantie 12	Kartta	22.02.	1.429	1.509	1.299
Teboil, Niittykumpu Luomankuja 2	Kartta	21.02.	1.519	1.599	1.369
Teboil, Kivenlahti Vanha Jorvaksentie 1	Kartta	21.02.	1.519	1.599	1.369
ABC, Juvankartanontie 33 (E85)	Kartta	21.02.	1.439	1.519	1.309
St1, Bemböle Turuntie 203 (Re85 0.919), 110-tie	Kartta	21.02.	1.469	1.549	1.319
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	21.02.	1.479	*1.599	1.329
Ysi5, Nihtisilta Kutojankulma 2	Kartta	20.02.	1.469	1.539	1.319

Figure 37. This Feb 23's 22:59

(remains the same at 04:00, 05:00, 08:00 Feb 24)

Espoo

Jakeluasema		PVM	95E10	98E	Di
Keskihinnat:			1.471	1.546	1.331
ABC, Nihtisilta 4	Kartta	24.02.	1.489	1.579	1.309
Neste Oil, Nihtisilta Nihtisillankuja 2	Kartta	24.02.	1.514	1.594	1.389
Ysi5, Nihtisilta Kutojankulma 2	Kartta	24.02.	1.459	1.539	1.319
ShellExpress, Sepänkylä Palomiehentie 2	Kartta	24.02.	1.469	1.549	1.299
Teboil, Niittykumpu Luomankuja 2	Kartta	24.02.	1.519	1.599	1.369
Ysi5, Suvela Tuomarilantie 2	Kartta	24.02.	1.459	1.539	1.319
Teboil Express, Mäkkylä Turuntie 14, 110-tie	Kartta	24.02.	1.494	1.574	1.359
Teboil, Kivenlahti Vanha Jorvaksentie 1	Kartta	24.02.	1.519	1.599	1.369
St1, Mäkkylä Turuntie 2 (Re85 0.999), 110-tie	Kartta	24.02.	1.499	-	1.364
ABC, Karaportti Karaniityntie 2	Kartta	24.02.	1.504	1.584	1.354
Neste Oil, Viherlaakso Rajamännynahde 1	Kartta	23.02.	1.499	1.579	1.369
St1, Leppävaara Hannu Olavinpojantie 3, 110-tie, Kehä I	Kartta	23.02.	1.479	1.559	1.334
St1, Karamalmi - KII Karapellontie 11 (RE85 0.899)	Kartta	23.02.	1.469	-	1.299
Neste Oil, Mankkaa Sinikalliontie 1-3, 1-tie	Kartta	23.02.	1.449	1.519	1.319
Neste Oil Express, Karakallio Tornihaukantie 1	Kartta	23.02.	1.459	1.539	1.289
St1, Lähderanta Pitkäjärventie 1	Kartta	23.02.	1.469	1.549	1.299
ABC, Kivenlahti Kivenlahdentie 7	Kartta	23.02.	1.489	1.569	1.339
Neste Oil Express, Kivenlahti Maininkitie 11	Kartta	23.02.	1.439	1.509	1.309
Neste Oil, Espoonlahti Kivenlahdentie 5	Kartta	23.02.	1.429	1.509	1.339
Teboil, Espoonlahti Kivenlahdentie 2	Kartta	23.02.	1.439	1.519	1.319
ABC, Otaniemi Miestentie 1	Kartta	22.02.	1.469	1.539	1.369
Neste Oil, Otaniemi Miestentie 1, Kehä I	Kartta	22.02.	1.449	1.499	1.359
Shell, Leppävaara Vanha maantie 2 (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Shell, Vallikallio Vallikallionkatu 9, Kehä I (*E99+)	Kartta	22.02.	1.494	*1.624	1.349
Teboil Express, Leppävaara/Perkkaa Hevosenkenkä 2, Kehä I	Kartta	22.02.	1.474	1.554	1.329
Neste Oil Express, Lommila Lommilanrinne 1	Kartta	22.02.	1.429	1.509	1.299
Neste Oil Express, Lommila Espoontie 32	Kartta	22.02.	1.429	1.509	1.299
Teboil Express, Lommila Lommilantie 12	Kartta	22.02.	1.429	1.509	1.299
ABC, Juvankartanontie 33 (E85)	Kartta	21.02.	1.439	1.519	1.309
St1, Bemböle Turuntie 203 (Re85 0.919), 110-tie	Kartta	21.02.	1.469	1.549	1.319
Shell, Bemböle Bellinmäki 2, 110-tie (*E99+)	Kartta	21.02.	1.479	*1.599	1.329

Figure 38. This Feb 24's 16:30 (10 updates)

Other observations include that:

- Usually, today's, yesterday's and, at times, earlier prices fill up the entire page;
- 30-min or hourly updates may start at 07:00 or halt for a part of the day.

With these observations, came the understanding that the usual price span was between today, yesterday and earlier prices, which later formed the three-day approach for own database structure. For the implementation details, refer to the *Admin app* subsection of 3.1.

Appendix 5. User and Admin apps: use cases

User app

A simplified UML scheme and a flowchart representation of the User app in action are shown in Figures 39,40.

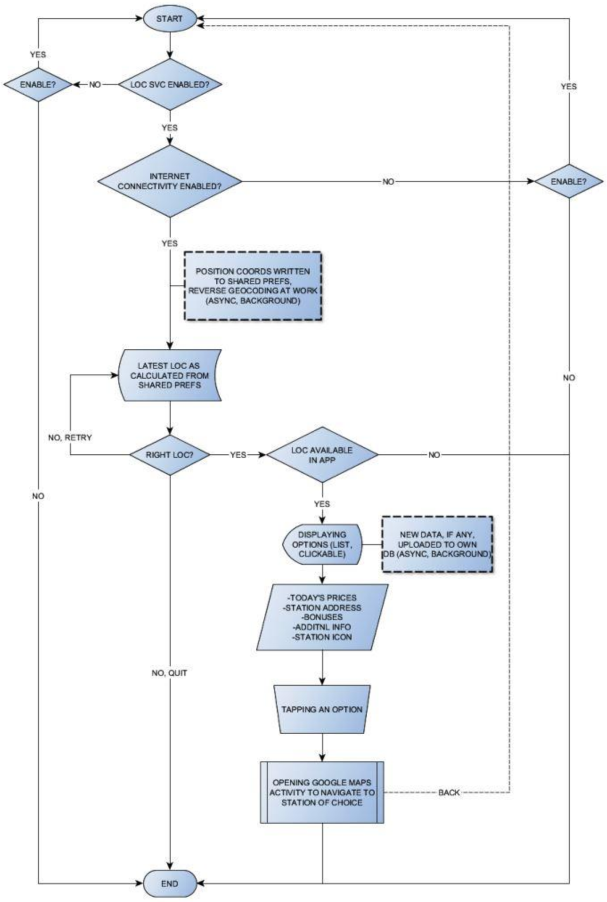


Figure 39. Flowchart of User app in action

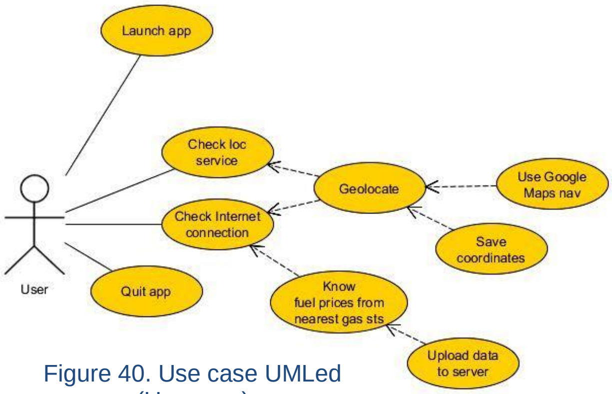


Figure 40. Use case UMLed (User app)

Admin app

Similarly, a simple UML scheme and a flowchart representation of the Admin app in action are given in Figures 41,42.

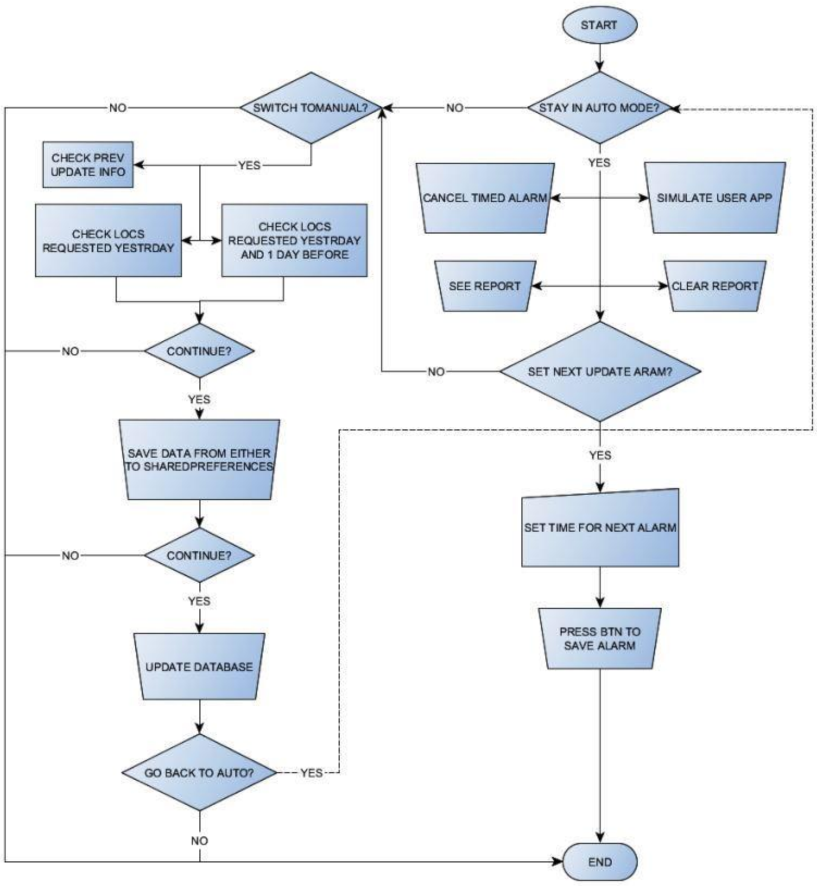


Figure 41. Flowchart of Admin app in action

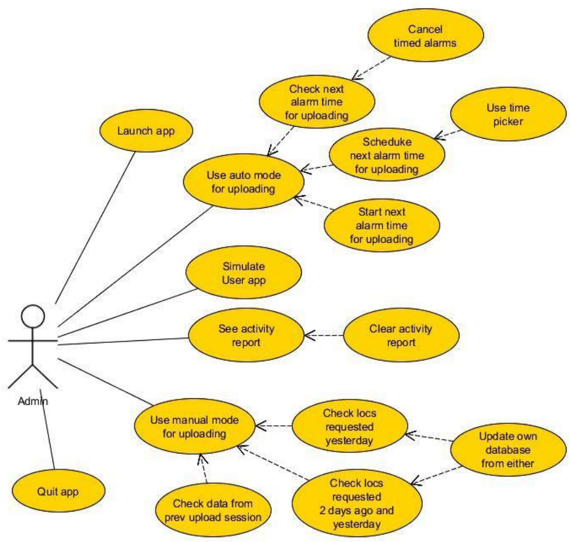


Figure 42. Use case UMLed (Admin app)

The Admin app is related to own database, so it is important to show its structure to understand the simple interactions there (see Figure 43 below).

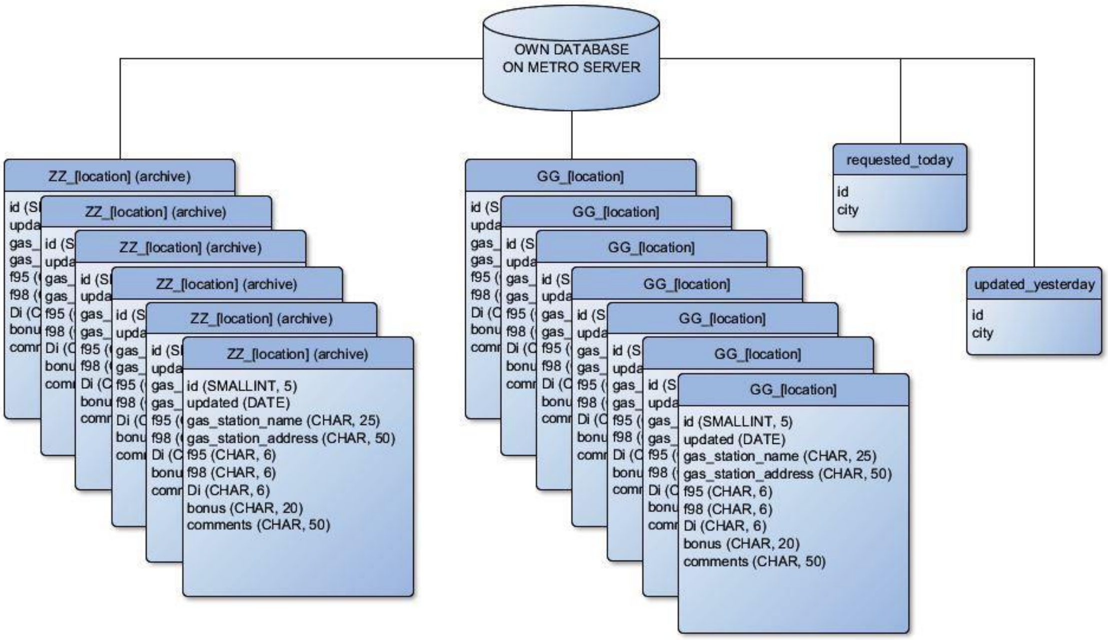


Figure 43. Simple, unrelated DB scheme of the project, with the three-day approach in mind

END OF DOCUMENT