

UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Arkadi Statsenko

**State registry data access notifications for
Estonian eID holders**

Bachelor's Thesis (9 ECTS)

Supervisor:
Daniel Würsch, MSc

Tartu 2025

State registry data access notifications for Estonian eID holders

Abstract:

Andmejälgija is a protocol developed by Information System Authority (RIA), the purpose of which is to provide a uniform interface for querying Estonian residents' data access logs. There is also an *Andmejälgija* web-view accessible from the state-portal *eesti.ee*.

The purpose of this thesis is to create a mobile application that would notify its users of updates in the access logs, letting them know that their data in some state database has been accessed. Implementation choices of different aspects of the solution are also going to be covered together with advantages and disadvantages of each. Additionally, the overview of the existing state databases will be provided, including whether they provide access logs or not.

CERCS: T180 Telecommunication engineering

Riiklike andmekogude andmete juurdepääsu teavitused Eesti e-ID omanikele

Lühikokkuvõte:

Käesoleva lõputöö eesmärk on luua rakendus, mis teavitaks kasutajaid juurdepääsulogide uuendustest, andes neile teada, et nende andmeid mõnes riigi andmebaasis on kasutatud. Rakendusel on olnud erinevad rakendusvariandid, mida käsitletakse ka koos igäühe eeliste ja puudustega. Lisaks antakse ülevaade olemasolevatest riiklikest andmebaasidest, sealhulgas sellest, kas nad pakuvad juurdepääsulogisid või mitte.

CERCS: T180 Telekommunikatsioonitehnoloogia

Contents

1. Introduction	5
2. Andmejälgija	6
2.1 The protocol	6
2.2 Usage	7
2.3 Adoption	7
3. Overview of state databases	10
3.1 Databases implementing <i>Andmejälgija</i>	10
3.1.1 Examples of good implementation	10
3.1.2 Other databases implementing <i>Andmejälgija</i>	10
3.2 Databases providing other form of data access tracking	11
3.2.1 E-File System (E-toimik)	11
3.3 Databases not providing any kind of data access tracking	13
4. Implementation	14
4.1 Description	14
4.2 Discussion of implementation choice	14
4.2.1 X-Road service	14
4.2.2 Standalone approach	15
4.3 Development	15
4.3.1 Authentication	17
4.3.2 Interacting with <i>eesti.ee</i> internal APIs	18
4.3.3 Keeping the session alive	19
4.3.4 WorkManager approach	20
4.3.5 Foreground service approach	21
4.3.6 Hybrid AlarmManager approach	21
4.3.7 Session limitations	22
4.3.8 Handling logs and notification delivery	24
4.4 Software distribution	25
4.5 User guide	25
4.6 Observations while using Data Access Notifier	27
4.7 Limitations	27
4.7.1 Tested devices	28
4.8 Battery usage	28

5. Conclusion	30
6. Discussion.....	31
6.1 Future Development Opportunities.....	31
6.2 Optimal Solution	31
References.....	32
7. Glossary.....	34
7.1 Estonian Government and Technology Terms	34
7.2 Android Development Terms	34
License	36

1. Introduction

In Estonia there are a lot of state databases holding users' data, like Population Registry (*Rahvastikuregister*) and Health Portal (*Terviseportaal*). People's data in these databases is accessed by different parties for the variety of purposes. Usually those are legitimate purposes, like a doctor accessing person's health data, or even people themselves accessing their data in some information systems. However, sometimes the purpose of data access is not clear.

For the purposes of making the process more transparent, the Estonian Information System Authority (RIA) created a special service, Data Tracker (*Andmejälgija*), in 2017, through which users can check which parties have accessed their data. [1]

The Data Tracker is a people-oriented service on the state portal *eesti.ee*, which aims to ensure transparency in the processing of personal data in the public sector. The data tracker relies on the ability of each data repository to store the data processing taking place within itself in the form of logs, in order to later display it to the individual, i.e. the data subject, via the service on *eesti.ee*. [2]

Architecturally, it is a fully distributed system, i.e. the information displayed to the user comes directly from the database that implemented the Data Tracker service. At the user's request, *eesti.ee* makes a query to each of the Data Tracker services and displays the query response without saving it. [2]

The Data Tracker should display to the individual information about data processing taking place locally in the database (activities of officials-employees with personal data) as well as an overview of when data has been transferred to a third party (via X-Road to another government agency, company, etc.). [2]

The Data Tracker doesn't notify, however, when the data is accessed by someone. In order to learn about the update in the data access logs, the person has to go to the *eesti.ee* web-view and manually query access logs from specific databases.

The primary objective of this thesis is to solve this problem by creating a mobile phone app that would notify its users about near-real time updates in the data access logs.

Additionally I would like to examine existing state databases, including whether they provide access logs or not.

2. Andmejälgija

2.1 The protocol

Andmejälgija is a protocol that state databases are responsible for implementing themselves. In order for the database to offer an *Andmejälgija* service, they have to create an X-Road interface according to RIA specification¹.

X-Road is a REST-based protocol which is used for secure data exchange between Estonian information systems over the Internet.

The *Andmejälgija* X-Road interface is expected to have the following endpoints:

findUsage

A query searches the data recorder database for usage records that match the constraints given in the input. The output of the query returns all records found.[3]

usagePeriod

The time period for which usage information can be requested.[3]

heartbeat

Requesting the availability status of the tracker's usage information.[3]

¹ https://github.com/e-gov/AJ/blob/master/doc/spetsifikatsioonid/Kasutusteabe_esitamise_protokoll.md

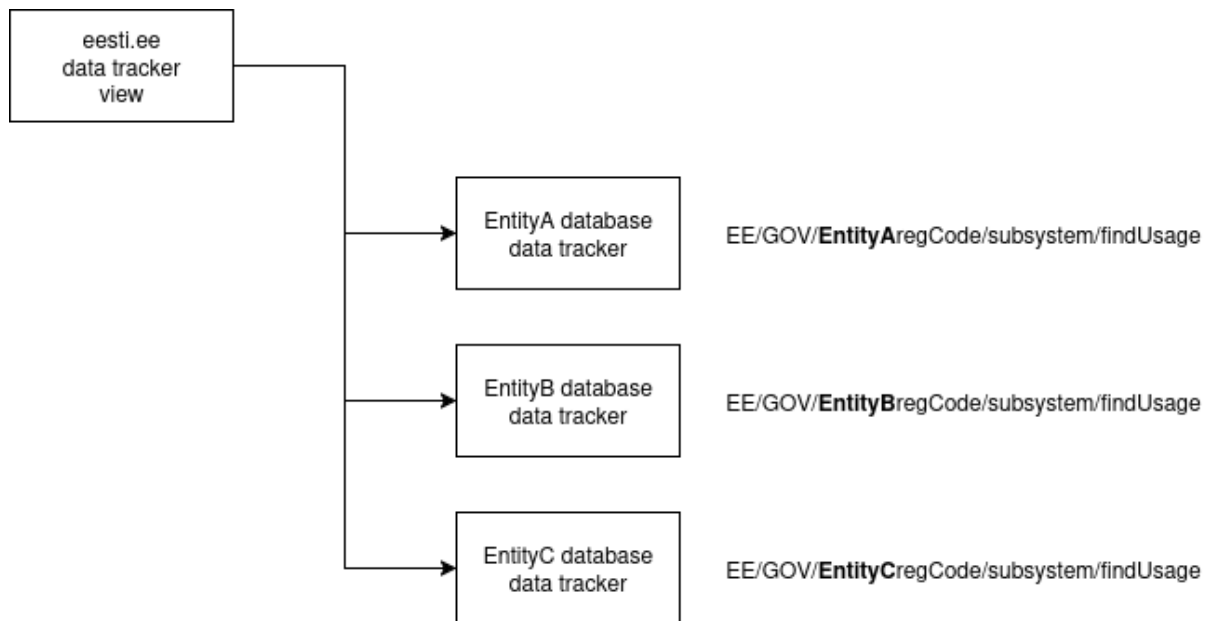


Figure 1. *Andmejälgija* (Data Tracker) system architecture showing the interaction between entities implementing data tracking and *eesti.ee* data tracker view. The data exchange happens via X-Road.

2.2 Usage

There are several ways for end-users to access their Data Tracker data. State portal *eesti.ee* provides a web-view for *Andmejälgija* where end users can access their data access logs. Recently, RIA have also published a mobile app for *eesti.ee*, that also features a view of data access logs.

Another way to access Data tracker data is through Estonian Population Registry, although, is seems to be Population Registry specific as it doesn't provide access logs from other databases

2.3 Adoption

The current adoption of *Andmejälgija* has some noticeable issues. Very often it is not at all clear why your data have been accessed, at least from the first glance. Explanation messages are vague and confusing, often being similar to "data access by personal code", making it difficult to understand the reason behind the data access, even if it was you who accessed it.

There is even an information sheet with recommendations for services implementing the *Andmeälgi* protocol, and it states that providing poor quality explanations for data access is a bad practice.²

Good practice example:

Data Processing Time	Activity	Party Receiving Personal Data
13.01.2015 10:20:27	Prescription viewed by doctor; prescription number 1018472350	Doctor Viktor Pihlakas
19.01.2018 10:58:23	Individual query for valid driver's licenses through state portal <i>eesti.ee</i>	Jaan Kask 32405023456

Table 1. Example of good data access descriptions

Bad practice example:

Data Processing Time	Activity	Party Receiving Personal Data
13.01.2015 10:20:27	PERSONAL DATA BY PERSONAL CODE	INSTITUTION X
19.01.2018 10:58:23	INDIVIDUAL EXTENDED INFO QUERY BY PERSONAL CODE	FOUNDATION Y

Table 2. Example of poor data access descriptions

Apparently, the advice is often ignored, as many institutions tend to follow the bad example in practice. The thesis author's data tracker view is flooded with entries containing meaningless descriptions like the following:

² <https://www.ria.ee/sites/default/files/documents/2022-11/Soovitusi-Andmeälgi-rakendamiseks.pdf>

Date & Time	Institution	Database	Activity
02.08.2025 11:15	Health and Welfare Information Systems Centre	Population Registry	INDIVIDUAL EXTENDED INFO QUERY BY PERSONAL CODE
01.08.2025 22:37	Health and Welfare Information Systems Centre	Population Registry	INDIVIDUAL NAME RETRIEVAL BASED ON PERSONAL CODE
01.08.2025 20:23	Education and Youth Board	Population Registry	PERSONAL DATA BY PERSONAL CODE

Table 3. Extract from author's data tracker view demonstrating poor quality descriptions

Furthermore, currently there is no law requiring institutions to implement the *Andmejälgija* protocol, meaning that its use is pretty much voluntary.

In the following chapters I will cover different state databases and their implementations of *Andmejälgija*, and whether they implement the protocol at all.

3. Overview of state databases

3.1 Databases implementing *Andmejälgija*

3.1.1 Examples of good implementation

The following information systems demonstrate an exemplary implementation of the *Andmejälgija* protocol. They provide very descriptive information about data access requests, making it clear to users what specific piece of data has been accessed, and by whom.

- Digital Registry (Digiregistratuur)
- Prescription Centre (Retseptikeskus)

3.1.2 Other databases implementing *Andmejälgija*

- Residence and Work Permit Register (*Elamislubade ja töölubade register*)
- Land Register (*Kinnistusraamat*)
- Professional Qualifications Register (*Kutseregister*)
- Taxpayers Register (*Maksukohustuslaste register*)
- Police Tactical Management Database (*Politsei taktikalise juhtimise andmekogu*)
- Agricultural Animals Register (*Põllumajandusloomade register*)
- Agricultural Subsidies and Land Blocks Register (*Põllumajandustoetuste ja põllumassiivide register*)
- Population Register (*Rahvastikuregister*)
- Social Protection Information System (*Sotsiaalkaitse infosüsteem*)
- Social Services and Benefits Register (*Sotsiaalteenuste ja toetuste register*)
- Labour Inspectorate Working Life Information System (*Tööinspektsiooni tooelu infosüsteem*)
- Unemployment Insurance Database (*Töötuskindlustuse andmekogu*)
- Information System POLIS (*Infosüsteem POLIS*)

3.2 Databases providing other form of data access tracking

3.2.1 E-File System (E-toimik)

E-File System provides its own web-view for displaying requests made about the user, independent of the standard *Andmejälgija* protocol.

Minu kohta tehtud päringud

Päringud, mis on tehtud karistusregistrist kahe viimase aasta jooksul minu, minu alaealise lapse või eestkostetava või minuga seotud juriidilise isiku kohta.

Aeg -

Päringud*

Otsi

Kuupäev	Päringu liik	Päringu sooritaja	Eesmärk
15.04.2025	Minu kohta tehtud päringute vaatamine	Arkadi Statsenko, 50305170834 (E-toimik)	Isik enda kohta
18.03.2025	Päring kõikide karistuste kohta	Kaitseressursside Amet, 70007647 (Kaitseväekohustuslaste register)	Automaatkontroll
18.02.2025	Päring kõikide karistuste kohta	Kaitseressursside Amet, 70007647 (Kaitseväekohustuslaste register)	Automaatkontroll

Figure 2. E-File System (E-toimik) web interface showing data access requests. This system operates independently of the *Andmejälgija* protocol and provides its own tracking mechanism for data access events [4].

An interesting observation regarding E-File is that similarly to the Data Tracker, it is not guaranteed to include all information about data access.

To be more specific, I once had to request a criminal record extract from Finland for employment purposes, as I had to undergo a background check. At some point I enquired about the status of my application and they responded, mentioning that it took longer than anticipated due to them also having requested my criminal record extract from Estonia, a country of my citizenship.

The inquiry to Finnish Legal Register was made on the 14th of May, 2025. Despite that there are no relevant records in the e-File system in the period from 14.05.2025 to 20.05.2025 (The day the extract was posted).



ORK Rikosrekisteri <rikosrekisteri@om.fi>

to me ▾

Mon, May 26, 3:20 PM



Hello!

Thank you for your email.

Since you also have the citizenship of Estonia, we requested a criminal record information from there as well, it caused a delay in the process. We have prepared the extract for you on 20th of May 2025. The extract and invoice have been sent separately by regular mail to ICOVER SAS.

Hopefully you/they will receive it soon.

Best regards,



Palveluasiantuntija / Servicespecialist / Service Specialist

Oikeusrekisterikeskus / Rättsregistercentralen / Legal Register Centre

PL / PB / P.O. Box 157, 13101 Hämeenlinna / Tavastehus

www.oikeusrekisterikeskus.fi



Figure 3. Email response from Finnish Legal Registry confirming that they requested criminal record information from Estonia, demonstrating how data access events may not be logged in Estonian systems when authorities provide information to foreign institutions [4].

Date	Query Type	Query Performer	Purpose
20.05.2025	Query for all convictions (incl. archive data)	Arkadi Statsenko (E-File)	Person about themselves
18.05.2025	Query for all convictions	Defence Resources Agency, 70007647 (Conscripts Register)	Automatic control
15.05.2025	Query for all convictions (incl. archive data)	Arkadi Statsenko (E-File)	Person about themselves
15.05.2025	Query for all convictions (incl. archive data)	Arkadi Statsenko (E-File)	Person about themselves
15.05.2025	Query for all convictions (incl. archive data)	Arkadi Statsenko (E-File)	Person about themselves

As shown in the table above, there is no record of Estonian authorities providing criminal record information to the Finnish Legal Registry during the period from May 14th to May 20th, 2025, despite confirmation from Finland that such a request was made. This demonstrates a significant gap in data access logging when Estonian institutions provide information to foreign authorities, highlighting limitations in the current tracking systems.

3.3 Databases not providing any kind of data access tracking

- Schengen Information System
- Border Control Database (*Piirikontrolli andmekogu - PIKO*)
- Vehicle Registration Database (*Liiklusregister*)
- Court Information System (*Kohtute infosüsteem - KIS*)
- Employment register (*Töötamise register (TÖR)*)

These are only a few examples. There are a lot more databases that don't provide any kind of data access tracking. It is in fact much easier to count those databases that do provide some kind of data tracking than those that don't. A more complete view of what state databases exist is available from the State Information System Management System (*Riigi Infosüsteemi Haldussüsteem*)³. At the time of writing (9.08.2025) the search returns information about 1386 databases in total, including 1100 about those in use.

³ <https://www.riha.ee/Infosüsteemid>

4. Implementation

4.1 Description

The main objective of this thesis has been to develop a solution that would notify users of data access events logged in the *Andmejälgija* (Data Tracker) system. The resulting solution provides near real-time notifications when government institutions or other authorized entities access an eID holder's personal data.

4.2 Discussion of implementation choice

There are several possible approaches for developing such a solution. In this section I will discuss different implementation options as well as their advantages and disadvantages.

4.2.1 X-Road service

Andmejälgija specifications require entities responsible for the state databases to implement an X-Road interface themselves, as described in 2.1, so one option would be to create an X-Road service that would query *Andmejälgija* data over X-Road directly from those entities. The main advantage of this approach would be the freedom in how to notify users of changes to the access logs. The service could support various communication channels, including instant messenger bots, e-mail and others. However, the list of requirements to operate such a service is daunting.

- In order to join X-Road, legal entity is needed.
- Permission has to be requested from every X-Road service you want to query data from. Specifically, considering that each information system is responsible for implementing the *Andmejälgija* protocol themselves, the permission would be required from each entity responsible for the information systems from which the access logs would be requested.
- As part of X-Road network, you need to operate a Security Server. It can be self-hosted anywhere for testing, but for production you need to have a Hardware Security Module (HSM), in order to be compliant with eIDAS requirements. For example, a Thales Luna PCIe HSM A700 suitable for eIDAS qualified digital signing costs €8,990 (€10,877.90 incl. VAT) [5]. Alternatively, renting a production ready Security server from Telia costs 210€ per month [6]
- Finally, considering that this approach would include creating a service that would store and transmit user's data, it would be required to follow strict regulatory requirements

including GDPR. Staying compliant requires a continuous effort, and in case of failure to do so, there are harsh fines.

Satisfying this criteria is difficult and expensive. Additionally, even if I succeeded, people would have to consent and entrust me with their data in order to be able to use the solution, which is not in the spirit of creating a solution for monitoring personal data access as another data controller would be introduced.

4.2.2 Standalone approach

This approach uses the *eesti.ee* session for accessing *Andmejälgija* data. Once the user is logged in to the *eesti.ee* state portal, certain internal API endpoints become available. Namely the GET endpoint `https://www.eesti.ee/andmejalgiija/api/v1/usages` can be used to query *Andmejälgija* data. The endpoint requires a parameter `dataSystemCodes` with which specific databases can be specified. For example, the GET request `/usages?dataSystemCodes=digiregistratuur&dataSystemCodes=rahvastikuregister` would request access logs from *Digiregistratuur* and *Rahvastikuregister*.

The main advantage of this approach is the absence of all disadvantages of the X-Road approach: there is no need for any kind of bureaucracy and the solution could be an open-source project, available for anybody to compile and use. There arises a problem, however. What about the notification part? Do I expect users to set everything up on their hardware, including relevant communication channels? That would narrow down the project's user base to technical people knowing how to self-host, and having a server.

That's why I thought that creating a mobile application would be the most optimal approach. A mobile application would keep the *eesti.ee* session alive and poll the *Andmejälgija* API. This approach combines ease of setup and use while keeping the solution standalone, without requiring a central server.

4.3 Development

First, I had to determine how feasible and reliable the standalone approach would be, especially as I was dealing with a user-facing service that I would have to effectively reverse-engineer. The main challenge was to keep the session alive for as long as possible.

The *eesti.ee* session is established using one of the strong authentication methods via the *TARA* (Trusted Authentication and Authorization) service. Once the user is authenticated using ID-card,

Mobile-ID, Smart-ID, or eID solution of another EU country, a JWT token is issued, valid for 30 minutes. The session management is then handled by *GovSSO* (Government Single Sign-On), which maintains the master server-side session token and imposes certain limitations as discussed in 4.3.7.

Once the user is authenticated, certain internal API endpoints become available, including those for refreshing the JWT token and getting *Andmejälgija* access logs. The JWT refresh endpoint returns Set-Cookie headers that include an updated JWTTOKEN on success, with an updated expiration date (30 minutes from the moment of extension).

To test whether the session could be kept alive indefinitely, I used the Tab Reloader extension for Firefox by James Fray [7]. Using this browser extension, I configured the browser to automatically reload the tab with the JWT token refresh endpoint. After an 8-hour experiment, I deemed the approach successful.

The next step was the development of the **Data Access Notifier** Android application. I chose to develop it using the Kotlin programming language due to it being the new standard for developing Android applications [8], as well as its more expressive syntax and promise of faster development speed.

The application is compatible with all Android versions starting from Android 8.1, meaning that it works on approximately 96.4% of Android devices according to the Android Studio API version distribution chart [9, 10].

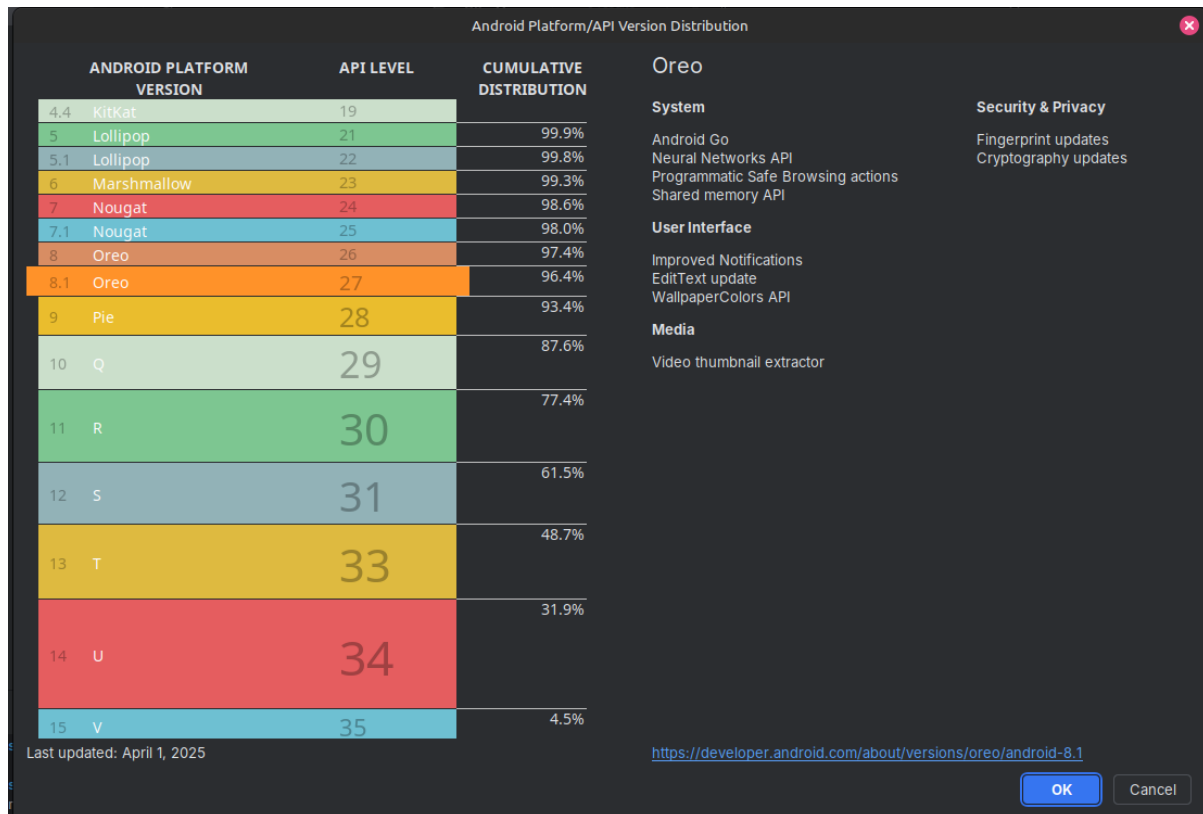


Figure 4. Android Studio API Level Distribution showing compatibility percentages for different Android versions. The chart indicates that Android 8.1 (API level 27) and higher covers approximately 96.4% of active devices.

The software is open-source, licensed under the MIT license.

Currently, the application support is limited to Android 8.1 and higher. While support for iOS and other platforms is not planned due to limited development time and resources, anybody who deems this solution useful is encouraged to port it to platforms other than Android

4.3.1 Authentication

To gain access to *eesti.ee* internal APIs, the user needs to be authenticated via *TARA*. My initial plan was to reverse engineer the authentication flow and authenticate the user by sending raw HTTP requests and returning responses. This approach, however, would be fragile and require substantial engineering effort to develop and maintain.

Instead, I decided to use Android WebView for authentication as a more reliable and convenient approach for both the user and the developer. By leveraging WebView, the **Data Access Notifier** application can present the official *TARA* login page directly to the user, allowing them to authenticate using their preferred method (ID-card, Mobile-ID, Smart-ID, or eID of another

EU country). This approach significantly reduces the risk of breaking changes due to updates on the *eesti.ee* or *TARA* platforms. Once the user is authenticated, the application can access the necessary session cookies and JWT token to interact with the internal APIs.

4.3.2 Interacting with *eesti.ee* internal APIs

For interacting with *eesti.ee* internal APIs, I used the OkHttp library by Square, Inc. [11]. I utilized the following API endpoints:

- **GET** `https://www.eesti.ee/timur/jwt/extend-jwt-session`

An endpoint for extending JWT session.

- **GET** `https://www.eesti.ee/andmejalgiija/api/v1/usages`

An endpoint for querying Data Tracker access logs.

Query parameters:

- `dataSystemCodes`: The name of the information system the access logs are queried from.

Response body format (JSON):

```
{
  "findUsageResponses": [
    {
      "logTime": "2025-08-03T22:40:47",
      "receiver": "Tervise ja Heaolu Infosüsteemide Keskus",
      "infoSystemCode": "rahvastikuregister",
      "action": "ISIKU NIME VÄLJASTAMINE ISIKUKOODI PÕHJAL"
    },
    ...
  ]
}
```

Response fields:

- `logTime`: Timestamp of the data access event in ISO 8601 format
- `receiver`: Registry code of the entity that accessed the data
- `infoSystemCode`: Code identifying the information system
- `action`: Description of the action performed (in Estonian)

Both endpoints require a valid JWT token to succeed. WebView's CookieManager serves as a single source of truth in this application, where all cookies are loaded from for making requests and written to upon receiving responses.

4.3.3 Keeping the session alive

The API requests are authenticated using a JWT token cookie, the lifetime of which is 30 minutes. This means that the **Data Access Notifier** application must either run a continuous foreground service or be awakened periodically to request an updated JWT token and send requests to *Andmejälgija*.

On Android, there are several ways to accomplish this, although it is worth mentioning that Android is very restrictive regarding background tasks and battery usage. I tried several approaches and will describe my experience with them in the context of developing this application.

4.3.4 WorkManager approach

According to the Android documentation, WorkManager is the recommended solution for persistent work. Work is persistent when it remains scheduled through app restarts and system reboots. This seemed to describe my use case very well, so this was the first approach that I tried.

Work is defined in WorkManager using a WorkRequest. There are several WorkRequest types available, including PeriodicWorkRequest and OneTimeWorkRequest.

Using PeriodicWorkRequest, it is possible to schedule periodic tasks. The important limitation here is that it is not possible to schedule a task to execute more frequently than every 15 minutes. While this may sound appropriate for the use case of having to renew the session at least every 30 minutes, in practice, the situation is different.

During development, I discovered a critical difference between session handling in web browsers versus HTTP client libraries. While the browser-based testing with Tab Reloader successfully maintained the session by refreshing the JWT token every 15 minutes, the same approach using OkHttp in the Android application failed to keep the session alive.

The exact cause of this discrepancy remains unclear. It appears that the *eesti.ee* session management system expects more frequent interaction or handles cookie management differently when requests originate from HTTP client libraries compared to full web browsers. Indeed, with shorter intervals of 5 minutes, the session remains alive. This behavioral difference rendered the PeriodicWorkRequest approach with 15-minute intervals unsuitable for maintaining persistent sessions in the application context.

While OneTimeWorkRequest is designed for one-time tasks, it is possible to chain them by having each one-time work create another one-time work at the end of its lifespan. This approach allows setting custom intervals shorter than 15 minutes, circumventing the limitation imposed by PeriodicWorkRequest.

While OneTimeWorkRequest may appear to be a suitable solution, WorkManager itself proved to be rather unreliable for my use case after testing. The limitation of WorkManager is that tasks scheduled by it are managed by the system and may be deferred if deemed necessary by the Android system. For example tasks may not execute if the phone has low battery level, or even in other scenarios depending on the aggressiveness of the Android variant regarding the restrictions imposed on application behavior.

4.3.5 Foreground service approach

A foreground service is one of the most reliable types of tasks an application can schedule. In contrast to background services, foreground services are considered to be of higher priority by the system and thus are not candidates for termination when the system is low on memory or the phone has low battery level.

However, an important limitation has been introduced in recent Android versions. More specifically, starting from Android 15, the system places restrictions on how long certain foreground services are allowed to run while the application is in the background. Currently, this restriction only applies to `dataSync` and `mediaProcessing` foreground service types. The foreground services of those types are allowed to run for a total of 6 hours in a 24-hour period [12].

The most appropriate foreground service type for my use case is `dataSync`, meaning that it falls under the 6-hour per day restriction. While it should be possible to choose an arbitrary foreground service type regardless of the actual task type, doing so is not the cleanest approach, especially considering the existence of better alternatives. Furthermore, doing so may also impact the application's acceptance into the Google Play Store.

4.3.6 Hybrid AlarmManager approach

After evaluating the limitations of `WorkManager` and the restrictions on foreground services, I ultimately settled on a hybrid approach combining `AlarmManager` with a `dataSync` foreground service. `AlarmManager` is a system service that allows scheduling operations to be executed at specific times, even if the application is not running. By using `AlarmManager` to trigger the application at regular intervals (e.g., every 5 minutes), it is ensured that the session renewal and data polling tasks are reliably executed.

When the alarm fires, the application starts a short-lived `dataSync` foreground service to perform the necessary network operations: refreshing the JWT token and querying the *Andmejälgi* API. This approach leverages the reliability of foreground services for critical tasks while minimizing battery usage by only running the service when needed. `AlarmManager` itself is not subject to aggressive background restrictions and can wake the application even in Doze mode, making it suitable for periodic tasks.

This solution proved to be robust and reliable across different Android versions and device manufacturers. Both `AlarmManager` and foreground services are extremely reliable even under

low battery conditions, ensuring that the session remains alive and notifications are delivered promptly to the user.

4.3.7 Session limitations

During the development and testing process, I discovered an important server-side limitation that affects the overall feasibility of the session-based approach. Despite the successful implementation of session renewal mechanisms, the *GovSSO* session management system imposes a hard limit of 12 hours on session duration. After this period, the session expires permanently and cannot be renewed.

The problem was discovered when it became apparent that JWT extension fails regularly with response code 500 (Internal Server Error) from the server and no further information. Following a number of retries, the server would send code 400 (Bad Request). At some point I started looking for a pattern and discovered that the session extension starts failing after exactly 12 hours. I investigated the *GovSSO* GitHub repository [13] and found the answer in the configuration file.

In the file `/src/main/resources/application.yml`, the following configuration parameter is defined:

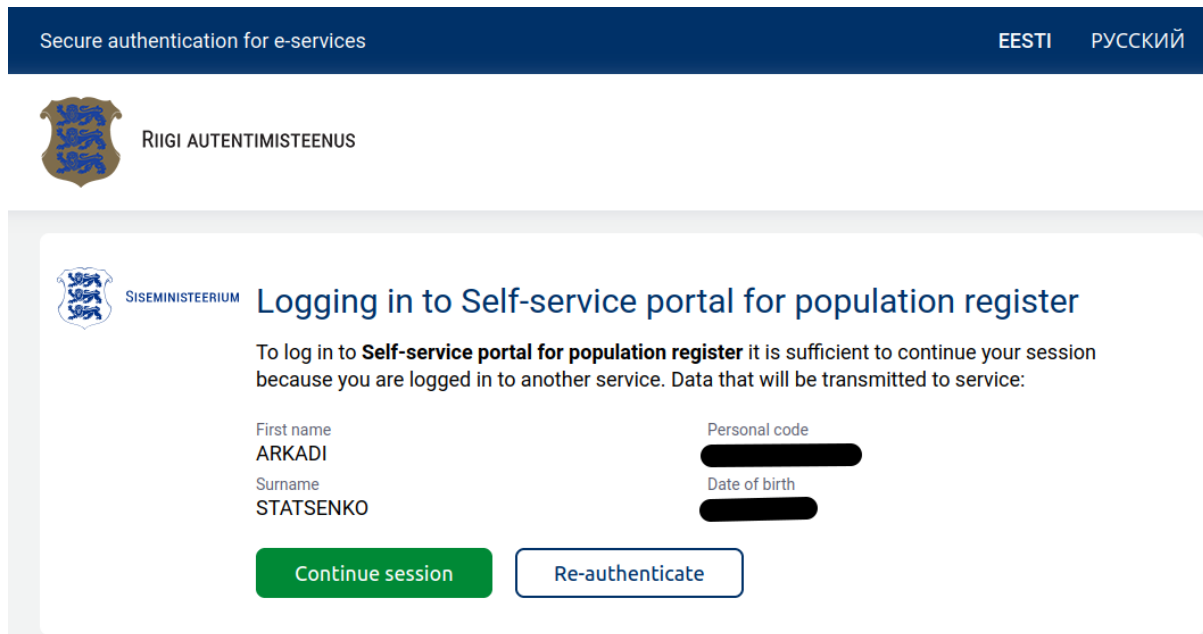
```
session-max-duration-hours: 12
```

The function responsible for checking the `notBefore` date of the JWT token is shown below [14]:

```
private boolean isNbfValid(JWT idToken) throws ParseException {
    Date idTokenDate = idToken.getJWTClaimsSet().getNotBeforeTime();
    Date currentDate = new Date();
    long diffInMillis = Math.abs(currentDate.getTime() -
                                   idTokenDate.getTime());
    long diffInSeconds = TimeUnit.SECONDS.convert(diffInMillis,
                                                  TimeUnit.MILLISECONDS);
    long maxDurationInSeconds = TimeUnit.SECONDS.convert(
        ssoConfigurationProperties.getSessionMaxDurationHours(),
        TimeUnit.HOURS);

    return diffInSeconds <= maxDurationInSeconds;
}
```

Initially, I hypothesized that this limitation was related to the JWT token issued to the client, with the server rejecting it once it reaches the age of 12 hours. This would have been an easy fix, as there is another way to extend the session that would issue a "fresh" JWT token each time. As can be derived from the name "GovSSO", this authentication service supports single-sign-on, meaning that if you are signed into one state e-service, you can log into another by clicking a "Continue session" button. The same can be achieved for the same state portal where the system was initiated, if you go to the right URL. Automating this kind of flow is straightforward.



Secure authentication for e-services EESTI РУССКИЙ

RIIGI AUTENTIMISTEENUS

SISEMINISTEERIUM

Logging in to Self-service portal for population register

To log in to **Self-service portal for population register** it is sufficient to continue your session because you are logged in to another service. Data that will be transmitted to service:

First name	Personal code
ARKADI	[REDACTED]
Surname	Date of birth
STATSENKO	[REDACTED]

Continue session Re-authenticate

Figure 5. A single sign-on interface showing the "Continue session" option that allows users to extend their authentication session across different government e-services [15].

However, after experimenting with this method, I discovered that the 12-hour limitation applies to a different token than initially expected. The time validation is not performed on the client-side JWT token that the application receives, but rather on a server-side master JWT token that is created during the initial *TARA* authentication process (ID-card, Mobile-ID, Smart-ID, etc.) and managed by *GovSSO*.

When a user logs into additional services using the single-sign-on functionality, these secondary sessions are all linked to the same original server-side JWT token managed by *GovSSO*. This architecture means that regardless of which session extension method is used, the 12-hour limit cannot be circumvented because all sessions share the same underlying authentication token that expires after 12 hours.

This limitation has significant implications for the notification system’s reliability. Users who expect continuous monitoring must re-authenticate manually at least once every 12 hours to maintain the service functionality. However, this constraint is substantially mitigated by the API’s design, which ensures that no data access events are permanently lost, though it does introduce periodic interruptions in real-time monitoring.

Specifically, the API returns comprehensive historical access logs, ensuring that once a user re-authenticates after a session expiry, they will receive all notifications for events that occurred during the period when they were logged out. This behavior is crucial for maintaining data completeness and ensuring that no critical access events are missed due to authentication gaps. The technical implementation of this log handling is detailed in 4.3.8.

4.3.8 Handling logs and notification delivery

In this section the following aspects of access log handling will be tackled: their storage, deduplication and filtering.

First of all, user inquiries about themselves are also considered data access and therefore produce access logs. This includes *Andmejälgija* data queries. However, making frequent queries to the *Andmejälgija* API produces a lot of logs.

This makes it important to distinguish logs made by the user themselves from the logs made by third-parties. Thankfully, the `receiver` field contains the user’s Estonian personal code in case the query is made by them. In fact, filtering by whether the receiver field contains user’s personal code is exactly how the access log filtering is implemented on *eesti.ee* itself when the user ticks the ”Exclude queries I have made/initiated” checkbox. Here is the *eesti.ee* code extract that demonstrates this approach [16]:

```
this.accountService.getUserInfo().subscribe((o) => {
  this.filteredData.data = this.filteredData.data.filter(
    (d) => !d.receiver.toUpperCase()
      .includes(o.personalCode.replace(/^EE/, ""))
  );
});
```

This is exactly how I implemented log filtering in the application. The user’s personal code that is needed for filtering is extracted from the JWT token, along with the first name which is used to greet the user.

For deduplication of access logs, I utilized the Set data structure. Each access log entry is represented as a data class, and a Set of these entries is maintained to ensure that duplicate logs are not stored or processed multiple times. This approach leverages the fact that Sets in Kotlin do not allow duplicate elements.

To persist the access logs across app restarts and device reboots, I used Proto Data Store, which is Android's recommended solution for storing typed objects or lists in a transactional and type-safe manner. Proto Data Store uses Protocol Buffers [17] for serialization, offering efficient storage and schema evolution capabilities.

Whenever new logs are fetched from the API, they are first filtered to exclude self-access events. The newly fetched logs are then merged with the existing logs stored in Proto Data Store, and the combined set is deduplicated using Kotlin's Set data structure. To determine which logs are new and should trigger notifications, the application computes the set difference between the combined deduplicated logs and the previously stored logs. Only the new items resulting from this difference are used to notify the user. This approach ensures that notifications are sent only for genuinely new access events, while avoiding redundant alerts for logs that have already been seen.

4.4 Software distribution

The **Data Access Notifier** application is distributed as an open-source project hosted on GitHub [18]. Users can download the APK file directly from the GitHub Releases page [19]. To keep the application up to date, users can utilize third-party applications such as Obtainium [20], which enables automatic updates for applications distributed outside of traditional app stores.

While the application is not currently available from app stores like Google Play Store, this possibility is not excluded in the future. Any updates regarding the application distribution would be posted on GitHub README.

4.5 User guide

After installation and opening for the first time, the **Data Access Notifier** application will prompt user for necessary permissions, which are the ability to schedule alarms (for AlarmManager to work) as well as to post notifications.

Once the necessary permissions are given, the Log In screen appears. When Log In button is pressed, an Android WebView is opened, redirecting the user to *TARA* Log In screen. There user is free to choose whatever state authentication option they choose.

And that's it! As soon as authentication is successfully completed, you will start receiving notifications as soon as there are any new data access logs returned from the server. There is also a view inside of the application showing all data access logs saved.

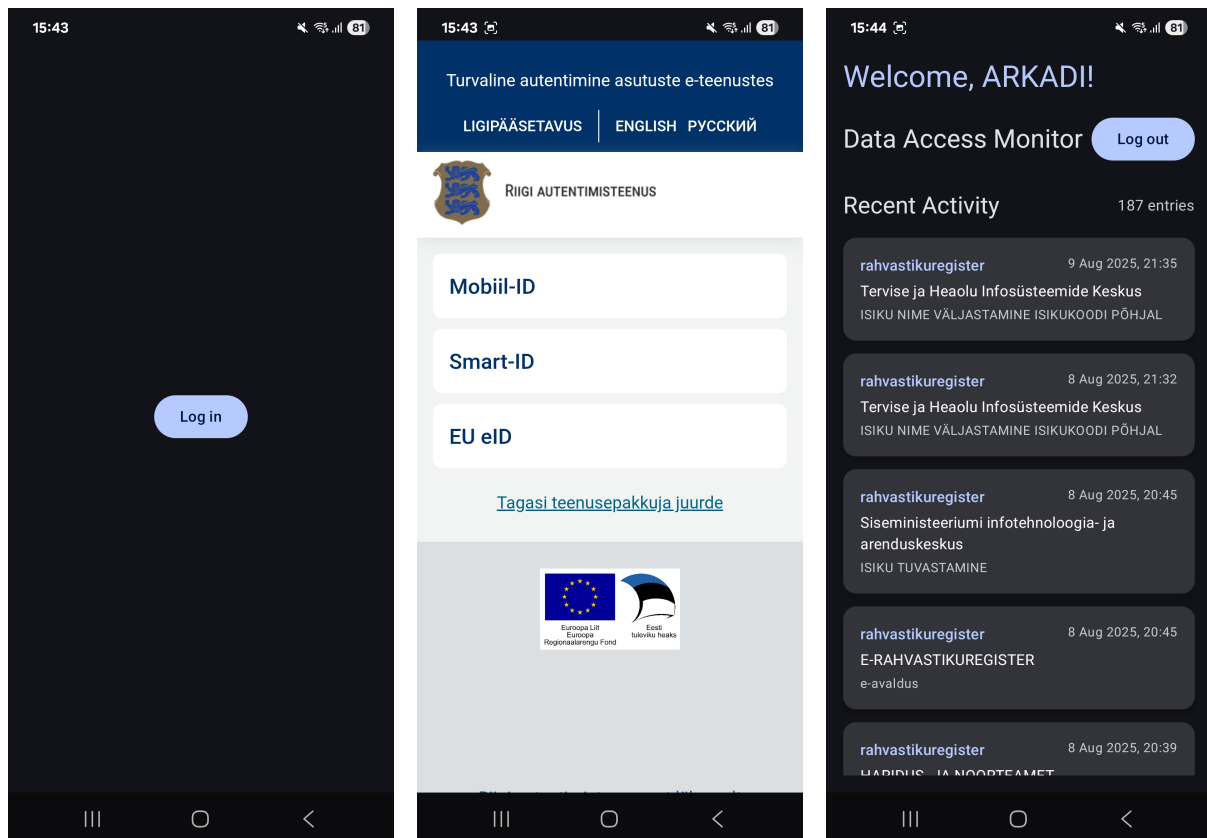


Figure 6. Authentication to Data Access Notifier

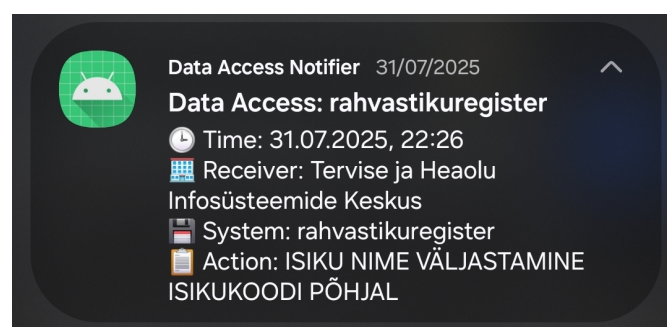


Figure 7. Data access notification

4.6 Observations while using Data Access Notifier

While using the **Data Access Notifier** application over several weeks, I made several interesting observations about the behavior of the *Andmejälgija* API and the reliability of real-time access logging.

One notable discovery was related to delayed log entries. I occasionally received notifications for access events dated several months in the past - specifically, events from May 2025 would sometimes appear in August 2025. Initially, I suspected this might be a deduplication issue within the application, but after examining the stored notification history, it became clear that these were genuinely new log entries that had not been seen previously by the application.

This behavior suggests that there may be delays in the centralized logging infrastructure of *Andmejälgija*, where some access events are not immediately recorded or made available through the API.

The application proved to be quite reliable in maintaining the authentication session during the 12-hour window. The hybrid AlarmManager approach (described in 4.3.6) successfully maintained connectivity even during extended periods of device inactivity or network fluctuations.

From a user experience perspective, the notifications provided valuable insights into data access patterns that would otherwise remain invisible to citizens. This transparency feature proved particularly useful for understanding which organizations access personal data and how frequently such access occurs.

4.7 Limitations

The most significant limitation of the **Data Access Notifier** application is the 12-hour session expiry imposed by the *GovSSO* authentication system. As discussed in 4.3.7, this server-side restriction cannot be circumvented through known technical means, requiring users to manually re-authenticate.

While the application successfully preserves all access log data during authentication gaps - ensuring no events are permanently lost - the periodic interruptions in real-time monitoring represent a substantial usability constraint.

Additionally, the application's scope is currently limited to Android devices running version 8.1 and higher. While this covers approximately 96.4% of the Android ecosystem, users of

iOS or other mobile platforms cannot benefit from this solution without platform-specific implementations.

The reliance on reverse-engineered internal APIs also introduces inherent fragility. Changes to *eesti.ee*'s internal API structure or authentication mechanisms could potentially break the application's functionality without notice, requiring an update to the application to fix the issue.

Additionally, the session-based approach requires the device to maintain active network connectivity and remain operational to perform periodic authentication renewals and API polling. Extended periods of device inactivity, airplane mode, or network unavailability will result in session expiry and temporary loss of monitoring capabilities.

The application also doesn't currently support the access logging implemented in *E-File*.

Finally, the *Andmejälgija* protocol itself is not regulated by the law, and information systems are not required to implement it, and even if they do, not all data access events are guaranteed to be timely distributed or logged at all.

4.7.1 Tested devices

The **Data Access Notifier** application has been tested on the following devices:

- Xiaomi Poco X3 Pro running CrDroid without Google Services
- Xiaomi Redmi 7A running Lineage OS without Google Services
- Samsung Galaxy S25 with stock OS (Google Services included)

4.8 Battery usage

Data Access Notifier application maintains relatively low power consumption despite its continuous background operation. The hybrid AlarmManager approach with short-lived foreground services appears to be efficient in terms of battery impact.

While the battery utilization hasn't been actively monitored, some screenshots have been taken on Samsung Galaxy S25 at different times during the active use period.

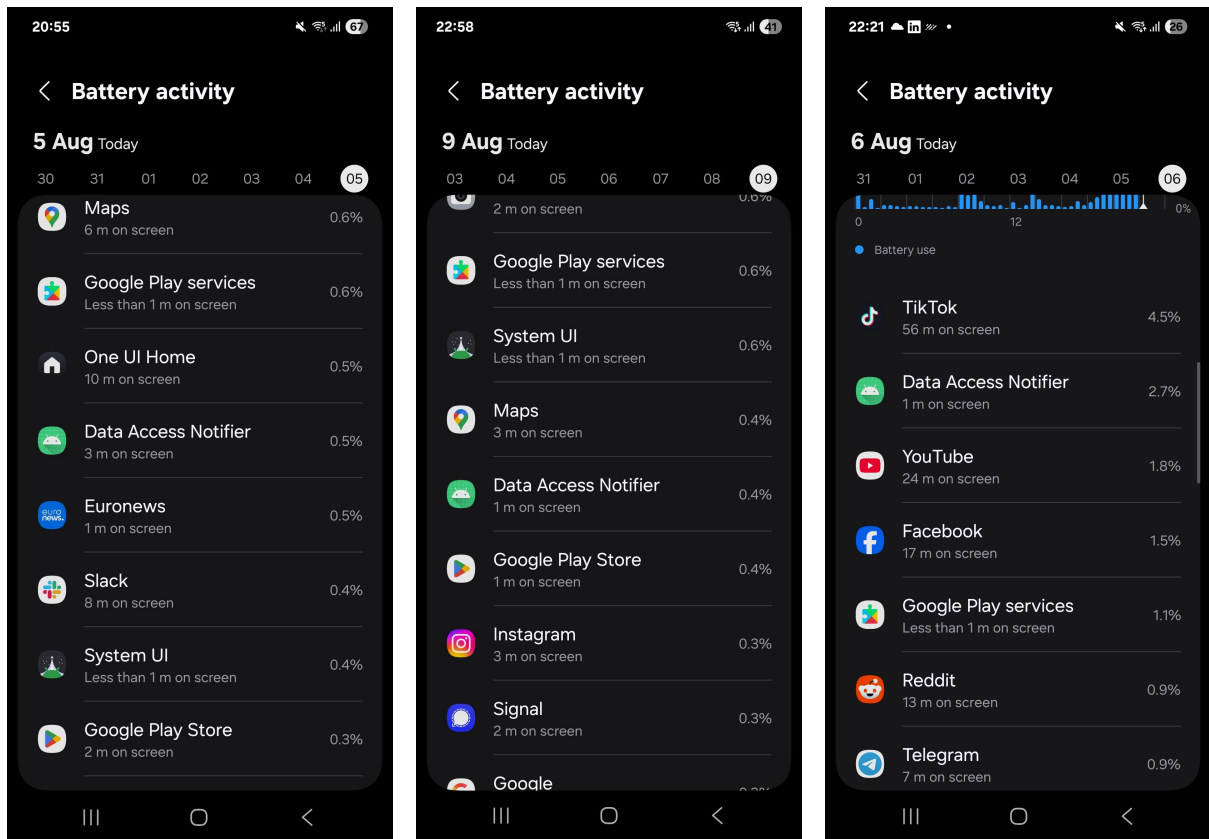


Figure 8. Battery usage for Data Access Notifier application on Samsung Galaxy S25 showing power consumption over different time periods during active use.

5. Conclusion

The prime objective of this thesis, which is to create a mobile phone app, Data Access Notifier, has been achieved. Even though it has known limitations, it still provides valuable functionality in terms of providing notifications about data access that could have been missed otherwise, especially considering that, as observed in section 4.6, sometimes old logs are delivered with delay of several months.

6. Discussion

While an initial solution has been successfully implemented, there remains considerable scope for further development and enhancement.

6.1 Future Development Opportunities

Several potential approaches could address the current 12-hour session limitation by exploring alternative authentication flows and API endpoints. One avenue for investigation would involve reverse-engineering the official *eesti.ee* Android application to analyze its authentication mechanisms, for example by decompiling its APK and studying the resulting source code. The application is known to support biometric authentication with PIN code fallback functionality. Through proper reverse engineering and reimplementation of these authentication flows, it may be theoretically possible to automate PIN code resubmission and maintain indefinite session validity.

However, such an approach would constitute a substantial undertaking that could warrant a separate research project. Additionally, this method would likely introduce significantly greater implementation complexity compared to the current solution, potentially making it more fragile and maintenance-intensive.

6.2 Optimal Solution

The most effective solution would be for RIA (Information System Authority) to implement equivalent functionality directly within the *eesti.ee* platform. Such an official implementation would be relatively straightforward to develop and would provide superior reliability and future-proofing compared to any third-party solution, including the one presented in this thesis.

References

- [1] ERR. Leht: rahvastikuregistris nuhitakse ebaseaduslikult inimeste andmetes. <https://www.err.ee/590454/leht-rahvastikuregistris-nuhitakse-ebaseaduslikult-inimeste-andmetes> (08/10/2025).
- [2] Information System Authority (RIA). Andmejälgija / Data Tracker. Version 1.0.3. Apr. 15, 2025. <https://github.com/e-gov/AJ>.
- [3] Information System Authority (RIA). Andmejälgija / Data Tracker protocol specification. Version 1.4.1. Apr. 15, 2025. https://github.com/e-gov/AJ/blob/master/doc/spetsifikatsioonid/Kasutusteabe_esitamise_protokoll.md.
- [4] Statsenko A. E-File System Screenshots. Screenshots taken from the Estonian E-File System (E-toimik) showing data access tracking interface. 2025.
- [5] QSCD.eu. Thales Luna PCIe HSM A700. Price: €8,990 (€10,877.90 incl. VAT). eIDAS qualified QSCD capable. <https://www.qscd.eu/hsms-hardware-security-modules/thales-luna-pcie-hsm-a700/> (08/09/2025).
- [6] AS T. E. X-tee turvaserver. <https://www.telia.ee/ari/it-teenused/serverid-ja-pilv/x-tee-turvaserver/> (08/09/2025).
- [7] Fray J. Tab Reloader. <https://addons.mozilla.org/en-US/firefox/addon/tab-reloader/> (08/09/2025).
- [8] Android Developers. Kotlin-first approach for Android development. <https://developer.android.com/kotlin> (08/09/2025).
- [9] Google. Android API Level Distribution Data. <https://dl.google.com/android/studio/metadata/distributions.json> (08/09/2025).
- [10] Statsenko A. Android Studio API Level Distribution Screenshot. Screenshot taken during Android project creation in Android Studio, showing API level compatibility statistics. 2025.
- [11] Square I. OkHttp. <https://square.github.io/okhttp/> (08/09/2025).
- [12] Android Developers. Behavior changes: apps targeting Android 15 - DataSync timeout. <https://developer.android.com/about/versions/15/behavior-changes-15#datasync-timeout> (08/09/2025).
- [13] Information System Authority (RIA). GovSSO Session. <https://github.com/e-gov/GovSSO-Session> (08/09/2025).
- [14] Information System Authority (RIA). GovSSO Session - HydraService.java. <https://github.com/e-gov/GovSSO-Session/blob/a95c781a8f7b05bde550acb34b8292b6c884b8>

- 02/src/main/java/ee/ria/govsso/session/service/hydra/HydraService.java#L472-L480
(08/09/2025).
- [15] Statsenko A. eesti.ee Portal Screenshots. Screenshots taken from the Estonian state portal (eesti.ee) showing GovSSO session management interface. 2025.
 - [16] Information System Authority (RIA). eesti.ee - Estonian State Portal. JavaScript code excerpt. <https://www.eesti.ee> (08/09/2025).
 - [17] Google. Protocol Buffers. <https://developers.google.com/protocol-buffers> (08/09/2025).
 - [18] Statsenko A. Data Access Notifier. <https://github.com/ArkadSt/DataAccessNotifier> (08/09/2025).
 - [19] Statsenko A. Data Access Notifier Releases. <https://github.com/ArkadSt/DataAccessNotifier/releases> (08/09/2025).
 - [20] R. I. Obtainium. <https://github.com/ImranR98/Obtainium> (08/09/2025).
 - [21] Centre of Registers and Information Systems (RIK). E-File. <https://www.rik.ee/en/international/e-file> (08/09/2025).
 - [22] Android Developers. WorkManager. <https://developer.android.com/reference/androidx/work/WorkManager> (08/09/2025).

7. Glossary

7.1 Estonian Government and Technology Terms

Andmejälgija Data Tracker - Protocol developed by RIA for tracking personal data access across state databases.

eesti.ee Official Estonian state portal providing access to various government e-services.

TARA Trusted Authentication and Authorization service - Estonian government service that handles initial user authentication using strong authentication methods (ID-card, Mobile-ID, Smart-ID, EU eID).

GovSSO Government Single Sign-On service - manages session state and enables users to access multiple Estonian government e-services with a single authentication. Works in conjunction with TARA to provide seamless access across government platforms.

Digiregistratuur Digital patient registry system in Estonia.

Rahvastikuregister Estonian Population Register containing basic demographic information.

E-File The E-File is a central information system that provides an overview of the different phases of criminal, civil, administrative and misdemeanour proceedings, procedural acts and court adjudications to all the parties involved, including the citizen and their representatives.[21] Includes the data access tracker for Criminal Records Database.

RIA *Riigi Infosüsteemi Amet* - Information System Authority, the Estonian government agency responsible for information systems.

X-Road Estonian data exchange platform that enables secure data exchange between information systems.

eIDAS Electronic Identification, Authentication and Trust Services - EU regulation for electronic identification and trust services.

7.2 Android Development Terms

AlarmManager Android system service for scheduling operations to be executed at specific times.

WorkManager Android Jetpack library for deferrable, guaranteed background work. Uses JobScheduler for API 23+, and AlarmManager + BroadcastReceiver for API 14-22.

Automatically chooses the best execution method based on Android version and system constraints [22].

Foreground Service Android service that runs in the background but displays a persistent notification to the user, making the user aware of the ongoing operation. Despite running in the background, it's called "foreground" because it maintains user visibility through the notification.

Background Service Android service that runs without user awareness and without displaying notifications. These services are heavily restricted by the system to preserve battery life and system performance.

WebView Android component that displays web content within an application.

OkHttp HTTP client library for Android and Java applications.

Proto Data Store Android library for storing typed objects using protocol buffers.

Doze Mode Android power-saving feature that defers background CPU and network activity for applications.

APK Android Package - file format used to distribute and install applications on Android.

JWT JSON Web Token - method for transmitting information between parties as a signed JSON object.

License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, **Arkadi Statsenko**,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

State registry data access notifications for Estonian eID holders,

supervised by **Daniel Würsch, MSc;**

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;
3. am aware of the fact that the author retains the rights specified in points 1 and 2;
4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Arkadi Statsenko

09/08/2025