UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Arkadi Statsenko

Estonian Data Tracker (Andmejälgija) Notifier

Bachelor's Thesis (9 ECTS)

Supervisor:
Daniel Würsch, MSc

Tartu 2025

Estonian Data Tracker (Andmejälgija) Notifier

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 it's 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. Additionaly, the overview of the existing state databases will be provided, including whether they provide access logs or not.

Keywords: Andmejälgija

CERCS: CODE Code name

Andmejälgija teavitaja

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 igaühe eeliste ja puudustega. Lisaks antakse ülevaade olemasolevatest riiklikest andmebaasidest, sealhulgas sellest, kas nad pakuvad juurdepääsulogisid või mitte.

Võtmesõnad: Andmejälgija

CERCS: KOOD Koodi nimi eesti keeles

Contents

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.[1]

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.[1]

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.).[1]

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 it's users about near-real time updates in the data access logs.

[1] https://www.err.ee/590454/leht-rahvastikuregistris-nuhitakse-ebaseaduslikult-inimeste-andmetes

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 specification[2].

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.[2]

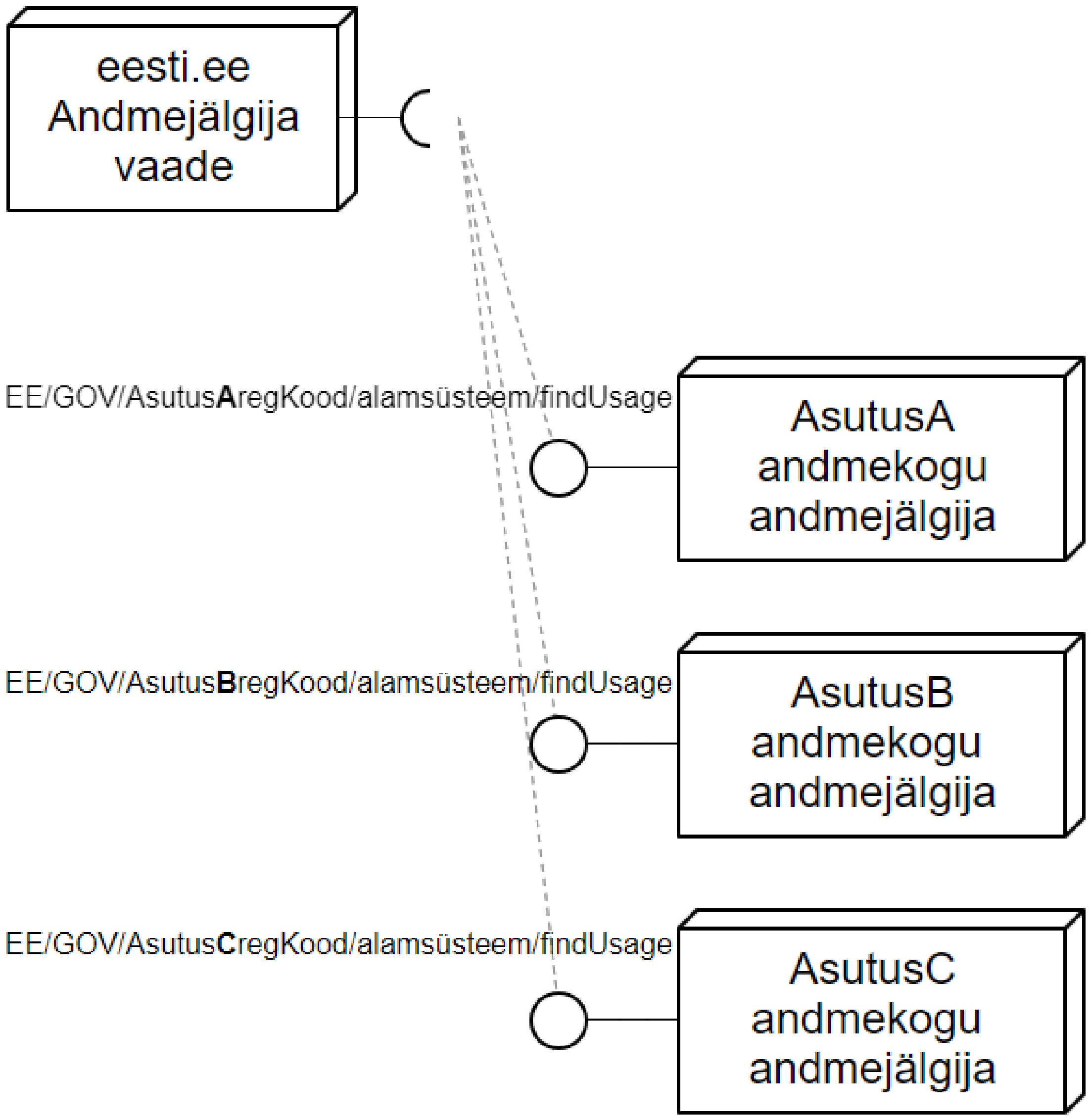
usagePeriod

The time period for which usage information can be requested.[2]

heartbeat

Requesting the availability status of the tracker's usage information.[2]

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


eesti.ee Andmejälgija vaade
EE/GOV/AsutusAregKood/alamsüsteem/findUsage
AsutusA andmekogu andmejälgija
EE/GOV/AsutusBregKood/alamsüsteem/findUsage
AsutusB andmekogu andmejälgija
EE/GOV/AsutusCregKood/alamsüsteem/findUsage
AsutusC andmekogu andmejälgija

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 noticable 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 Andmejälgija protocol, and it states that providing poor quality explanations for data access is a bad practice.[3]

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 eesti.ee	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:

[3] https://www.ria.ee/sites/default/files/documents/2022-11/Soovitusi-Andmejalgija-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 Digital Registry (Digiregistratuur)

The Digital Registry demonstrates exemplary implementation of the Andmejälgija protocol. They provide very descriptive information about data access requests, making it clear to users why their data was accessed and by whom.

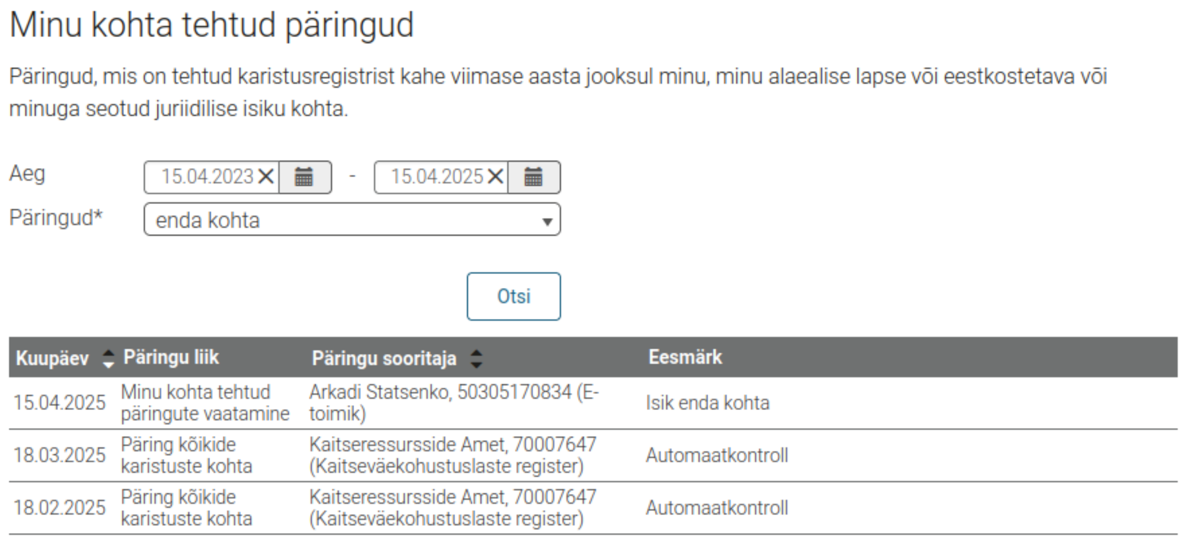
3.1.2 Other databases implementing Andmejälgija

- 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)
- Prescription Centre (Retseptikeskus)
- Social Protection Information System (Sotsiaalkaitse infosusteem)
- Social Services and Benefits Register (Sotsiaalteenuste ja toetuste register)
- Labour Inspectorate Working Life Information System (Tööinspektsiooni tooelu infosusteem)
- Unemployment Insurance Database (Töötuskindlustuse andmekogu)

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	15.04.2023 - 15.04.2025
Päringud*	enda kohta

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

3.3 Databases not providing any kind of data access tracking

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

4. Raw access logs vs Andmejälgija

5. Implementation

5.1 Description

The main objective of this thesis has been to develop a solution that would notify the user about new entries in Data Tracker.

5.2 Discussion of implementation choice

There is a number of possible ways to develop such a solution. In this section I will discuss different implementation options, the advantages and disadvantages of each, as well as why I eventually decided to settle on creating an Android application.

5.2.1 X-Road service

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

- 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
- 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. HSM costs typically around 10000€. Renting a production ready Security server from Telia costs 210€ per month [4]

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.

[4] https://www.telia.ee/ari/it-teenused/serverid-ja-pilv/x-tee-turvaserver/

5.2.2 Standalone approach

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

The main advantage of this approach is the abscence 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 app would be the most optimal approach. The app would keep the eesti.ee session alive and poll the Data Tracker API. This approach would combine the ease of setting up and use with solution remaining standalone, without a central server.

5.3 Development

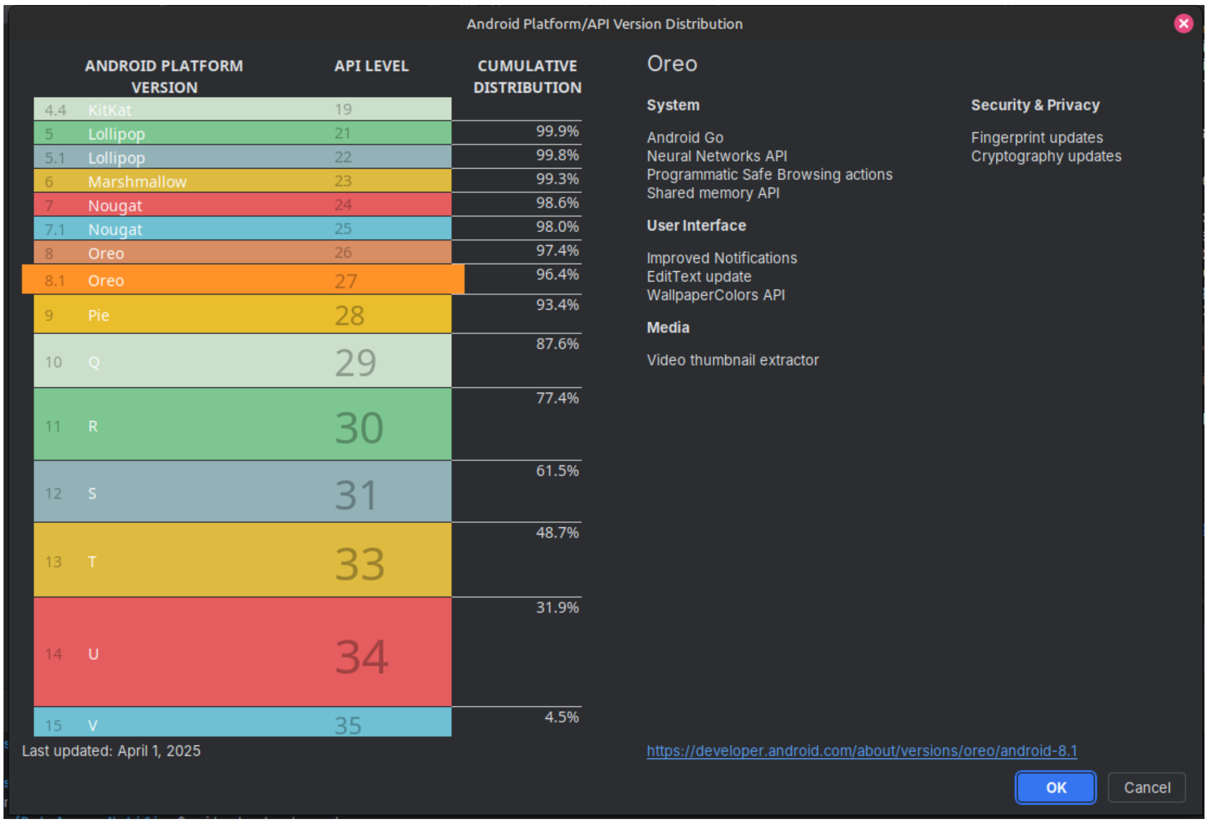
First of all I had to determine how feasible and reliable the standalone approach would be, especially as I was dealing with user-facing service that I would have to effectively reverse-engineer. The main challenge was to keep the session alive for as long as possible,

The eesti.ee session is established using one of strong authentication methods via TARA authentication 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. Once the user is authenticated, certain internal API endpoints become available, including those for refreshing JWT token and getting Data Tracker access logs. The JWT refresh endpoint returns Set-Cookie headers that include an updated JWTTOKEN on success, with updated expiration date (30 minutes from the moment of extention).

In order to test whether the session could be kept alive indefinitely I used Tab Reloader extention for Firefox by James Fray [5]. Using the browser extention I set up browser to auto reload tab with JWT token refresh endpoint. After 8 hours of experiment I deemed it a success.

Next step was the development of Android application. I have chosen to develop it using Kotlin programming language due to it being a new standard for developing Android apps as well as it having more expressive syntax and promising faster development speed.

The app is compatible with all Android versions starting from Android 8.1, meaning that it would work on approximately 96.4% of Android devices according to Android Studio API version distribution chart.



Android Platform/API Version Distribution

ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.4	KitKat	19	
5	Lollipop	21	99.9%
5.1	Lollipop	22	99.8%
6	Marshmallow	23	99.3%
7	Nougat	24	98.6%
7.1	Nougat	25	98.0%
8	Oreo	26	97.4%
8.1	Oreo	27	96.4%
9	Pie	28	93.4%
10	Q	29	87.6%
11	R	30	77.4%
12	S	31	61.5%
13	T	33	48.7%
14	U	34	31.9%
15	V	35	4.5%

Last updated: April 1, 2025

Oreo

System

Android Go
Neural Networks API
Programmatic Safe Browsing actions
Shared memory API

User Interface

Improved Notifications
EditText update
WallpaperColors API

Media

Video thumbnail extractor

Security & Privacy

Fingerprint updates
Cryptography updates

https://developer.android.com/about/versions/oreo/android-8.1

The software is open-source, licenced under MIT licence.

At the moment 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, the author would like to encourage anybody who would deem this solution useful to port it to platforms other than Android.

[5] https://addons.mozilla.org/en-US/firefox/addon/tab-reloader/

5.3.1 Authentication

In order to get access to eesti.ee internal APIs, the used needs to be authenticated via TARA. An initial plan has been to reverse engineer the authentication flow and authenticate the user by sending raw http requests and sending back 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 app 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 app can access the necessary session cookies and JWT token to interact with the internal APIs.

5.3.2 Interacting with eesti.ee internal APIs

For interacting with eesti.ee internal API I used OkHttp library by Square, Inc. [6]. I have used 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/andmejalgija/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.

[6] https://square.github.io/okhttp/

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 in order 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 on responses.

5.3.3 Keeping the session alive

The API requests are authenticated using JWT token cookie, the lifetime of which is 30 minutes. This means that the app has to either run a continuous foreground service or be waken up every once in a while in order to request an updated JWT token and send requests to Data Tracker.

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

WorkManager

According to 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 that was the first approach that I have 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 reality things look differently.

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. And indeed, with shorter intervals of 5 minutes the session stays alive. This behavioral difference rendered the PeriodicWorkRequest approach with 15-minute intervals unsuitable for maintaining persistent sessions in the application context.

While OneTimeWorkRequest is meant 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 it's lifespan.This approach allows to set custom intervals shorter than 15 minutes, curcumventing the limitation imposed by PeriodicWorkRequest.

While OneTimeWorkRequest may sound like a good solution, WorkManager itself appeared to be rather unreliable for my use case after some testing. The core reason behind limitations to follow is that tasks scheduled by WorkManager are managed by the system and may be deffered if deemed needed by the Android system, for example in the low battery scenario, or even in other scenarios depending on the aggressivenes of the Android variant in regards to the restrictions imposed on the behaviour of the apps.

Foreground Service

Foreground service is one of the most reliable types of tasks the application can schedule. As opposed to background services, foreground services are considered to be of higher priority to the system and thus are not candidates for being killed when the system is low on memory or the phone is low on battery.

An important limitation has been introduced in the latest Android updates though. More specifically, starting from Android 15, he system places restrictions on how long certain foreground services are allowed to run while your app is in the background. Currently, this restriction only applies to dataSync and mediaProcessing foreground service type foreground services.

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

AlarmManager + dataSync foreground service

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 app is not running. By using AlarmManager to trigger the app at regular intervals (e.g., every 5 minutes), I ensure that the session renewal and data polling tasks are reliably executed.

When the alarm fires, the app starts a short-lived dataSync foreground service to perform the necessary network operations: refreshing the JWT token and querying the Data Tracker 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 app 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 on low battery, ensuring that the session remains alive and notifications are delivered promptly to the user.

5.4 User guide

5.5 Software distribution

5.6 Known problems

6. Conclusion

7. Discussion

References

[1] Information System Authority (RIA). Andmejälgija / Data Tracker. Version 1.0.3. Apr. 15, 2025. https://github.com/e-gov/AJ.

[2] 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.

License