TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Gunnar Joosep Hint
164237IAPB

KASUTAJATE ANALÜÜTIKASÜSTEEM ANDROID RAKENDUSE PÕHJAL

Bakalaurusetöö

Juhendaja: Tarvo Treier
Magister MSc

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Gunnar Joosep Hint

19.05.2019

Annotatsioon

Antud lõputöö eesmärgiks oli arendada välja universaalne analüütikalahendus. Analüütika allikaks põhineks Android-rakendused, kuid süsteem peaks olema liidestatav ka teiste allikatega, nagu näiteks iOS rakendused, veebibrauserid või Asjade Interneti seadmed.

Lõputöö realisatsioon põhines viie suurema komponendi arendusest – Android testrakendus, klientrakenduse SDK, veebipõhine kasutajaliides, REST-teenus ja andmebaas. Kõikide komponentide arenduses jälgiti põhimõtteid, et iga komponent oleks eraldi skaleeritav ja universaalne.

Realisatsiooni tulemusel tekkis analüütikasüsteem, kus on võimalik näha statistikat reaalaaja kohta, kasutajate jäävuse kohta, eesmärgiteekondade kohta ja ka ajaloolist statistikat sündmuste kohta. Süsteem pakub ka lihtsat liidestust Android-rakendust, kuid tänu REST-teenuse universaalsele arhitektuurile, on võimalik liidestada sisuliselt kõiki teenuseid ja seadmeid, mis on ühendatud internetiga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 6 peatükki, 30 joonist, 3 tabelit.

Abstract

Users Analytics System Based on Android Application

The aim of this thesis was to create a universal analytics tool for online services and devices. The analytics for this thesis was constrained to Android applications, but the system should be designed in a way that any service could integrate with it if needed.

The realization of this thesis was based on five main components – Android application, client SDK, web-based user interface, REST-service and a database. These components were designed with the idea that each component would be scalable and universal.

The end product was an analytics tool which showed live user statistics, historical statistics, funnels and user retention. Android applications are supported out of the box with the client SDK, but thanks to the universal architecture of the REST-service, any tool or service, which has access to the internet, could be integrated with this system.

The thesis is in Estonian language and contains 22 pages of text, 6 chapters, 30 figures, 3 tables.

Lühendite ja mõistete sõnastik

Anis	Süsteemi nimetus lõppkasutaja jaoks
SDK	*Software development kit,* Tarkvaraarenduskomplekt
REST	*REpresentational State Transfer*, veebiressurside liides
Lõppkasutaja	Rakenduse haldaja, kes kasutab süsteemi oma rakenduses
Klient	Rakenduse kasutaja, kelle pealt kogutakse analüütilist informatsiooni
UUID	*Universal Unique Identifier*, 128-bitine unikaalne identifikaator
API	*Application Programming Interface*, rakendustarkvara liides
JPA	*Java Persistence API*, Java andmebaasiliidestuse teek
AWT	*Abstract Window Toolkit*, Javal põhinev graafilise liidese ehitamise teek
Funnel	Eesmärgiteekond
JWT, JWToken	*Json Web Token*, krüptitud andmete edastusviis kahe osapoole vahel
Docker	Teostab virtualiseerimist operatsioonisüsteemi tasemel ehk konteineriseerimist

Sisukord

Jooniste loetelu 7
Tabelite loetelu 9
1 Sissejuhatus 10
1.1 Ülesehitus 11
2 Olemasolevad lahendused 12
2.1 Integreerimine rakendusega 13
2.2 Püstitatud nõuded 15
3 Analüüsi olemus 16
3.1 Analüüsi selgitused 17
4 Realisatsioon 18
4.1 Süsteemi arhitektuur 18
4.2 Klientrakenduse SDK 19
4.2.1 Rakendusega liidestamine 20
4.2.2 Sündmuste saatmine 21
4.2.3 Kasutamine reaalses rakenduses 22
4.3 REST-teenus 22
4.3.1 Testimine 25
4.4 Andmebaasisüsteem 25
4.4.1 Tabelite ülesehitus 26
4.5 Kasutajaliides 27
4.6 Autentimisteenus 28
5 Tulemused 29
5.1 GDPR nõuetele vastavus 30
6 Kokkuvõte 31
Kasutatud kirjandus 32
Lisa 1 – Kasutajaliidese vaated 33

Jooniste loetelu

Joonis 1. Firebase integreerimiseks vajalik kood [5] 13
Joonis 2. Firebase sündmuste saatmise koodinäide [6] 14
Joonis 3. MixPaneli jaoks vajalik koodirida build.gradle failis [7] 14
Joonis 4. MixPaneli jaoks vajalik kood AndroidManifest.xml failis [7] 14
Joonis 5. MixPaneli jaoks vajalik kood Javas [7] 14
Joonis 6. MixPaneli sündmuse saatmise koodinäide [7] 14
Joonis 7 Süsteemi komponentide diagramm 19
Joonis 8. Rakenduse jaoks vajalik build.gradle koodirida 20
Joonis 9. Anis SDK moodulite käivitamise käsk 20
Joonis 10. Analüütika käivitamiseks vajalik kood 21
Joonis 11. SDK annotatsioonide koodinäide 21
Joonis 12. SDK koodis välja kutsumiste koodinäide 22
Joonis 13. Sündmuse lisamise näide 23
Joonis 14. Sündmust iseloomustava parameetrite näited 23
Joonis 15. Sündmuste pärimise näide 24
Joonis 16. REST-teenuse käivitamise käsk 24
Joonis 17. Andmebaasi käivitamise käsk 26
Joonis 18. Andmebaasi relatsiooniline mudel 26
Joonis 19. Veebiliidese käivitamiskäsud 28
Joonis 20. Lõppkasutaja vooskeem 29
Joonis 21. Esilehe vaade 33
Joonis 22. Sisselogmislehe vaade 33
Joonis 23. Rakenduste haldamise vaade 34
Joonis 24. Hetke statistika vaade 34
Joonis 25. Ajaloolise statistika vaade 35
Joonis 26. Kasutajate jäävuse vaade 35
Joonis 27. Eesmärgiteekondade haldamise vaade 36
Joonis 28. Eesmärgiteekonna lisamise vaade 36
Joonis 29. Eesmärgiteekonna graafiline vaade 37

Joonis 30. Rakenduse seadete vaade 37

Tabelite loetelu

Tabel 1. Firebase ja MixPaneli võimalustemaatriks [1], [4] 13
Tabel 2. Sündmuste väljasaatmise loogika..... 20
Tabel 3. Testide kattuvus teenusklassidel..... 30

1 Sissejuhatus

Tänapäeval on äärmiselt oluliseks saanud firmadele teadmine, mis toimub nende poolt pakutavate teenustes. Kui pakutav teenus on disainitud halvasti või esineb selle muid tehnilisi probleeme, siis on oluline nendest teada saada võimalikult kiiresti.

Teenustest paremini mõistmine ja selle probleemidest kiiremini teada saamine võimaldab firmadel püsida konkurentsivõimelisemad ja olla rohkem teadlikumad klientide soovidest. Selle tõttu ongi vaja koguda andmeid teenuste või seadmete kasutamise kohta, et saaks täpsemat informatsiooni klientide tegevuse kohta.

Hetkel on olemas juba head lahendused sellise analüüsi teostuse tegemiseks, kuid need on üldiselt tasulised ja neid ei saa muuta vastavalt kindla süsteemi vajadustele. Lisaks on need enamasti suunatud kas veebi- või mobiilirakendustele.

Antud lõputöö eesmärgiks on luua avalik teenus, mis pakuks universaalset kasutajate analüüsi. Teenus oleks suunatud arendajatele ja müügiinimestele, et analüüsida teenuse edukust. Antud lõputöö raames on loodava süsteemi analüüsi osa piiratud Android rakendustele.

Esimeseks alamülesandeks on tausta uurimine, ehk selgitamine, millised on olemasolevad lahendused ja mida on vaja analüüsida, et sellest oleks ka kasu. Lisaks uuritakse ka olemasolevate lahenduste tehnilist poolt, et välja valida nende tugevaid külgi.

Teiseks alamülesandeks on süsteemi arhitektuuri paika panemine ja selle realiseerimine lähtudes analüüsi tulemustest ja paika pandud tehnilistest nõuetest.

Kolmandaks alamülesandeks on tekkinud süsteemi valideerimine, et kas arhitektuuri valikud olid õigesti tehtud ja kas see ikka vastab püstitatud tehnilistele nõuetele.

Töö koosneb neljast sisulisest peatükist peale sissejuhatuse ja kokkuvõtte.

1.1 Ülesehitus

Antud töö teises peatükis uuritakse erinevaid olemasolevaid lahendusi, nende eeliseid ja nende puudujääke. Lisaks uuritakse ka, milliseid tugevaid külgi saaks võtta arendamisel kasutusse ja milliseid nõrkuskohti peaks vältima.

Seejärel kolmandas peatükis uuritakse analüüsi olemust ja miks seda vaja on. Uuritakse, kellel seda vaja on, ning mis on analüüsi peamine mõte. Lisaks selgitatakse ka erinevaid analüüsimeetrikaid ja nende olulisust.

Töö neljandaks peatükiks, ehk põhiosaks on realisatsiooni arendus, mis on jaotatud viieks põhiliseks osaks: testrakenduse arendus, klientrakenduse SDK arendus, REST-teenuse ja andmebaasi ehitamine ja kasutajaliidese arendus.

Seejärel viiendas peatükis uuritakse valminud süsteemi ja valideeritakse, kas valminud süsteem vastab püstitatud tehnilistele nõuetele. Lisaks kontrollitakse tekkinud süsteemi vastavust GDPR nõuetele.

2 Olemasolevad lahendused

Selles peatükis uuritakse olemasolevaid lahendusi, nende eeliseid ja nõrkusi. Seda tehakse selleks, et kasutada ära olemasolevate süsteemide tugevaid külgi ja vältimaks nende nõrkuskohti.

Hetkeseisuga on tegelikkuses olemas juba mitmeid analüütilisi lahendusi sellise probleemi jaoks. Enimlevinud süsteemideks on Google Analytics, Firebase ja MixPanel. Google Analytics on suunatud veebile, kuid Firebase ja MixPanel on suunatud rohkem Android ja iOS rakenduste analüüsiks. Mõlema süsteemi põhimõte seisneb selles, et saad registreerida antud süsteemis oma kasutaja ja seejärel saad lisada endale rakendusi, mille kallal analüüsi teostada.

MixPanel pakub palju võimalusi analüüsiks, nagu näiteks eesmärgiteekonnad, kasutajate säilitamine, enda poolt defineeritud sündmuste saatmine (tasuta variandis ainult üks oma sündmus), reaalaja statistika, konversioonid. Kuigi MixPanelil on ka palju muid funktsioone, siis nende kasutamine nõuab juba tellimuse tegemist, mis algab 150 dollarist kuus. Lisaks on see analüütika funktsionaalsus piiratud ainult iOS ja Android rakendustele. [1]

Firebase pakub oma teenuses analüüsi osa tasuta. Sarnaselt MixPanelile pakub Firebase analüütikavõimalusi nagu reaalaja statistika, eesmärgiteekonnad, kasutajate säilitamine, segmenteerimine ning konversioonid. Lisaks saab ka Firebase abil saata oma sündmusi ning need on piiratud rohkematele sündmusele võrreldes MixPaneliga, lubades 500 erinevat enda poolt defineeritud sündmust. Lisaks on erinevus toetatud platvormides, sest peale Androidi ja iOSi toetab Firebase ka veebibrausereid. [2], [3]

Võrreldes MixPaneliga pakub Firebase sarnast funktsionaalsust, kuid tasuta. Tabelil 1 on välja toodud võimalustemaatriks erinevate teenuste vahel. Tärniga märgistatud plussid tähistavad tasulisi funktsioone.

Funktsioon	Firebase	MixPanel	Kommentaarid
Reaalaja analüütika	+	+	Kahe süsteemi puhul näidatakse reaalajas analüütikat erinevalt
Eesmärgiteekonnad	+	+	
Enda poolt defineeritud sündmused	< 500tk	1tk	Tasulistes variantides saab arvuliselt rohkem jälgida
Konversioonid	+	+*	
Kasutajate jäävus	+	+	
Segmendid	+	+	Firebase puhul kui *Audiences*
Ennustamine	+	+*	
Anomaaliate tuvastus	+	+*	

Tabel 1. Firebase ja MixPaneli võimalustemaatriks [1], [4]

Pakutud võimalustepagasi poolest on Firebase ja MixPanel väga sarnased. Nende suureks miinuseks on see, et need teenused on piiratud täielikult mobiilirakendustele ja veebi. Kui arvestada seda, et asjade internet on muutumas üha populaarsemaks, siis võiks arvata, et nendele analüütika toe pakkumine on samamoodi muutumas üha olulisemaks.

2.1 Integreerimine rakendusega

Kuna Firebase'i ja Androidi puhul on tegu Google poolt pakutud teenustega, on integreerimine Android rakendustega tehtud väga lihtsaks. Selle jaoks on vaja luua Firebase konsoolis uus projekt, kopeerida sealt antav *google-services.json* fail projekti kausta ning lisada projekti ja *app* kaustas olevatele *build.gradle* failidele koodiread sektsiooni *dependencies*, mis on kirjeldatud joonisel 1. [5]

```
Projekti build.gradle:
classpath 'com.google.gms:google-services:4.2.0'

app/build.gradle:
implementation 'com.google.firebase:firebase-core:16.0.9'
```

Joonis 1. Firebase integreerimiseks vajalik kood [5]

Seejärel saab hakata saatma rakenduse koodis sündmusi. Sündmuste saatmise näide on välja toodud joonisel 2.

```
Bundle params = new Bundle();
params.putString("image_name", name);
params.putString("full_text", text);
mFirebaseAnalytics.logEvent("share_image", params);
```

Joonis 2. Firebase sündmuste saatmise koodinäide [6]

MixPaneli integreerimiseks on vaja samuti projekti *build.gradle* faili uuendada koodiga, mis on kirjeldatud joonisel 3. Seejärel on vaja lisada ka *AndroidManifest.xml* faili rakenduse õiguste jaoks koodiread, mis on kirjeldatud joonisel 4. Viimase sammuna on vaja lisada Java koodi MixPaneli initisialiseerimiseks kood, mis on kirjeldatud joonisel 5. [7]

```
implementation 'com.mixpanel.android:mixpanel-android:5.+'
```

Joonis 3. MixPaneli jaoks vajalik koodirida build.gradle failis [7]

```
<uses-permission
  android:name="android.permission.INTERNET" />

<uses-permission
  android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission
  android:name="android.permission.BLUETOOTH" />
```

Joonis 4. MixPaneli jaoks vajalik kood AndroidManifest.xml failis [7]

```
public static final String MIXPANEL_TOKEN = "YOUR_TOKEN";

// Initialize the library with your
// Mixpanel project token, MIXPANEL_TOKEN, and a reference
// to your application context.
MixpanelAPI mixpanel =
    MixpanelAPI.getInstance(context, MIXPANEL_TOKEN);
```

Joonis 5. MixPaneli jaoks vajalik kood Javas [7]

Seejärel saab MixPanelisse hakata saatma sündmusi. MixPaneli sündmuste saatmise koodinäide on välja toodud joonisel 6.

```
MixpanelAPI mixpanel =
      MixpanelAPI.getInstance(context, MIXPANEL_TOKEN);

JSONObject props = new JSONObject();
props.put("Gender", "Female");
props.put("Plan", "Premium");

mixpanel.track("Plan Selected", props);
```

Joonis 6. MixPaneli sündmuse saatmise koodinäide [7]

Vaadates integreerimist ja kasutust, siis võib öelda, et MixPanel ja Firebase on klient-SDK loomu poolest sarnased. Sündmuste saatmine on väga sarnane, kuid erinevus seisneb SDK integratsioonis. MixPaneli liidestamine nõuab rohkem samme ja on keerukam, kui Firebase'i lahendus.

2.2 Püstitatud nõuded

Võttes inspiratsiooni eelnevalt kirjeldatud süsteemidest, valis autor analüüsi tegemise jaoks eesmärgiteekonnad, oma sündmuste saatmine, kasutajate säilitamise mõõtmine ja reaalajas jälgimine. Nende meetrikate täpsemad selgitused on välja toodud peatükis 3.1.

Eesmärgiteekonnad said valitud, kuna need on olulised teenuse eesmärkide mõõtmisel ja nendega on teoreetiliselt võimalik ka ära katta konversioonid. Oma sündmuste saatmine on üks tähtsamaid võimalusi sellise süsteemi arendamisel, kuna nendest algab kogu analüüs. Reaalajas jälgimine on juba sellepärast kasulik, et lõppkasutajatel on hea jälgida ning testida reaalajas oma integratsiooni. Lisaks on kasutajate säilitamine oluline meetrika peaaegu iga teenuse analüüsis.

Kuna andmeanalüüsi teenus hõlmab endas suure hulga andmete vastuvõtmist ja töötlemist, siis peab olema antud süsteem skaleeritav iga komponendi ulatuses, et saaks vastavalt vajadusele eri komponente teha võimsamaks. Antud süsteem peab olema ka testitav ning kui reaalsete andmemahtudega testida ei saa, siis peavad põhilisemad andmeanalüütikat tegevad osad olema testitud.

Avaliku teenusena peab olema süsteem ka autenditav ning andmed peaksid jääma konfidentsiaalseks igale lõppkasutajale. Lisaks ei tohi see süsteem mõjutada ka liidestavate teenuste tööd, nagu näiteks võrguühenduse liigne koormamine, kui liidestatav teenus sõltub internetiühenduse kiirusest.

Süsteemi universaalsuse tagamiseks peaks olema see ehitatud niiviisi, et see oleks liidestatav võimalikult paljude süsteemidega. Näiteks Android rakenduste puhul peaks loodav SDK toetama ka vanemaid operatsioonisüsteeme.

Klientrakenduse SDK arendamisel sai püstitatud ka nõue, et selle integratsioon reaalse rakendusega oleks sarnaselt lihtne MixPaneli ja Firebase'i poolt pakutavate variantidega.

3 Analüüsi olemus

Selles peatükis uuritakse analüüsi olemust, ehk kuidas analüüsida süsteeme ja mida erinevad väljendid tähendavad. Lisaks vaadatakse, mida tänapäeval analüüsitakse ja mida tähendavad erinevad meetrikad analüüsis.

Tänapäeval on analüütika saanud osaks suuremast süsteemist. Süsteemi disainimisel juba peab arvestama analüütikat kui selle osa, mitte kui järelmõtet. Selleks, et analüütika toimiks, peab arvestama ka süsteemiks vajaminevate andmetega. [8]

Ühe lihtsa näitena oleks e-poodide puhul oluline jälgida ostmiste hulka, müüdavate esemete osakaale ja klientide geograafilisi asukohti. Selliste kriteeriumite põhjal oleks näiteks võimalik optimiseerida toodete tarneaegu ja tellitavate toodete hulkasid ning esitada toodete soovitusi klientidele.

Samas ei ole ka mõtet kõiki statistilisi sündmusi talletada, kuna suurema andmehulga hoiustamine ja analüüsimine nõuab suuremat riistvaralist kulu.

Näiteks koguti mõned aastat tagasi veebilehe päringuid, millega mõõdeti külastatavuse arvu. Tänapäeval tehakse juba ühe lehe laadimisel lisameediat ja pilte, mis võib tähendada juba 25 päringut serveri poole. Selle järgi statistikat pidada on väga ressursimahukas ning selle põhjal saadav informatsioon ei ole väga pädev. [9]

Lisaks saab tuua ka muid näiteid, nagu veebilehe kaasamine ehk sessioonide arv. Selle alusel saab lihtsalt arvata, kas inimesed ei leia, mida tahavad, või perfektse süsteemi puhul saavad kohe kõik, mis vaja. Veel oleks tarbetu koguda lihtsalt peamisi lehelt väljumise vaateid, kuna paljude veebipoodide puhul võivad inimesed enamus leida kohe toote, mida otsivad ja selle osta ning lahkudagi lehelt. [9]

Erinevalt eelmistest näidetest, peaks analüüsima hoopis süsteemi kasutajate peamist lõppeesmärki, põhiprotsesside lõpetamise edukust, kasutajate segmenteerimist näiteks asukoha või muude parameetrite alusel. Lisaks tuleks mõelda analüüsi tegemisel, kellele ja milleks seda vaja on. Analüüsimise peamine mõte on ikkagi teada saada, mida toote kliendid tahavad, mitte toote omanikud. [9]

3.1 Analüüsi selgitused

Tänapäeval kogutakse palju erinevaid meetrikaid, mille nimetused ei pruugi alati selgitada, mis need täpsemalt teevad, ja miks need kasulikud on. Järgnevalt on lahti seletatud erinevad meetrikad ja nende kasulikkus.

Konversioonid näitavad, kui suur osa kasutajatest on täitnud teatud kindlat eesmärki. Näiteks on selle puhul hea mõõta, kui palju kasutajatest on hakanud kasutama mõnda uut teenust või tellimust.

Eesmärgiteekonnad tähistavad kindlate sammude järjekorda, mis näitavad, mitu protsenti kasutajatest on jõudnud igal sammul eesmärgini lähemale. Antud meetrika on mõeldud selleks, et teada saada, kuhu võivad kasutajad seisma jääda enne lõppeesmärgi saavutamist. [10] Näiteks võib lugeda eesmärgiteekonnaks ostukeskkonda sissselogimist, toodete vaatamist, toote lisamist ostukorvi ja seejärel ostu sooritamist.

Kasutajate jäävus näitab seda, mitu protsenti kasutajatest on jäänud teatud aja jooksul teenust kasutama. [11] See on hea, et teada saada kasutajate suunitluse kohta. Näiteks veebipoe puhul on arusaadav, et kasutajate jäävus on väike, kuid foorumite puhul on see oluline, et peale esimest korda külastamist tuleksid kasutajad ka tulevikus uuesti tagasi.

Segmentide abil on võimalik grupeerida kasutajaid teatud parameetrite alusel, mis võimaldab analüüsida kindlat kasutajate gruppi. Selle abil on näiteks võimalik uurida kasutajaid eraldi nende vanuse, soo, asukoha või mõne muu omaduse järgi. [12]

4 Realisatsioon

Selles peatükis on ära kirjeldatud realisatsiooni arhitektuur ja selle komponendid. Eraldi on välja toodud iga komponent ning selle kommunikatsioon süsteemi teiste osadega. Lisaks on kirjeldatud ka funktsioone ning ressursse, et lõppkasutajad saaksid liidestada süsteemi ka oma lahendustega.

4.1 Süsteemi arhitektuur

Klientrakenduse ja REST-teenuse suhtluses on ära kirjeldatud kaks põhiobjekti:

1. Sündmus (ingl. k *Event*) – rakenduse poolt välja saadetav analüütika info. Näiteks rakenduse avamise sündmus, registreerimise sündmus, ostu õnnestumise/mitteõnnestumise sündmus

2. Rakendus (ingl. k *Application*) – rakendus, teenus või seade, mille kohta kogutakse kokku andmeid

Süsteemi nimetus lõppkasutajale on nimetatud Anis. Arhitektuuriliselt jaguneb see viieks suuremaks komponendiks:

1. Klientrakenduse SDK

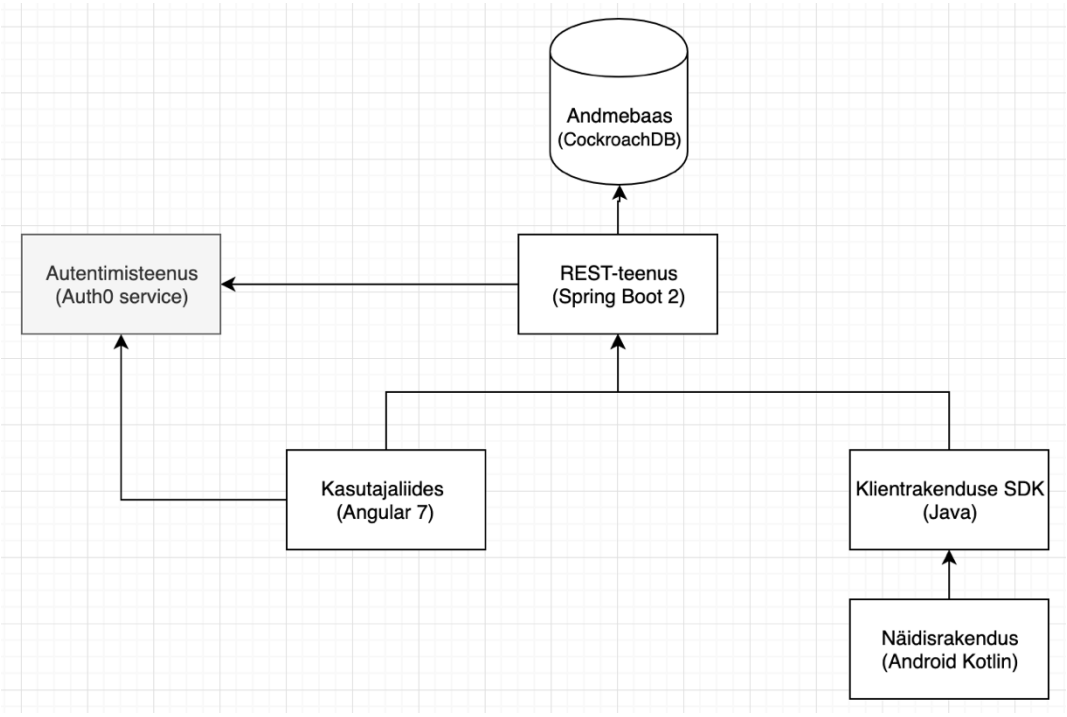
2. REST-teenus

3. Andmebaas

4. Kasutajaliides

5. Autentimisteenus

Joonisel 7 on näha, kuidas on seotud omavahel erinevad süsteemi komponendid. Erinevate osade selgitused ja nendevaheline suhtlus on kirjeldatud täpsemalt järgmistel peatükkidel. Joonisel on eraldi märgitud autentimisteenus, kuna tegu on välise teenusega, millega ülejäänud süsteemi osad lihtsalt suhtlevad.



Joonis 7 Süsteemi komponentide diagramm

4.2 Klientrakenduse SDK

Klientrakenduse SDK on üles ehitatud kahe eraldi mooduli peale – anis-core ja anis-android. Mõlemad moodulid on üles ehitatud Java 7 keeletaseme peale selleks, et hoida tagasiulatuvat ühilduvust ka vanemate Android seadmetega. Koodi jaotamine kahe mooduli peale oli tehtud selleks, et tulevikus oleks lihtsam laieneda ka muude Javal põhinevate süsteemide või rakenduste peale nagu Spring Boot või Java AWT.

Anis-core vastutab sündmuste kokku kogumise kirjeldamise ja sündmuste välja saatmisega. Lisaks võimaldab see moodul algatada uut instantsi, et hakata sündmusi koguma.

See moodul hoiab ka abifunktsioone, et lihtsustada sündmuste kokku kogumist. Sündmuste haldus toimib sellisel põhimõttel, et sündmused kogutakse kokku, salvestatakse kohalikku seadmesse ja nende välja saatmine toimub eraldi klassis, mis toimib lihtsalt algoritmil, mis sõltub seadme internetiühenduse kiirusest.

Antud algoritm toimib põhimõttel, et mida väiksem on internetikiirus, seda harvemini ja väiksemas koguses saadetakse sündmusi välja. Algoritmi olemus on selgitatud tabelil 2. Tabelil on näha, mitu sündmust saadetakse välja, arvestades, kui kaua aega kulus eelmiste

sündmuste saatmiseks. Esimesel korral saadetakse lihtsalt üks sündmus välja, mille põhjal hakatakse järgmistel kordadel aega mõõtma.

Eelnevalt kulunud aeg	**Saadetavate sündmuste maks. hulk**
< 100ms	40
< 300ms	30
< 500ms	20
< 1s	10
< 1.5s	3
> 1.5s	1

Tabel 2. Sündmuste väljasaatmise loogika

Anis-android moodul hoiustab endas Android-spetsiifilisi sündmuse kogumise klasse. See moodul implementeerib sündmuste kokku kogumist. Selle jaoks kasutab see AspectJ moodulit, et vähendada üleliigse koodi kirjutamist.

AspectJ võimaldab kirjutada abifunktsioone, mis kuulavad Java meetodite käivitumist, et enne või pärast neid käivitada oma koodi. Tänu AspectJ annotatsioonide kasutamisega on automaatselt võimaldatud erinevate vaadete, seadme informatsiooni ja kasutaja asukoha jälgimine.

4.2.1 Rakendusega liidestamine

Selleks, et kasutada seda teeki Android rakendustes, on vaja lisada rida rakenduse peakaustas olevale *build.gradle* failis sektsiooni *buildscript* alasektsioonis *dependencies*, mis on näidatud joonisel 8. Antud koodirida võimaldab kasutada AspectJ abil tekitatud annotatsiooniprotsessoreid.

```
classpath 'com.hujiang.aspectjx:gradle-android-plugin-aspectjx:2.0.4'
```

Joonis 8. Rakenduse jaoks vajalik build.gradle koodirida

Lisaks on vaja rakenduse *app/libs* kausta lisada anis-core.jar ja anis-android.aar failid, mis on genereeritud anis-core ja anis-android moodulite poolt. Neid saab genereerida minnes vastavate moodulite kaustadesse, mis asuvad anis-sdk kaustas ja käivitades mõlema puhul käsku kujutatud joonisel 9.

```
./gradlew build
```

Joonis 9. Anis SDK moodulite käivitamise käsk

Anis-android mooduli pakendamise jaoks on vajalik Android SDK olemasolu. Reaalse teenuse pakkumise puhul saaks need üles laadida Maven repositooriumisse, et lõppkasutaja ei peaks neid mooduleid käsitsi kompileerima.

Seejärel on vaja käivitada Anis instants, mis hakkab suhtlema REST-teenusega, mis on näidatud joonisel 10. See tuleb käivitada Android rakenduse puhul mõnes klassis, mis laiendab klassi *Activity*.

```
Anis.builder()
    .setApiId("apiId")
    .setApiSecret("secret")
    .setEventDataSource(LocalStorageDataSource(this))
    .startInstance()
```

Joonis 10. Analüütika käivitamiseks vajalik kood

Väärtused *ApiId* ja *secret* saab teada rakenduse lisamisel kasutajaliidesest, mis on kirjeldatud peatükis 4.5.

4.2.2 Sündmuste saatmine

Selleks, et saata sündmusi, on võimalik kasutada kahte erinevat viisi: annotatsioonid ja koodis välja kutsumine. Nende kohta on toodud näited järgnevates lõikudes.

Annotatsioonidega koodikasutuse näide on joonisel 11. Annotatsioonide puhul saab kasutada mitut erinevat kuju, mis täidab osa nõutud sündmuse infot automaatselt ära. Täidetav informatsioon on aru saada annotatsiooni pealkirja järgi. Igat annotatsiooni saab panna meetodite ja klasside peale.

```
@SendEventPurchase(true or false) // true if puchcase was successful
void method() {...}

@SendEvent("EVENT_NAME") // EVENT_NAME can be any string
void method() {...}
```

Joonis 11. SDK annotatsioonide koodinäide

Tavalise koodi välja kutsumise näited on joonisel 12. Koodis välja kutsumise eelis on see, et saab jooksvalt anda kaasa igale sündmusele parameetreid ja enda poolt loodud sündmuse liike, mis annab suure paindlikkuse sündmuste saatmise jaoks.

Lisaks oli loodud mitu abifunktsiooni selleks, et lihtsustada koodis välja kutsumise loogikat. Joonisel 12 on samuti välja toodud sisse logimise alustamise ja lõpetamise saatmise loogika, ostu alustamise ja lõpetamise loogika.

```
AnisProcessor.sendEvent("EVENT_NAME", Map<String, Object> properties);

AnisProcessor.sendLoginStartEvent();

AnisProcessor.sendLoginFinishEvent();

AnisProcessor.sendPurchaseStartEvent();

AnisProcessor.sendPurchaseFinishEvent();
```

Joonis 12. SDK koodis välja kutsumiste koodinäide

Iga loodava sündmuse külge tuleb panna sündmuse liik ehk *eventType* ja lisaks saab valikuliselt lisada ka erinevaid lisaandmeid *properties* kujutisse. Kujutise *properties* alla võib lõppkaustaja lisada endale vajaminevat informatsiooni, kuid osad väärtused on kindlalt ära määratletud, mille kohta on kirjeldatud peatükis 4.3.

4.2.3 Kasutamine reaalses rakenduses

Selleks, et kontrollida, kas loodav SDK oleks ikka ühilduv juba eksisteerivate rakendustega, sai valmis tehtud lihtne poerakendus, kuhu see integreerida. Poerakenduse funktsionaalsuseks on sisse logimine, registreerimine, toodete sirvimine, ostukärusse lisamine ja ostmine.

Rakendus oli puhtalt kasutajaliidese mõttes loodud, s.t kõik andmed olid võltsandmed ja reaalne infosüsteemi teenus sellele puudus. Rakenduse peamine mõte oli selleks, et sellega saaks ära katta ja testida põhilised sündmuste allikad, mida võiks lõppkasutajatel vaja minna – sisse logimine, registreerimine, ostu sooritamine.

4.3 REST-teenus

Antud lahenduse keskseks osaks on REST-teenus, mis suudab vastu võtta ja väljastada kasutajate poolt loodud sündmusi. Teenus on üles ehitatud Spring Boot 2 ja Kotlini peale ning kasutab andmete edastamiseks ja vastuvõtmiseks REST arhitektuuri. Andmebaasisuhtluseks kasutab see teenus JPA teeki.

Läbi selle teenuskihi käib läbi kogu infovahetus SDK, kasutajaliidese ja andmebaasi vahel. Iga loodav rakendus on seotud ühe kindla kasutajaga ja kõik sündmused on seotud kindla rakendusega.

Rakenduse loomisel küsitakse kasutajalt kahte välja: rakenduse nimi ja identifikaator. Lisaks määratakse rakenduse loomisel sellele ka salajane võti. Unikaalse identifikaatori ja salajase võtme alusel võtab vastu ja saadab teenus sündmusi. Süsteemis ning ka järgnevalt on need parameetrid kirjeldatud kui *apiId* ja *secret*.

Selleks, et saata oma sündmusi, tuleb teha POST päring, mis on kirjeldatud joonisel 13. Kui kõik andmed on õiged, siis tuleb vastuseks sõne "OK". Päringu päises olev väärtus *client* tähistab klienti, kelle alt sündmusi saadetakse. Selle väärtuseks võib olla ükskõik milline UUID. Android rakenduse puhul genereeritakse *client* parameeter esmasel käivitamisel igale kasutajale. Lisaks on vaja päringu päisesse lisada sõne *secret*, mis antakse rakenduse loomisel ja *applicationId*, mis on määratud kasutaja poolt. Seda sama ressurssi kasutatakse ka klientrakenduse SDK's, et saata sündmusi.

```
POST /api/events/{applicationId}
Header secret
Header client
Body:
[
  {
    "eventType": "string",
    "properties": {"key": "value"},
    "sessionId": "UUID string",
    "timestamp": "2019-05-08T00:00:00.000Z"
  }
]
Response: "OK"
```

Joonis 13. Sündmuse lisamise näide

Joonisel 14 on ära kirjeldatud erinevate *properties* kujutise väärtused, mis on süsteemis juba eelnevalt defineeritud.

```
Kasutatud seadmete informatsioon:
"manufacturer" : "Samsung"
"model"        : "Galaxy S6"
Kasutaja asukoht:
"latitude"     : "60.00"
"longitude"    : "60.00"
Rakenduse versioon:
"app_version"  : "v1.0"
Operatsioonisüsteemi versioon:
"os_version"   : "Android 5.0"
Vaate nimetus (eventType peab olema "VIEW_OPEN"):
"view_name"    : "ProductListActivity"
```

Joonis 14. Sündmust iseloomustava parameetrite näited

Oma sündmuste pärimiseks on võimalik teha GET päring, mis on kirjeldatud joonisel 15. Valikuliselt saab lisada ka parameetri *eventType*, mille alusel on võimalik sündmusi filtreerida tüübi alusel. Lisaks on vaja ka selle päringu päisesse lisada sõne *secret*.

```
GET /api/events/{applicationId}?eventType={eventType}
Header secret
Response:
[
  {
    "eventType": "string",
    "properties": {"key": "value"},
    "sessionId": "UUID string",
    "timestamp": "2019-05-08T00:00:00.000Z"
  }
]
```

Joonis 15. Sündmuste pärimise näide

Lisaks on ka muid REST-teenuse ressursse, mida kasutab sisemiselt kasutajaliides, et pärida erinevate meetrikate kohta andmeid.

Kasutajate autentimine on tehtud läbi välise teenuse Auth0, mis võimaldab väga mugavalt realiseerida kasutajate sisse logimise ja registreerimise. Kui kasutaja logib sisse või registreerib läbi kasutajaliidese, määratakse sellele kasutajale JWT võti, mille alusel verifitseeritakse kõik päringud. Sellega tagatakse, et iga rakendus ja sündmus oleks igale kasutajale privaatne.

Tänu valitud lahendustele on antud komponent ka skaleeritav, kuna kasutab olekuvabasid sessioone kasutajate ja sündmuste identifitseerimiseks, mistõttu saab seda rakendust püsti panna mitme isendina näiteks Kubernetes keskkonnas. Lisaks on olemas ka *Dockerfile*, mille abil on võimalik teenust konteineriseerida.

Teenuse käivitamiseks on vaja eelnevalt installeerida JDK 11 ja käivitada andmebaas kohalikus masinas, mille käivitamine on kirjeldatud peatükis 4.4. Kui andmebaas on käivitatud, siis piisab teenuse käivitamiseks käsust, mis on näidatud joonisel 16.

```
./gradlew bootRun
```

Joonis 16. REST-teenuse käivitamise käsk

4.3.1 Testimine

Selleks, et valideerida REST-teenuse toimimist, sai loodud testraamistik andmebaasioperatsioonide jaoks. Antud raamistik toetub Dockeri konteinersüsteemile, luues enne testide käivitamist tühi CockroachDB andmebaasi konteiner, mis täidetakse sündmustega ja valideeritakse, kas käivitatavad analüüsipäringud on korrektsed.

Konteinerite loomiseks sai kasutusele võetud Testcontainers teek, mis algselt toetas ainult Postgres andmebaasi. Kuna selle implementeerimise käigus selgus, et Postgres ja CockroachDB poolt kasutatavad SQL dialektid erinevad, siis ei saanud Postgres andmebaasi testide käivitamiseks kasutada. Selle tõttu sai loodud abiklassid, et antud teegi abil saaks kasutada ka CockroachDB andmebaasi.

4.4 Andmebaasisüsteem

Antud süsteem kasutab CockroachDB andmebaasilahendust. Antud lahendus on väga lähedane Postgres andmebaasiga, kuid pakub teatud eelisi, mida Postgres ei paku. Kuigi CockroachDB ei toeta veel kõiki SQL standardi võimalusi [13], on sellel siiski piisav tugi olemas antud projekti raames.

Võrreldes Postgres andmebaasiga, pakub CockroachDB võimalusi nagu automaatne skaleeritavus, ehk andmebaasi saab ära jaotada mitme sõlme vahel ning andmed on sõlmede vahel mitmekordselt ära varundatud. Lisaks võimaldab CockroachDB puuduvaid andmeid ka parandada sõlmede vahel. [14]

Postgres suudab ka skaleerida andmebaasi mitme sõlme vahel, kuid sellisel juhul peab jääma üks sõlmedest *master* andmebaasiks. Sellisel juhul, kui see sõlm läheb maha, läheb kogu sõlmestik maha, mistõttu peab hakkama ülejäänud sõlmi kohandama. [15]

Algne plaan andmebaasi jaoks oli MongoDB, kuid arendustegevuse käigus selgus, et dokumendipõhises andmebaasisüsteemis ei saa teha autori hinnangul nii keerukaid päringuid, kui saaks teha relatsioonilises andmebaasis. See erinevus tuli välja keerulisemate andmebaasipäringute koostamises, mida MongoDB puhul oleks olnud kordades keerulisem teha.

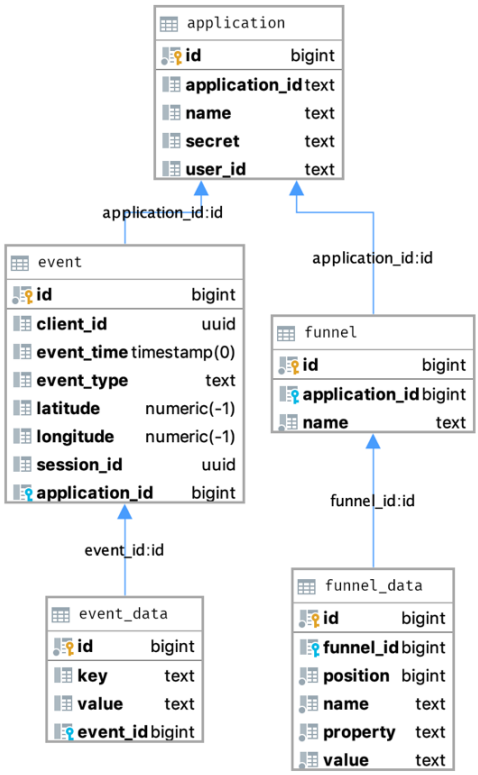
CockroachDB kohalikuks käivitamiseks on kõige mõistlikum kasutada konteinerite lahendust Docker. Selle abil piisab ühe käsu käivitamisest, mis on kirjeldatud joonisel 17.

```
docker run -d \
--name=roach1 \
--hostname=roach1 \
-p 26257:26257 -p 8081:8080  \
-v "${PWD}/cockroach-data:/cockroach/cockroach-data“
cockroachdb/cockroach:v2.1.6 start –insecure
```

Joonis 17. Andmebaasi käivitamise käsk

4.4.1 Tabelite ülesehitus

Andmebaasi sisuline osa on üles ehitatud kolmele põhitabelile – *application, event, funnel* ja kahele abitabelile – *event_data, funnel_data*. Joonisel 18 on kirjeldatud nende omavahelised seosed. Kõik tabelid on genereeritud REST-teenuse projektis olevate SQL failide alusel.



Joonis 18. Andmebaasi relatsiooniline mudel

Tabel *application* on mõeldud kasutajate poolt loodud rakenduste talletamiseks, *event* ja *event_data* tabelid talletavad kasutajate poolt loodud sündmusi ja nendega seotud lisainformatsiooni. Tabelid *funnel* ja *funnel_data* talletavad informatsiooni kasutajate poolt määratud eesmärgiteekondade kohta.

Lisaks on andmebaasis ka abitabel *flyway_schema_history,* mida kasutab REST-teenuse projektis olev teek *Flyway,* mis on mõeldud andmebaasi versiooni haldamiseks. See võimaldab väga lihtsalt uuendada juba olemasolevat toimivat andmebaasi.

4.5 Kasutajaliides

Selleks, et näha kogutud andmete põhjal analüütilist informatsiooni graafiliselt, sai loodud ka veebiliides. Selles liideses saab iga inimene sisse logida ja hallata oma rakendusi ning näha iga oma rakenduse kohta statistikat.

See on üles ehitatud ngx-admin malli peale ja kasutab Angular 7 raamistikku. Sellised tehnoloogiad said valitud, kuna antud malli peale oli väga lihtne ehitada haldusliides ja Angulari abil oli mugav luua erinevaid komponente ja teenusklasse.

Kasutajaliidese loomisel sai oluliseks põhimõte, et kõik vaated peaksid olema võimalikult lihtsad ja kergesti arusaadavad. Lisas 1 on näidatud ära, millised vaated näevad välja ja milline on rakenduse üldmulje.

Kasutajaliides on disainitud nii, et esmane sisselogimata vaade tutvustab lühidalt teenust, kus on ka nupp sisse logimiseks ja registreerimiseks. Pärast sisse logimist ilmub rakenduste nimekiri, kus saab ka uut rakendust lisada. Rakenduse lisamisel kuvatakse ka SDK liidestamiseks vajalikud parameetrid nimetustega *apiId* ja *secret*.

Rakenduse peale vajutades tuleb ette *Dashboard* vaade, kus on näha reaalaja statistika. Lisaks sellele ilmub ka vasakule ekraani menüüvaade, kus on võimalik liikuda edasi statistika, *retention*, *funnel* ja seadete vaadetele.

Dashboard vaade sisaldab endast hetkel aktiivsete kasutajate arvu, kasutajate asukoha kaarti, keskmist sessiooni aega ja päevast sündmuste arvu. Statistika vaates on võimalik näha peamisi kasutatud seadmeid, rakenduse versioonide jaotust ja operatsioonisüsteemide versioone. Lisaks on seal võimalik näha päevade kaupa kasutajate aktiivsust, vaadete aktiivust ja sündmuste aktiivust liigiti.

Retention vaates on võimalik näha kasutajate jäävust, ehk kui palju inimesi on jäänud kasutama rakendust 14 päeva jooksul. *Funnels* vaade sisaldab kasutajate endi poolt määratud eesmärgiteekondi, lisaks on selles vaates võimalik lisada uusi teekondi. Mõne eesmärgiteekonna peale vajutades, ilmub selle kohta informatiivne graafik.

Seadete vaates on võimalik näha rakendusega seotud tehnilist informatsiooni, muuta rakenduse salavõtit klient-SDK jaoks, sündmusi eemaldada ja kogu rakendust eemaldada.

Veebiliides on ühendatud REST-teenuse ja Auth0 teenusega. Enne autentimist kuvatakse lõppkasutajale lihtne tutvustusleht ja sisselogimisel suunatakse edasi Auth0 teenuse lehele, mis on visuaalselt tehtud sarnaseks ülejäänud leheküljega. Kasutajaliideses on kasutatud Auth0 teeki, mille abil on lihtne kätte saada Auth0 poolt antav JWToken, mis pannakse igal REST-teenuse päringul päisesse *Authorization* väljale.

Selleks, et käivitada kohalikult arendamiseks veebiliides, on vaja käivitada käsud, mis on kirjeldatud joonisel 19.

```
npm install
npm start
```

Joonis 19. Veebiliidese käivitamiskäsud

4.6 Autentimisteenus

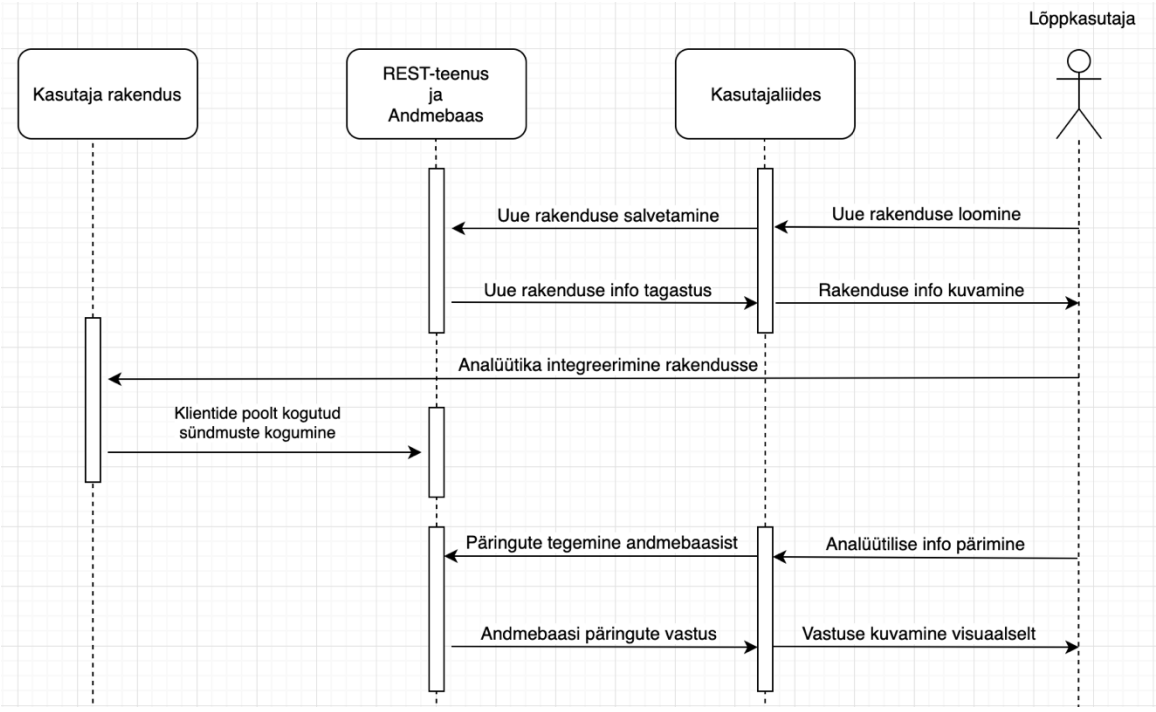
Selleks, et ei peaks tegelema eraldi kasutajate haldamise ja autentimisega, sai kasutusele võetud Auth0 autentimisliides. See teenus pakub kasutajate haldust ja sisselogimislehte, mille ise tegemine oleks võtnud palju arendusressurssi, mis sai suunatud olulisemate osade arendamisse.

Selles teenuses saab hallata kasutajate nimekirjasid ja nendele määratud õigusi. Kasutajate autentimiseks pakub see teenus REST-teenuse ja kasutajaliidese jaoks teeke, mis lihtsustasid integreerimist oluliselt. Kasutajate valideerimiseks kasutatakse JWT võtmeid. Need võtmed saab kasutajaliides Auth0 teenuselt pärast sisse logimist ja igal päringul lisatakse need päringu päisesse.

5 Tulemused

Realisatsiooni tulemusel tekkis süsteem, mis võimaldab kasutajatel teha analüütikat sisuliselt igal teenusel või seadmel, mis omab internetiühendust. Tänu avatud süsteemi põhimõtetele arhitektuuris on võimalik liidestada paljusid süsteeme väljaspool rakendusi ja veebilehti. Sisuliselt piisaks oma lahenduse integratsiooni jaoks ainult internetiühendusest.

Joonisel 20 on kirjeldatud lõpliku kasutaja vooskeem rakenduse loomiseks ja sündmuste analüüsi tegemiseks. Sellel joonisel on näha, mis osadest liigub kasutaja süsteemis läbi, kui Ta seda teenust kasutab.



Joonis 20. Lõppkasutaja vooskeem

Kuigi realisatsiooni ei saanud testida reaalse rakendusega, siis olid selle jaoks tehtud ühiktestid põhiliste andmebaasioperatsioonide jaoks. Testidega on kaetud enamik rakenduste loomisega seotud loogikat, sündmuste lisamise ja nende analüütikaga seotud loogikat ning ka eesmärgiteekonna haldamise loogikat.

Tabelil 3 on näha ka testide katvus antud teenusklassidel, mis suhtlevad andmebaasiga. Nagu näha, on testide abil kaetud suurem osa sündmuste analüütikaga seotud klassist ja ka rakenduse haldamisega seotud klassist. Lisaks on eesmärgiteekonna haldamisega ja vanade sündmuste puhastamisega seotud klassid 100% kaetud.

Klass	**Klass, %**	**Meetod, %**	**Rida, %**
ApplicationService	100% (1/ 1)	80% (8/ 10)	76.5% (13/ 17)
CleanupService	100% (1/ 1)	100% (4/ 4)	100% (10/ 10)
EventService	100% (1/ 1)	70% (14/ 20)	85.5% (53/ 62)
FunnelService	100% (3/ 3)	100% (5/ 5)	100% (7/ 7)

Tabel 3. Testide kattuvus teenusklassidel

5.1 GDPR nõuetele vastavus

Kuna antud lõputöö on kirjutatud Euroopa Liidus, siis kehtivad ka selle poolt loodavale süsteemile GDPR nõuded. GDPR ehk isikuandmete kaitse üldmäärus sätestab seda, kuidas võivad süsteemid käsitleda isikuandmeid.

Analüüsiandmete kogumisel on arvestatud sellega, et kogumisel ei talletataks liigselt isikut tuvastavat informatsiooni. Näiteks on kõik inimesed identifitseeritud unikaalsete identifikaatoritega, mida ei ole võimalik seostada ühegi isikuga. Kõik sündmused, mis on automaatselt kogutavad, on tehtud sellise põhimõttega, et nendega ei saaks seostada ühtegi isikut.

Ainuke kogutav informatsioon, mis saaks isikut identifitseerida, on tema asukoht [16]. Selle kaitsmiseks on salvestamisel piiratud isiku asukoha täpsus ainult linna piiresse ja enne asukoha määramist küsitakse isikult rakenduses nõusolekut. Lisaks on tehtud ka automaatprotsess, mis eemaldab iga päev andmebaasist üle 6 kuu vanuselistelt sündmustelt asukoha informatsiooni.

6 Kokkuvõte

Selle lõputöö eesmärgiks oli luua universaalne analüüsilahendus, mis oleks laiendatav ka teiste süsteemidega peale Android-rakenduste. See eesmärk sai enamasti täidetud, kuna tänu õigetele arhitektuurivalikutele tekkis süsteem, mis on kergelt skaleeritav ja laiendatav.

Eelnevalt paika pandud tööprotsessi järgimine andis loodetud tulemuse, sest selle abil sai luua süsteemi, mis pakub liidestust olemasolevate Android rakendustega. Süsteemi kõik komponendid on eraldi skaleeritavad, sealhulgas ka andmebaas, mille koormust ja andmeid saab kergelt jaotada erinevate sõlmede vahel.

Lisaks sai süsteemi tugevaks eeliseks see, et see toetab sisuliselt kõiki analüütikat vajavaid teenuseid, millel on internetiühendus. Üheks näiteks oleksid asjade interneti seadmed, mis hakkavad üha rohkem kasutust leidma, mistõttu nendele analüüsi toe pakkumine muutuks äärmiselt oluliseks.

Kuigi analüüsi osa sai oluliselt lihtsustatud praeguse töö raames limiteerides statistikat ainult 14 päeva peale, on kogu informatsiooni kogumine siiski olemas ja tulevikus saaks olemasolevaid analüüsimeetodeid ka täiendada.

Lisaks saaks tulevikus teha ka muid täiendusi – rohkem analüüsivõimalusi, nagu näiteks konversioonid, detailsem sündmuste põhine statistika. Lisaks saaks pakkuda ka rohkem tuge teistele platvormidele nagu näiteks Javascript, iOS või Spring Boot, arendades neile oma SDK lahendused.

Lisaks enne süsteemi laiemat avalikustamist on äärmiselt oluline ka analüüsi vaheandmete talletamine kõrvalises andmebaasilahenduses, nagu näiteks Redis. Hetkel tehakse iga kasutajaliidese päringu peale andmebaasipäring, mis juba tuhandete kasutajate puhul võib osutuda problemaatiliseks, kuna käivitatavad päringud on keerulised ja koormaksid selle tõttu tohutult andmebaasi.

Kasutatud kirjandus

[1] MixPanel, „Pricing Guide,“ [Võrgumaterjal]. Available: https://mixpanel.com/pricing/. [Kasutatud 08 05 2019].

[2] Google, „Dashboard,“ [Võrgumaterjal]. Available: https://support.google.com/firebase/answer/6317517. [Kasutatud 12 05 2019].

[3] „Funnels,“ [Võrgumaterjal]. Available: https://support.google.com/firebase/answer/6317523. [Kasutatud 12 05 2019].

[4] Google, „Reports and configuration,“ [Võrgumaterjal]. Available: https://support.google.com/firebase/topic/6317489. [Kasutatud 19 05 2019].

[5] Google, „Add Firebase to your Android project,“ [Võrgumaterjal]. Available: https://firebase.google.com/docs/android/setup. [Kasutatud 15 05 2019].

[6] Google, „Log Events,“ [Võrgumaterjal]. Available: https://firebase.google.com/docs/analytics/android/events. [Kasutatud 18 05 2019].

[7] MixPanel, „Android,“ [Võrgumaterjal]. Available: https://developer.mixpanel.com/docs/android. [Kasutatud 15 05 2019].

[8] N. J. R. E. S. Ron Kohavi, „Emerging Trends in Business Analytics,“ *Communications of the ACM,* kd. 45, nr 8, pp. 46-47, 2002.

[9] A. Kaushik, „Traditional web analytics is dead,“ *Web Analytics: An Hour a Day*, Sybex, p. 9.

[10] Google, „Funnels,“ [Võrgumaterjal]. Available: https://support.google.com/firebase/answer/6317523. [Kasutatud 19 05 2019].

[11] Google, „Cohorts,“ [Võrgumaterjal]. Available: https://support.google.com/firebase/answer/6317510. [Kasutatud 19 05 2019].

[12] Google, „Audiences,“ [Võrgumaterjal]. Available: https://support.google.com/firebase/answer/6317509. [Kasutatud 19 05 2019].

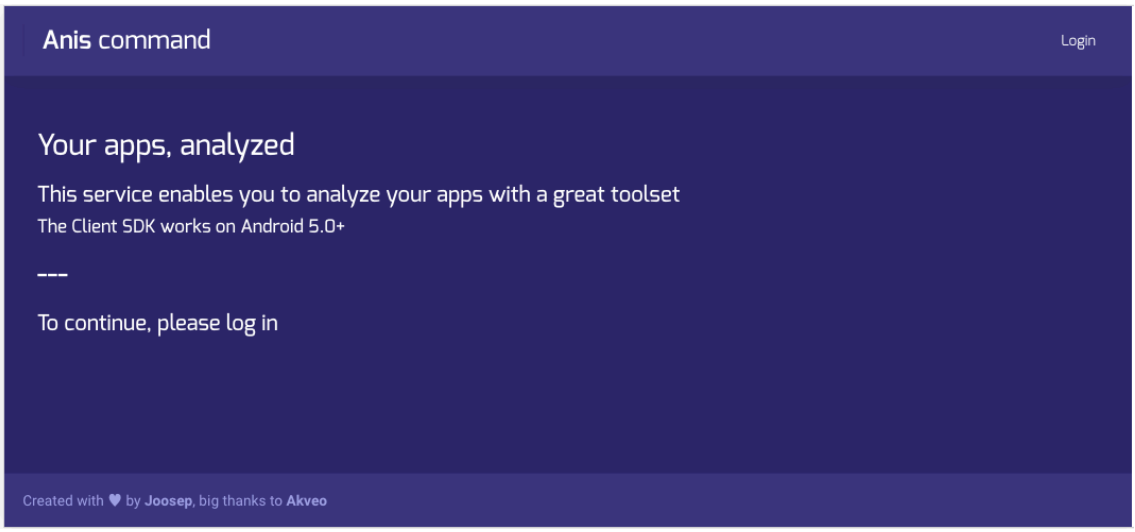
[13] Cockroach Labs, „SQL Feature Support in CockroachDB v2.1,“ [Võrgumaterjal]. Available: https://www.cockroachlabs.com/docs/v2.1/sql-feature-support.html. [Kasutatud 07 05 2019].

[14] Cockroach Labs, „CockroachDB in Comparison,“ [Võrgumaterjal]. Available: https://www.cockroachlabs.com/docs/v2.1/cockroachdb-in-comparison.html. [Kasutatud 07 05 2019].

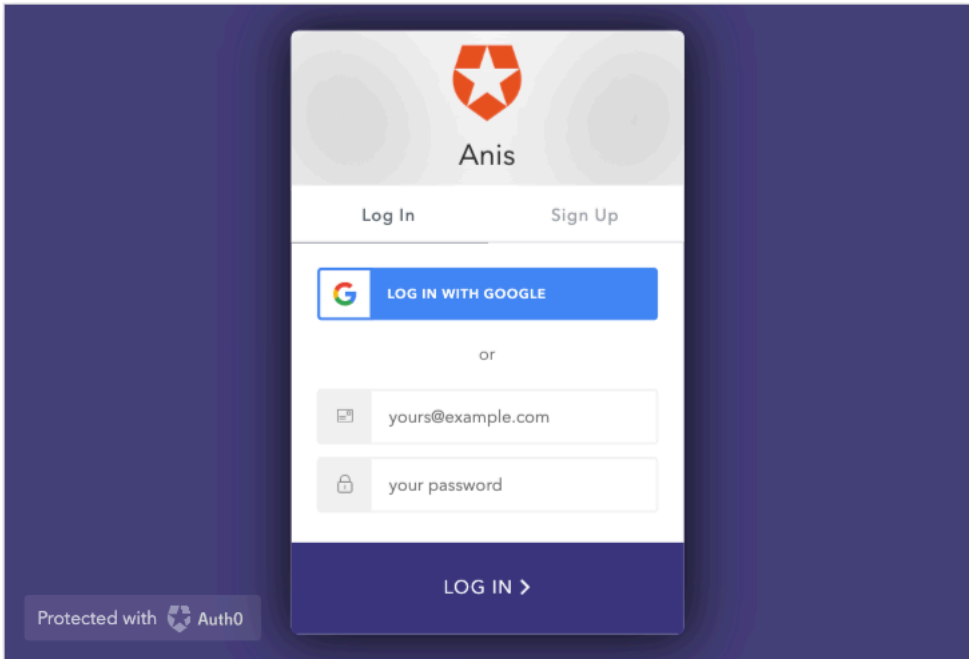
[15] The PostgreSQL Global Development Group, „26.3. Failover,“ [Võrgumaterjal]. Available: https://www.postgresql.org/docs/11/warm-standby-failover.html. [Kasutatud 08 05 2019].

[16] Intersoft Consulting, „Art. 4 GDPR Definitions,“ [Võrgumaterjal]. Available: https://gdpr-info.eu/art-4-gdpr/. [Kasutatud 14 05 2019].

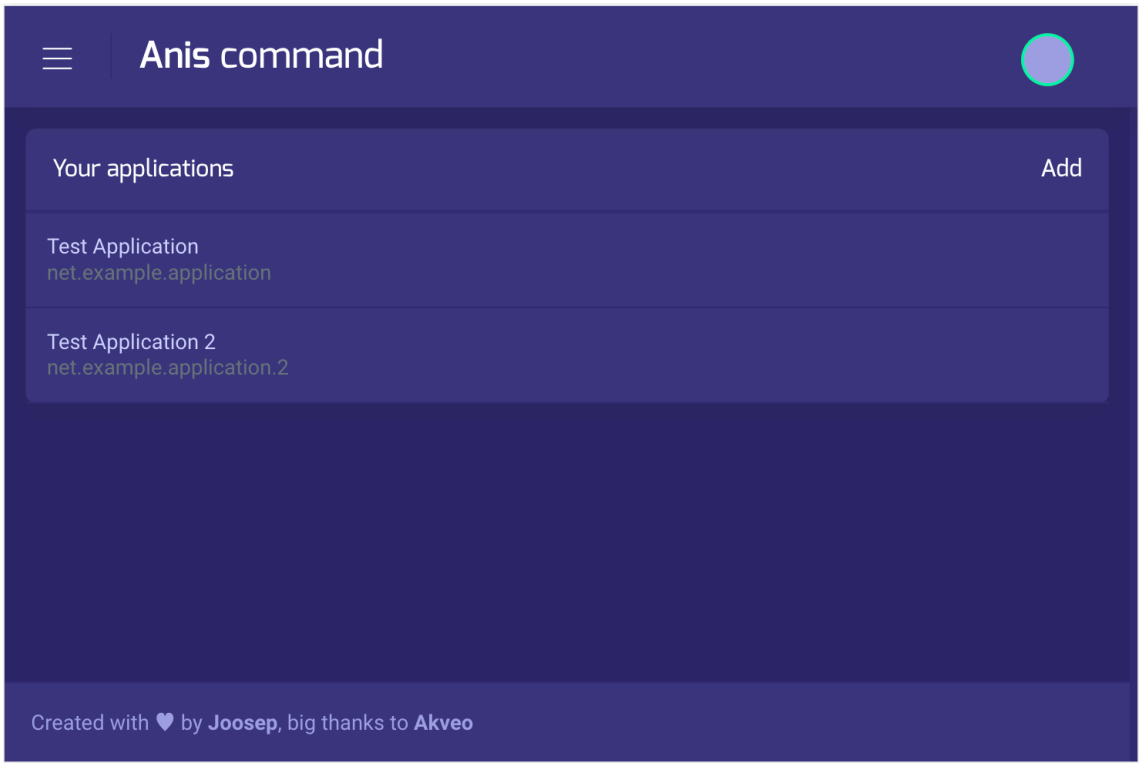
Lisa 1 – Kasutajaliidese vaated



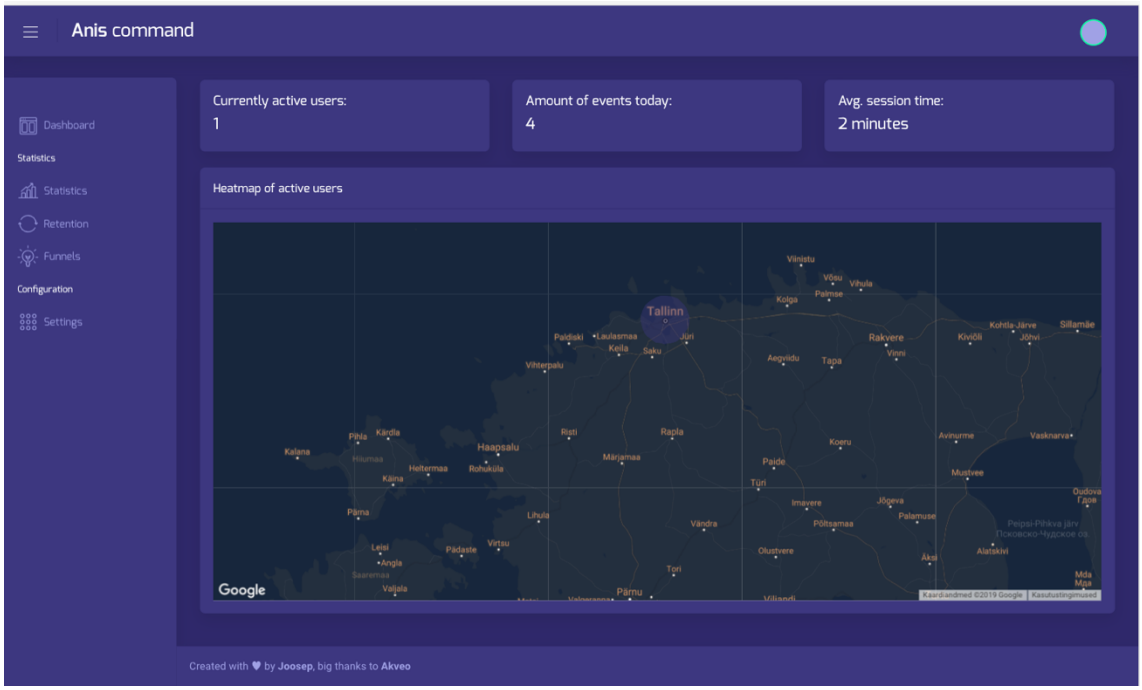
Joonis 21. Esilehe vaade



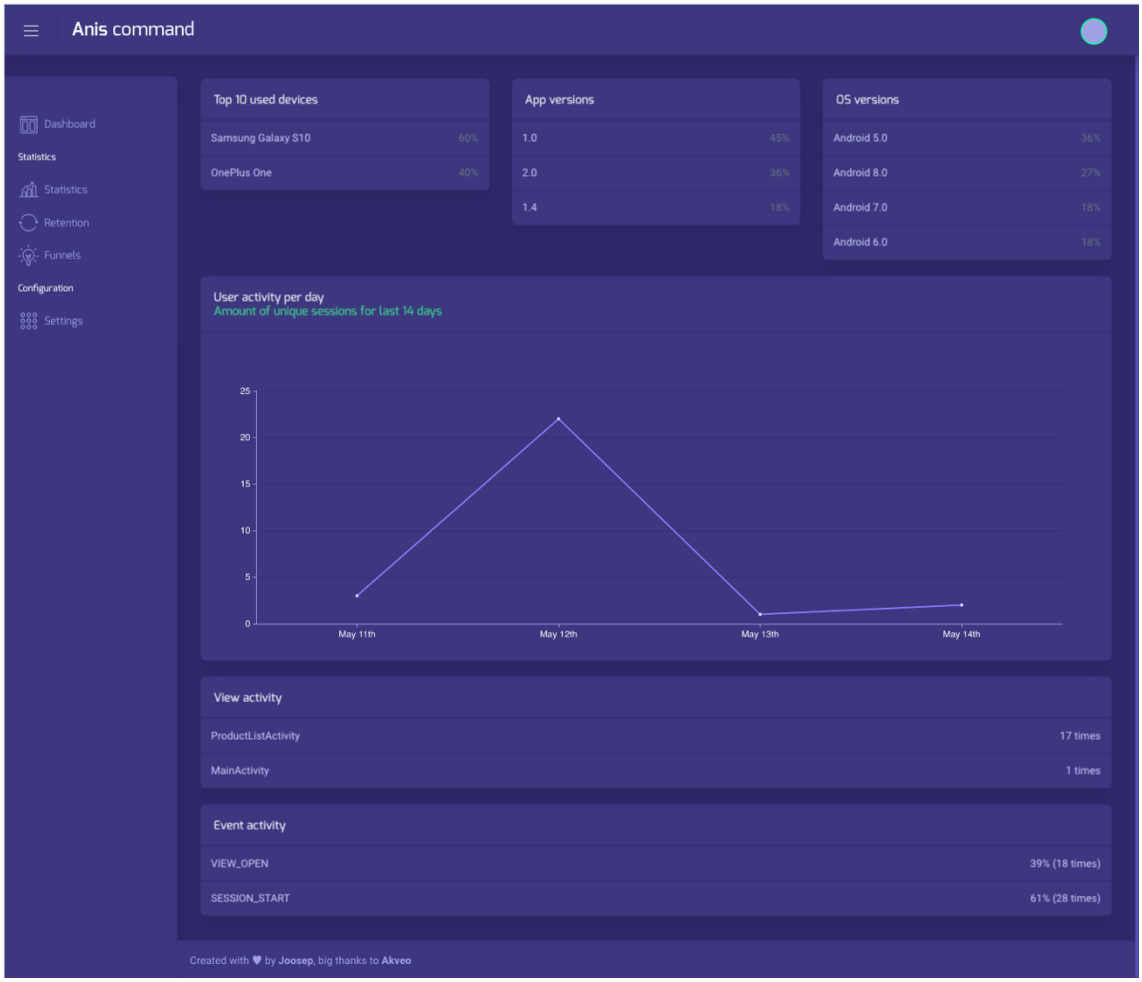
Joonis 22. Sisselogmislehe vaade



Joonis 23. Rakenduste haldamise vaade

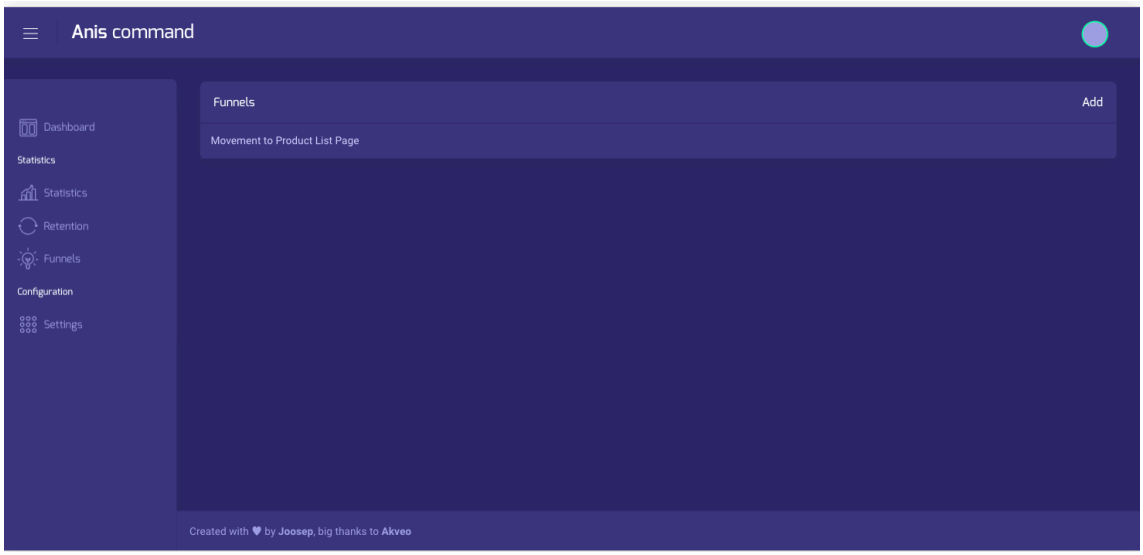


Joonis 24. Hetke statistika vaade

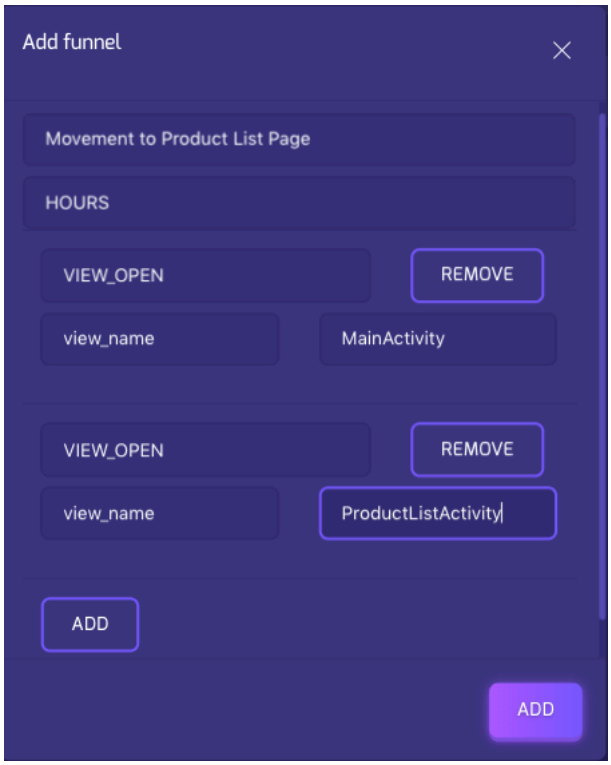


Joonis 25. Ajaloolise statistika vaade

Joonis 26. Kasutajate jäävuse vaade

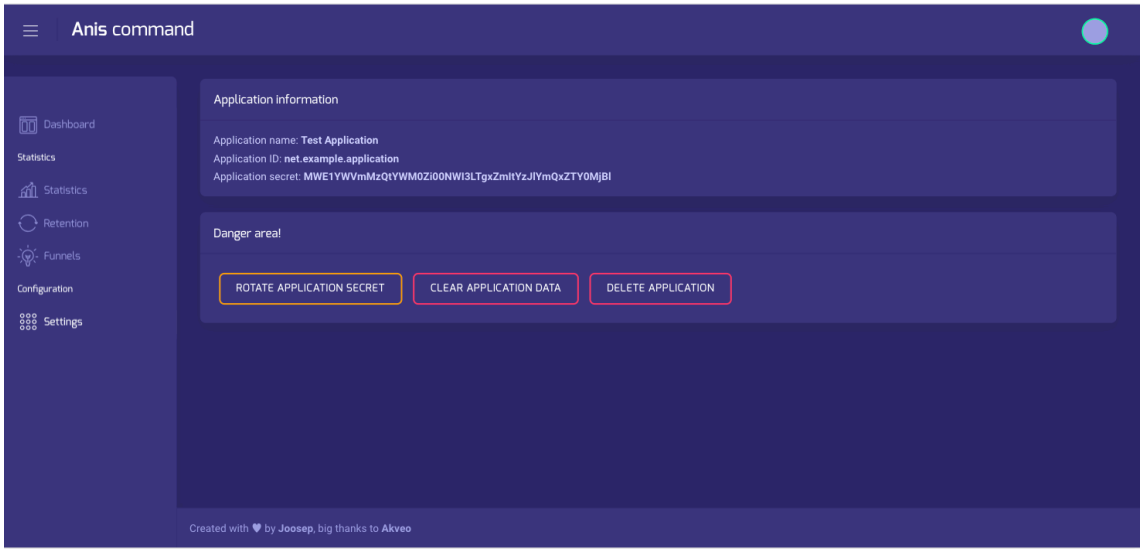


Joonis 27. Eesmärgiteekondade haldamise vaade



Joonis 28. Eesmärgiteekonna lisamise vaade

Joonis 29. Eesmärgiteekonna graafiline vaade



Joonis 30. Rakenduse seadete vaade