Expertise
and insight
for the future

Teemu Leino

Collecting Usage Data with Google Analytics for Firebase

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

14 May 2021

Author Title Number of Pages Date	Teemu Leino Collecting usage data with Google Analytics for Firebase 26 pages 14 May 2021
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Janne Salonen, Head of Department (ICT)

The aim of this thesis was to define best practices related to collection of usage data and explore the capabilities of Google Analytics solution on Firebase mobile platform. The objective of the related final year project was to provide means for a client company to gain a deeper insight into their mobile application's user base.

In this paper, prerequisites and processes needed for deploying Google Analytics onto a mobile solution already utilizing Firebase platform were investigated. To fulfill the needs of the client company, four real-world use cases utilizing these services were designed and implemented.

The results showed that when implementing a Google's analytics solution in the EU, it is vital to learn and follow data privacy laws and Google's terms. The use cases of the client company were implemented, but their use in production has not yet been started. Therefore, all the usage data collected during this project and analyzed in this paper was generated by the employees of the client company.

In conclusion, Firebase platform provides powerful tools for collecting and inspecting application usage data. Once the implemented solutions are in production use, the client company can utilize the collected data to better understand the end users. However, there are many things to keep in mind in order to respect the privacy and rights of these data subjects.

Keywords	Google Analytics, Firebase, analytics, hybrid mobile application, usage data collection, data privacy

Tekijä Otsikko Sivumäärä Aika	Teemu Leino Collecting usage data with Google Analytics for Firebase 26 sivua 14.05.2021
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tietotekniikan koulutusohjelma
Ammatillinen pääaine	ohjelmistotekniikka
Ohjaajat	osaamisaluepäällikkö Janne Salonen

Insinöörityön tarkoituksena oli selvittää Google Analytics -analytiikkaratkaisun käytettävyys Firebase-mobiilialustalla ja määritellä tähän liittyviä parhaita käytäntöjä käyttödatan kerääjän näkökulmasta. Insinöörityö tehtiin yhteistyössä yrityksen kanssa, ja työn tavoitteena oli tarjota yritykselle kyvykkyys ymmärtää tämän kehittämän mobiilituoteratkaisun käyttäjäkuntaa paremmin.

Tarkoitukset ja tavoitteet saavuttaakseen insinöörityössä selvitettiin, miten Google Analytics for Firebase otetaan käyttöön tuoteratkaisussa, joka jo ennestään hyödyntää Firebaseä. Tämän lisäksi työssä suunniteltiin ja kehitettiin neljä käyttötapausta käyttödatan keräämiselle, kun otetaan huomioon yhteistyöyrityksen tavoitteet.

Tulokset osoittivat, kuinka tärkeää tietosuojalakien ja Googlen ehtojen ymmärtäminen ja huomioiminen on, kun hyödynnetään Googlen analytiikkaratkaisuja EU:n sisällä. Insinöörityössä suunnitellut ratkaisut käyttötapauksille toteutettiin, mutta ratkaisujen käyttöönottoa odotetaan vielä. Tästä johtuen kaikki työssä kerätty ja hyödynnetty käyttödata on yhteistyöyrityksen ohjelmistokehittäjien työssään generoimaa dataa.

Johtopäätöksenä voidaan todeta, että Firebase-alusta tarjoaa tehokkaita työkaluja käyttödatan keräämiselle ja tutkimiselle. Kun työ on otettu tuotantokäyttöön, voidaan sen avulla kerättävää käyttödataa hyödyntää käyttäjien syvempää ymmärtämistä varten. Tästä huolimatta on monia seikkoja, joita datan kerääjän tulee ottaa huomioon, kun dataa kerätään eettisesti ja lakia noudattaen — varsinkin Googlen analytiikkaratkaisuja hyödynnettäessä.

Avainsanat	Google Analytics, Firebase, analytiikka, hybridimobiilisovellus, käyttödatan keräys, tietosuoja

Contents

Glossary

CGI	The client company with whom the final year project was done.
CoffeeScript	A programming language that compiles into JavaScript. Used extensively in the client company's application.
Cordova	A development framework that allows HTML, CSS and JavaScript to be used to build native mobile applications.
cordova-plugin-firebase	A Cordova open-source library for Firebase SDK and Google Analytics for Firebase.
EEA	European Economic Area. It consists of EU member countries, Iceland, Liechtenstein and Norway.
GDPR	General Data Protection Regulation. An EU law that considers collecting and processing of personal data.
SaaS	Software as a Service. A business model where the client typically pays for a customizable online service instead of a license to use the software.
SDK	Software Development Kit. A set of tools for software development on a specific platform.
serverless	A term used to describe computing services that scale on an as-used basis for more flexibility.

1 Introduction

Over the past few years, it has become hard to browse the Internet or use mobile applications without encountering banners and dialogs asking for our consent to use cookies and other tracking technologies. The reason is simple: our consent allows the service provider to collect and store information about us, usage data, as we interact with the service. The thesis explores what kind of usage data these services may collect and analyzes what are some of the ways this data can be utilized in product development. The thesis is part of a final year project, whose goal was to implement and deploy usage data collection on top of an existing hybrid mobile application.

The project was done in collaboration with a client company called CGI Suomi Oy, which is a part of a multinational IT services and consulting company, CGI Inc., with around 3,700 employees in Finland [1]. An in-production hybrid mobile application built using Apache's Cordova mobile application development framework served as the base for the project. The application is a part of a workforce management SaaS solution and, at the time of writing, has between 15,000 to 20,000 monthly Android users in production. Majority of the application's users work on either facility care or home care and use the application in their day-to-day work. CGI hoped that Google's Google Analytics for Firebase solution would help them gain a deeper insight into their user base and learn more about how the users interact with their mobile application. Google's solution was chosen by CGI because the application was already using Firebase platform for its push technology — among other features.

Four use cases with the aim of learning more about the user base were identified together with the client company. Potential solutions for these use cases were designed and implemented independently of the client company as a part of the final year project. The thesis evaluates the capabilities of Firebase platform's analytics features for fulfilling the use cases.

2 Usage Data, Google Analytics and Firebase

2.1 Usage Data

Usage data is information that is generated by a service's, for example, a website's or a mobile application's, end users as they interact with the service. The service's provider then stores the data for later use. Some applications for this data are further explored in Section 3.4. This collected data can include personal information, such as an IP address of a user, or information that is more general in nature, for example:

- How long a user spent using the service during a session.
- Through what medium, as in a web browser or a mobile device, a user accessed the service.
- Which interactive elements, as in buttons or links, a user clicked or tapped in the service.

In case a service is collecting personal data, its users should be able to find more information about what personal data is being collected and for what reasons from the privacy notice or data usage policy of that service. The EU's General Data Protection Regulation law — or GDPR for short — requires this from organizations that collect personal data from citizens and residents in the European Economic Area, such as in Finland [2]. For services that use Google Analytics for Firebase, Google's policies have similar requirements for its users and require them to disclose how end users' data is being collected and processed [3].

2.2 Google Analytics and Firebase

2.2.1 Background

Google Analytics is Google's free web analytics tool that helps application or site owners get a deeper understanding of their end users and customers. This is done by, for example, viewing dashboards and generated reports that tell how people are engaging with the application or site. The reports are automatically generated from usage data that can be collected from the end users as they interact with the application or site. This deeper

understanding provided by Google Analytics can be useful when figuring out means for improving one's service or sales. [4]

Firebase, on the other hand, is Google's mobile platform that is targeted for developers, marketers and product managers. Firebase offers various features and integrations that can be used to help improve application performance and gain insight into the application's end users. At its core, Firebase has Google Analytics that integrates across Firebase's features. These features, such as Google Analytics for Firebase, can be used individually, but there is a strong incentive to use them together as they are all on the same platform and have the possibility to share data together. [5]

While this thesis mainly concentrates on Google Analytics and Firebase, Google is not the only provider of such analytics and mobile services on the market. However, similar solutions generally share the same goal: providing means for gaining an insight into end users' behaviors.

2.2.2 History

Google launched Google Analytics in November 2005. Google claims it was the first web analytics product of its class that was available for free. [6]

For years, Google has not revealed how many sites are using Google Analytics, but according to BuiltWith web technology tracking service, Google Analytics is in use by almost 29 million websites [7]. Google Analytics is also available in close to 40 languages — including Finnish [8].

A company called Firebase Inc. launched Firebase on April 12, 2012 in beta [9]. Just two and a half years later, on October 21, 2014, Google acquired Firebase [10] as illustrated by Figure 1.

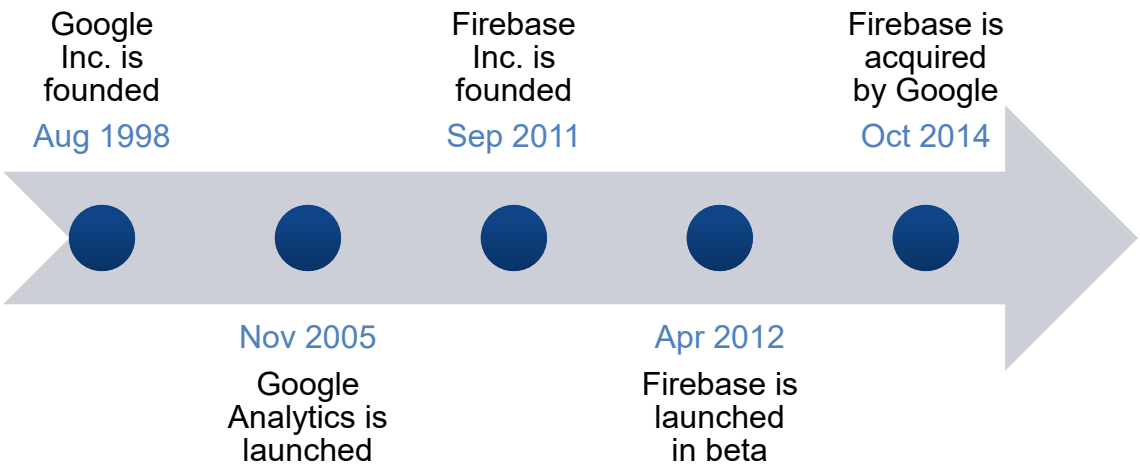

Figure 1. Timeline of the history of Google Analytics and Firebase [11; 12].

According to Google as of September 2019, there are over 2 million applications actively using Firebase every month [13]. Among these are the applications of The New York Times, Alibaba.com and Trivago [14].

2.3 Firebase Platform

Firebase comes with a software development kit, Firebase SDK, that is available with no cost on Android, iOS and the web. While Firebase can be used without charge using the entry-level Spark Plan, more features and integrations become available when paying for a subscription plan: Blaze Plan. Both plans include a certain amount of free usage, such as 5 GB of cloud storage, with the more feature-rich plan functioning as a scalable pay-as-you-go plan. Among these features of Blaze Plan are

- machine learning
- authentication
- file storage
- databases
- crash reporting

- application performance monitoring
- testing services
- push notifications
- and, of course, analytics. [14; 15]

Firebase SDK's documentation can be found at firebase.google.com. It contains high-level descriptive instructions along with full technical API documentation that help getting started with a new Firebase project. The documentation also contains code snippets and sample projects for iOS, Android, JavaScript, C++, Unity, Java and Node.js [16]. When working on the final year project, this documentation helped provide answers to many questions that were raised during the development.

2.3.1 Firebase Console

Firebase console is an interactive single-page application located at console.firebase.google.com and is used to access the features of Firebase. To gain access to the Firebase console, the user must first agree to Firebase's terms and create a Firebase project while signed in with a Google Account. Creating a Google Account is further explored in Section 3.1. Creating a Firebase project with Google Analytics is free when using the entry-level Spark Plan and can be done in three steps:

- First, the project is given a name.
- Second, the decision to enable Google Analytics is made.
- Third, a Google Analytics account is given a name, the country of the organization is selected, data-sharing settings are toggled and Google Analytics terms are accepted.

Once a project has been created, the user is prompted to add an application to the project. Firebase console should look akin to Figure 2.

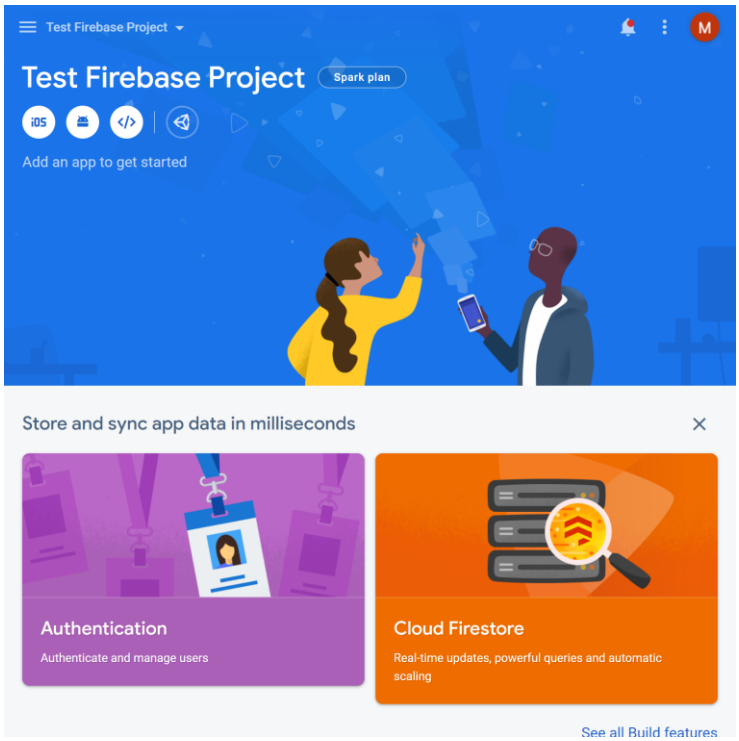

Figure 2. Firebase console after creating a new Firebase project.

The Firebase project can now act as a container for added applications. Within these containers, the applications share data between the various Firebase features [17]. These features are explored more in the next sections.

2.3.2 Google Analytics

With Google Analytics for Firebase, the collection of usage data is done through events. Events have types and contain parameters that carry the actual usage data. Some event types and parameters are inherently a part of the Google's solution, but Google Analytics for Firebase allows developers to create custom event types and parameters. From an end user's point of view, there is never a need to submit events or parameters manually. The data collection is done automatically in the background at certain intervals.

Inherent event types are in place without implicitly having to implement them and are almost exclusively related to the application's updates, removal and unhandled exceptions or crashes. Inherent parameters include information such as when the event took place and what was the application's version. These pieces of information are always a

part of all events. By default, Google Analytics for Firebase automatically collects and stores information about

- number of users and sessions
- session duration
- operating systems
- device models
- geography
- first launches
- application opens
- application updates
- in-application purchases. [18]

Custom event types and parameters provide means for more controlled and specific usage data collection. During the development of this final year project, it was found out that designing and implementing different kinds of event types and parameters requires a lot of thought, and this ended up being the bulk of the work. This topic is explored further in Section 3.4.

Google Analytics mainly utilizes cookies and similar tracking technologies to track users' interactions on websites and applications that use Google Analytics [19; 20; 21]. For Google Analytics for Firebase and mobile applications, the users' devices are identified using advertising IDs on both Android and iOS or alternatively Android Secure ID on Android [18].

Google Analytics allows the customer to choose how long the data is retained before it is automatically deleted. These retention options go from 14 to 50 months while there is also an option for the data not to automatically expire [22]. For Google Analytics for Firebase, Google retains the collected Analytics data for a maximum of 14 months [23].

2.3.3 Integrations

Firebase claims to support multiple integration solutions that can be used to make the most of Firebase. These integrations are not all a part of the free Spark Plan; some

require the user to upgrade to the pay-as-you-go Blaze Plan. While most of the integrations are a part of Google's product family, there are also a few third-party options. Firebase claims to support the following integrations:

- Google Ads
- AdMob
- Google Marketing Platform
- Google Play Store
- Data Studio
- BigQuery
- Slack
- Jira
- PagerDuty. [15; 24]

Perhaps unsurprisingly, a notable part of the integrations — i.e., Google Ads, AdMob and Google Marketing Platform — is related to digital advertising. After all, advertising is the main source of revenue for Google [25]. Events generated by Firebase can be automatically distributed to these advertising solutions in order to:

- See how ad campaigns affect application installs and in-application purchases.
- Display ads tailored to the application.

The events can also be distributed to Google Play Store and tools, such as Data Studio and BigQuery, to give insight on the data. An integration to the instant messaging platform Slack allows Firebase to generate notifications of application crashes in Slack channels, while Jira integration provides means for, for example, allowing the investigation of these crashes to be managed. Firebase also supports integrating with PagerDuty to page on-call responders when certain events are logged.

Out of the nine previously listed Firebase integrations, BigQuery was the only one deemed essential for the final year project. From BigQuery user's point of view, BigQuery is fundamentally a serverless database that allows querying the data collected with Google Analytics for Firebase using SQL database management language. BigQuery can be accessed from Google Cloud Platform at console.cloud.google.com/bigquery and behaves as a single-page application similarly to the Firebase console.

With BigQuery, this data can be viewed and exported in ways that would normally not be possible. For example, Firebase console provides means to see how many times a certain event type, such as an event about changing the application's language, was logged during a given date, but not much more than that. BigQuery, on the other hand, allows viewing detailed information about each individual instance of the event type and can answer questions such as:

- What was the exact time the event was logged?
- What kind of a device logged the event?
- What was the city where the event was logged?
- What were the custom parameters attached to the event?

BigQuery can be queried using SQL clauses such as SELECT, FROM, WHERE and LIMIT. Figure 3 demonstrates a single result for a query of "SELECT event_timestamp, device.mobile_brand_name, device.mobile_marketing_name, geo.city, event_params FROM `test-firebase-project.analytics_180174203.events_20210430` WHERE event_name = 'change_language' LIMIT 1000". Through the BigQuery's user interface, the query results can also be saved to another BigQuery table or by exporting them into JSON, CSV or Google Sheets. BigQuery usage in practice is further explored in Section 3.4.4.

Row	event_timestamp	mobile_brand_name	mobile_marketing_name	city	event_params.key	event_params.value.string_value	event_params.value.int_value
1	1619773064912003	Samsung	Galaxy A70	Helsinki	target_language	en_GB	*null*
					source_language	fi_FI	*null*
					engaged_session_event	*null*	1
					connection_status	ONLINE	*null*
					ga_session_number	*null*	1
					ga_session_id	*null*	1619773030
					firebase_event_origin	app	*null*
					firebase_screen_class	MainActivity	*null*
					firebase_screen_id	*null*	-5324355935423188

Figure 3. Querying for details of an event type about changing the application's language using BigQuery.

For the final year project, BigQuery was essential in order to be able to get the most of the collected usage data. For example, the client company was interested in being able to see which error and warning messages end users encounter the most to know where

the focus should be when trying to improve user experience. Two ways were identified to provide a solution to this use case:

- Create an event type for each different message that the user might encounter.
- Create a single event type and use a custom parameter, the unique key of the message, to differentiate the messages.

The first way was deemed unfeasible. There were potentially over 700 different messages, and Google Analytics for Firebase allows reporting only up to 500 [26] different types of events per application. The second way would not fulfill the use case when only using Firebase console to view event information because Firebase console does not provide means to view a list of all instances of an event type separated by a custom parameter. More use cases are explored in Section 3.4.

3 Collecting Usage Data

3.1 Prerequisites

Before usage data collection can be started using Google Analytics for Firebase, certain prerequisites must be fulfilled first. First, a Google Account is needed to gain access to Firebase console. Registered users, such as all users of Google's Gmail, can login with their existing Google Account's credentials to access Firebase console. Creating a Google Account is free, and during the registration process, the user is asked to submit the following details:

- first and last name
- an existing email address or a new Gmail email address
- a password
- optionally a mobile phone number or a recovery email address
- birthday
- sex.

Second, Google's Terms of Service regarding Google Analytics for Firebase require its customers to include an appropriate privacy policy inside applications that use the service. The privacy policy ought to be easily accessible by the end users and must contain information how the end users are being identified — for example, by cookies or pseudonymous user identifiers — along with how the data is being collected and processed. The way suggested by Google to comply with these terms is to provide a noticeable link to Google's site "How Google uses data when you use our partners' sites or apps", which is located at www.google.com/policies/privacy/partners/. [3]

Third, Google's Use Policy for Google Analytics for Firebase also has a few requirements of its own. Among other information, it requires its customers to inform the end users about which of the Google Analytics for Firebase features have been implemented and how the end users can opt-out of these features. It also demands complying with Google's European Union User Consent Policy that concerns end users residing in the EEA. In short, this policy requires obtaining the end users' consent to the use of cookies or other similar means of local storage along with the use of collection, sharing and use of personal data for personalization of ads. [27; 28]

3.2 Deployment Actions

The main goal of the final year project was to implement and deploy Google Analytics for Firebase solution on top of an in-production mobile application. Technically the deployment required making minor modifications to the application's source code. From a legal point of view, this required reading and comprehending data privacy laws and Google's terms, which turned out to be difficult and time-consuming for a person without a law degree.

The application was already utilizing a Firebase-related Cordova plugin, an open-source library managed by the open-source community, called cordova-plugin-firebase. The application was using the plugin for Firebase Cloud Messaging and its push technology. This meant that the already existing Firebase project and the Firebase SDK required little configuration for the application to start utilizing Google Analytics for Firebase. For new users getting started with Firebase, Firebase console provides systematic instructions how to add Firebase to an Android application.

Unfortunately, the Cordova plugin had not been updated since October 2018 and required modifications [29]. The plugin's source code had to be altered so that the Google Analytics for Firebase data collection would be disabled by default and would not collect advertising IDs or Android Secure IDs. This was achieved by changing the plugin's configuration file, plugin.xml, as illustrated by Listing 1. These IDs could be used to track and identify end users' for, for example, advertising purposes [30]. This was deemed unnecessary for the application, as it does not contain advertisements.

```
<config-file target="AndroidManifest.xml" parent="/manifest/application">
     <meta-data android:name="firebase_analytics_collection_enabled"
          android:value="false" />
     <meta-data android:name="google_analytics_adid_collection_enabled"
          android:value="false" />
     <meta-data android:name="google_analytics_ssaid_collection_enabled"
          android:value="false" />
</config-file>
```

Listing 1. Disabling the Analytics-related features in the cordova-plugin-firebase's plugin.xml file.

To comply with Google's Terms of Service and Use Policy, one of the required actions was to initialize Google Analytics for Firebase only after the end user had given his or her consent for usage data collection. During the development of this final year project, the obtaining of user consent for usage data collection was decided to be handled after the user had logged in into the system. This way no usage data would be collected out of any interactions of non-authorized end users who likely would not be familiar with the application's privacy policy. As mentioned in Section 3.1, Google's Terms of Service require providing a readily accessible link to the application's privacy policy for the end users. Once this project is in production use, end users will be able to find a link to the application's privacy policy from the login view.

3.3 Data Collection Best Practices

When collecting usage data with Google Analytics for Firebase, extra care ought to be taken to respect the privacy of the end users, follow Google's policies and make sure the data stays relevant. Failure to follow the data privacy regulations and laws, such as GDPR, can lead to fines in the millions of euros for organizations [31]. Not following the policies and terms defined by Google can prompt Google to revoke one's right to use the

service [3]. By keeping the collected usage data relevant, trust in the data's integrity remains uncompromised.

3.3.1 Avoiding Collecting Personal Data

To be GDPR-compliant, one should be mindful about collecting personal data in case it is at any time stored in servers that reside outside the EEA. Unfortunately, the data collected using Google Analytics for Firebase can be stored in the United States [32]. A legal contract regarding data protection when transferring personal data outside the EEA, i.e., a Standard Contractual Clause, can be made to tackle this problem [20; 33], but that is out of the scope of this thesis.

Like GDPR, Google prohibits the users of Google Analytics for Firebase from collecting information that Google deems to be usable for identifying the data's subject. Google calls this information personally identifiable information — or PII for short. Since there is an overlap between GDPR's personal data and Google's PII, here is a generalized non-exhaustive list that considers both for an idea what kind of information one should not log with events and parameters:

- direct and indirect personal identifiers, such as individuals' names, home addresses, email addresses, phone numbers, social security numbers, heights, hair colors, geographic coordinates, usernames, database user IDs and license plate numbers
- online identifiers, such as IP addresses, MAC addresses, cookie identifiers, IMEI numbers and RFID tags
- direct user input, such as form data or URLs with query parameters that consist of user input
- information related to an individual, such as medical history and criminal records. [34; 35; 36]

One should also be mindful about accidentally collecting outlier subject information. For example, creating a custom parameter for user role information can violate user privacy; the system might contain only a single individual for any specific role — for example, an administrator —, which can then be identified due to being an outlier in the collection of data.

3.3.2 Separating Production and Test Data

Test data generated during development is best kept separate from production data. This makes it easier to debug problems in the data collection while also ensuring integrity of the production data. Data integrity is important in order to be able to keep making correct, informed decisions with the help of the data. When investigating means to achieve this integrity, it was found out that Firebase supports multiple projects and having separate projects for development and production environments [37]. This seemingly was an effective and simple way to have separate containers for different data sources.

A second Firebase project was created as described in Section 2.3.1. After that, the application's Firebase SDK needed to be reconfigured to allow the collected data to flow into the correct container: test data into development project and production data into production project. Firebase SDK on Android required this decision to be made during build time; it could not be changed on the fly during runtime [37]. Therefore, there was a need to alter the application's build process.

Fortunately, the application's build process had already been divided into two types: debug for development and release for production. A further distinction between these two build types were made by appending the Android application ID with ".debug" in case of a debug build. This effectively made these two build types into separate applications allowing them to be installed at the same time on an Android device. This decision was made after realizing it made the development process easier for the client company's developers; developers no longer had to uninstall the application when switching between the two build types. It also allowed the Firebase SDK to be configured so that the package name would define which Firebase project would be used.

The build.gradle file in src/android/ folder of the Cordova plugin was updated with configuration that made the Android application ID reflect the build type. Listing 2 illustrates how this was achieved.

```
android {
      buildTypes {
            debug {
                  applicationIdSuffix ".debug"
            }
      }
}
```

Listing 2. Appending debug build type's application ID with ".debug" in cordova-plugin-firebase's src/android/build.gradle file.

3.4 Client Company's Use Cases

The final year project was done in collaboration with a client company, CGI. CGI had hopes that an analytics solution would help them better understand how their software was being used and thus make better product development decisions. In total, four main use cases were fulfilled with over 30 different custom event types and over a hundred custom parameters — all implemented by one software developer.

When designing custom event types for the use cases, consistency in the structure of the events was a priority. The design goal was to make it easy to determine what information would each parameter represent and in which format just by looking at its name and knowing the design principles. The goal would be considered fulfilled when one would not have to look at the documentation when, for example, writing queries in BigQuery. In practice, this meant that all:

- ID-related custom parameters would be stored as strings representing alphanumeric values and underscores.
- Boolean values would be stored as lowercase strings: "true" or "false".
- Numeric values would be stored as doubles representing double-precision floating-point numbers.
- Enumerated types would be stored as uppercase strings.
- Custom event types and parameters would have their names in lowercase and follow the naming conventions for events [26] defined by Google.

3.4.1 Errors and Warnings Encountered

The purpose of the first use case was to provide insight into what are the areas where the end users struggle the most. Knowing which error and warning messages were the

most prevalent was believed to be valuable for guiding the focus in product development — especially when trying to improve the user experience. Cutting down the number of times end users encounter errors was thought to make users take them more seriously. This use case was already partly covered in Section 2.3.3.

Table 1 illustrates the structure of the show_prompt custom event type that would be logged whenever any message would be displayed to the end user. Parameter prompt_type has two possible values that differentiate between an error and an informative message, which is also used for warnings in the application. Parameter language_keys contains information about which predefined message or a combination of messages were displayed.

Table 1. Event structure of the show_prompt custom event type.

Event parameters	**Expected values**	**Data type**
prompt_type	ERROR, INFO	string
language_keys	*varies*	string

Implementing this use case was comparatively simple. There was one utility function, showPrompt, in the code that would always end up being called whenever a message was to be displayed to the user. Adding just one line of code at the end of showPrompt function was enough to log a show_prompt event every time when needed.

3.4.2 Feature Interactions in Numbers

The purpose of the second use case was to provide a better understanding about which features are the most used and whether there are features that are ignored by the end users. This information could then be used, for example, to:

- Determine whether a new feature was being adopted at a desirable rate or was there a need to somehow improve it.
- Decide whether a feature could be removed due to being underutilized.
- Guide the product development in prioritization of improving the features.

Table 2 illustrates the structure of the show_view custom event type that would be logged whenever an end user navigated to a view in the application. Parameter view_name provides a human-readable identifier for the entered view, whereas view_id tells the unique identifier.

Table 2. Event structure of the show_view custom event type.

Event parameters	**Expected values**	**Data type**
view_id	*varies*	string
view_name	*varies*	string

Even though the application had close to 50 different views, the call to log a show_view event could be handled within just one place. This was because all the views in the application inherited the same base class and called the same onShow method when changing a view. The views also all had an ID and a name stored in variables, so there was no need to pass the same information as arguments manually when calling the log event function.

Table 3 illustrates how the tap_item custom event type is structured. This event would be logged whenever an end user tapped a certain button or similarly interactive user interface element. Parameter item_id is for the unique identifier of the tapped element, and parameter item_info is sometimes used to provide more information about the situation. For example, when tapping a button to delete received messages in the application, the number of selected messages would be logged in the item_info parameter.

Table 3. Event structure of the tap_item custom event type.

Event parameters	**Expected values**	**Data type**
item_id	*varies*	string
item_info	*varies*	*varies*

In total, the function responsible for logging a tap_item event ended up being called from 22 different places in the code base. This was because the trigger to call the function had to be manually attached to each individual interactive element. However, not every interactive element in the application was deemed to have enough value to be a part of the usage data collection; there was no use to have the buttons used to navigate to a view also log a tap_item event.

3.4.3 Feature Usage in Details

The third use case had the purpose of shedding more light on how the end users interact with the features. CGI wished that an analytics solution would make it possible to find out, for example, how much typing is done within the mobile application. This, in turn, could help with estimating the value of new features, such as a speech-to-text voice recognition.

There are tens of forms an end user can fill-in inside the client company's mobile application. Some of them send information to the backend, whereas others, such as filter forms, change only the information stored locally on the device. Similarly, there are features that, when used, do not leave any trace for anyone outside the mobile device to see. Therefore, there was no way to know how and how often end users were using these features without an analytics solution. To fulfill the requirements of this third use case, over 20 custom event types were created as a part of this final year project. Among the usage data logged with these events was:

- When sending a message: how long the message was, how many recipients it had, was it being forwarded or was it a reply to another message.
- When filtering for contacts, recipients of messages or the contents of received and sent messages: how long the filter text was and what the unique ID of the filter UI element was.
- When changing the language of the application: what the selected language was and what the previous one was.
- When submitting an error report: how long the description was and how long the attached log file was.
- When using Bluetooth to scan for nearby wireless devices: how many devices were found.

CoffeeScript code for one of the event logging methods, logChangeLanguage, is presented in Listing 3. The method is very typical when compared to the other event logging methods written as a part of the final year project. It requires two string arguments that are used as custom event parameters but only if the analytics solution has been initialized by the application and toggled on in the backend system's configurations. In the end, the method calls the cordova-plugin-firebase's logEvent method, which is responsible for sending the event information into Firebase platform.

```
# Required: source (string)
# Required: target (string)
logChangeLanguage: (sourceLang, targetLang) ->
  return if not @isEnabled()
  if not _.isString(sourceLang) or _.isEmpty(sourceLang) or
  not _.isString(targetLang) or _.isEmpty(targetLang)
    logger.warn "Analytics: Logging Change Language failed: Invalid arguments"
    return
  logger.debug "Analytics: Logging Change Language"
  eventParameters =
    source_language: sourceLang
    target_language: targetLang
  @logEvent("change_language", eventParameters)
```

Listing 3. A CoffeeScript method that logs a change_language event.

3.4.4 Battery Drain

There had been queries made by the end users about the mobile application's drain on the device battery. Some had claimed that the effect had increased after updating the application. CGI hoped that an analytics solution would provide means to keep track of battery usage between different versions of the application. This was believed to help proving the problem truly was there and when investigating causes for the problem. This fourth use case about battery usage was designed to be fulfilled with one new generic-sounding custom event: logout.

The logout custom event type would be logged whenever an end user logged out of the application. The name of the event type was decided to not be specific to battery usage in case there would be a need to log even more data when logging out in the future. In addition, changing the structure or name of an event type after it had been logged would make it harder to utilize the previously logged usage data. For the scope of this project, it was decided that just logging the information about changes in the battery levels would be enough.

There are five custom parameters in the logout custom event type — as illustrated by Table 4. The parameters were designed in such a way that they would provide answers to the questions:

- How much battery was drained during a session?
- How long was the session?
- How long was the application being used actively during the session?
- Was the battery charged during the session?

Table 4. Event structure of the logout custom event type.

Event parameters	**Expected values**	**Data type**
battery_session_current_level	0 - 100.0	double
battery_session_start_level	0 - 100.0	double
battery_session_length_in_seconds	>= 0	double
battery_session_length_active_usage	>= 0	double
battery_session_charged	true, false	string

With the usage data provided by the logout custom event type, it is possible to find out how many percentages, on average across all the devices, battery is being drained in an hour while the end user is logged in. To prove this, two queries were written and run in BigQuery for test usage data collected during the April of 2021. The queries do not consider the application's version, but this is only because all the test data were collected using the same application version. As mentioned in Section 2.3.2, version information is logged automatically with Google Analytics and, therefore, could be used as a part of this use case.

First, a query was written for creating a new BigQuery table out of the results. This table would contain the data collected with the logout events in a neat format for easier querying. The results are illustrated by Figure 4.

```
SELECT
  ((SELECT value.double_value FROM UNNEST(event_params) WHERE key = "battery_session_start_level") -
  (SELECT value.double_value FROM UNNEST(event_params) WHERE key = "battery_session_end_level")) AS Drain,
  (SELECT value.double_value FROM UNNEST(event_params) WHERE key = "battery_session_length_in_seconds") AS Session_length,
  (SELECT value.string_value FROM UNNEST(event_params) WHERE key = "battery_session_charged") AS Charged
FROM
  `test-firebase-project.analytics_180174203.events_202104*`
WHERE
  event_name = "logout"
```

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 3.5 MB processed)

Job information | Results | JSON | Execution details

Row	Drain	Session_length	Charged
1	0.0	29.0	true
2	-1.0	412.0	true
3	0.0	337.0	false
4	0.0	1326.0	false
5	0.0	1111.0	false
6	0.0	79.0	false
7	0.0	116.0	false
8	0.0	43.0	false
9	0.0	196.0	false
10	0.0	460.0	false
11	0.0	139.0	false

Rows per page: 100 1 - 57 of 57 First page Last page

Figure 4. BigQuery query and its results for the logout event type and its parameters.

Based on the results, there were 57 instances of logout events logged in April. After saving these results into a new table called battery usage, a second query was written. This query would:

- Count the total number of data points (i.e., Sessions).
- Sum the total hours of the sessions (i.e., Total_usage).
- Sum the total percentage of the battery drain (i.e., Total_drain).
- Calculate the average battery drain based on the totals (i.e., Average_drain).
- Ignore sessions during which the battery was charged.
- Ignore sessions that were ten minutes or longer in length.

After writing the second query, it was run for the new table in BigQuery. The results are demonstrated in Figure 5.

```
1   SELECT
2     COUNT(*) AS Sessions,
3     SUM(Session_length / 3600) AS Total_usage,
4     SUM(Drain) AS Total_drain,
5     (SUM(Drain) / SUM(Session_length / 3600)) AS Average_drain
6   FROM
7     `test-firebase-project.analytics_180174203.battery_usage`
8   WHERE
9     Charged = "false"
10    AND Session_length < 600
```

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 1.3 KB processed)

Job information | Results | JSON | Execution details

Row	Sessions	Total_usage	Total_drain	Average_drain
1	43	2.1900000000000004	8.0	3.6529680365296797

Figure 5. BigQuery query and its results for a table containing battery usage data.

The results indicate that the average battery drain — based on test data generated in April — was close to 4 percentages per hour while logged in. While the results are not statistically very precise or even useful for the client company as-is, they prove that the logout events, Google Analytics for Firebase and BigQuery all together allow calculating the average battery drain while the application is running and the end user is logged in.

In conclusion, these methods ought to provide the client company means to keep track of the battery drain on a monthly basis. However, some possibly affecting factors that vary month-by-month, such as the ambient temperature, can have an impact on the results and should be kept in mind when interpreting the data.

4 Conclusion

It has been demonstrated that Google's Firebase platform provides two powerful tools, Google Analytics and BigQuery, for collecting and exploring usage data. From the outcome of the investigation, it is possible to conclude that deploying usage data collection requires a deeper understanding of the data privacy laws, such as GDPR. However, the results indicate that implementing an analytics solution on top of an in-production mobile application using Google's Firebase SDK, is possible and does not require a large team to do so.

The implemented analytics solution has great potential for other use cases than those that were fulfilled as a part of this project. For example, once the implementation is in production use, CGI could sell parts of collected usage data to the clients to provide the clients a deeper insight into their own workforce.

Clearly, further research will be needed to answer questions about the costs of using the Google's analytics solutions in production with tens of thousands of end users. It is reasonable to believe that future work will involve designing and implementing more custom event types and more complex queries that explore the collected data. The data could also be imported into visualization tools, such as Google Data Studio, which was not covered in this paper. All this ought to help CGI gain potentially an even better understanding of their application's end users.

References

1 *CGI yrityksenä*. CGI Suomi Oy. Online. <www.cgi.com/fi/fi/cgi-yrityksena>. Accessed 9 May 2021.

2 *Writing a GDPR-compliant privacy notice (template included)*. Proton Technologies AG. Online. <www.gdpr.eu/privacy-notice>. Accessed 9 May 2021.

3 *Google Analytics for Firebase Terms of Service*. Google LLC. Online. <firebase.google.com/terms/analytics>. Accessed 9 May 2021.

4 *Benefits of Analytics for Data-Driven Marketing*. Google LLC. Online. <marketingplatform.google.com/about/analytics/benefits>. Accessed 9 May 2021.

5 *Google Analytics*. Google LLC. Online. <firebase.google.com/docs/analytics>. Accessed 9 May 2021.

6 *Instant Access Now Available for Google Analytics*. Google LLC. Online. <googlepress.blogspot.com/2006/08/instant-access-now-available-for-google_15.html>. Accessed 9 May 2021.

7 *Google Analytics Usage Statistics*. BuiltWith Pty Ltd. Online. <trends.builtwith.com/analytics/Google-Analytics>. Accessed 9 May 2021.

8 *Available languages – Analytics Help*. Google LLC. Online. <support.google.com/analytics/answer/1008006?hl=en>. Accessed 9 May 2021.

9 *The Firebase Blog: Developers, meet Firebase!*. Google LLC. Online. <firebase.googleblog.com/2012/04/developers-meet-firebase.html>. Accessed 9 May 2021.

10 *The Firebase Blog: Firebase is Joining Google!*. Google LLC. Online. <firebase.googleblog.com/2014/10/firebase-is-joining-google.html>. Accessed 9 May 2021.

11 *Firebase - Crunchbase Company Profile & Funding*. Crunchbase Inc. Online. <crunchbase.com/organization/firebase>. Accessed 9 May 2021.

12 *From the garage to the Googleplex*. Google LLC. Online. <about.google/ourstory>. Accessed 9 May 2021.

13 *The Firebase Blog: September 2019*. Google LLC. Online. <firebase.googleblog.com/2019/09>. Accessed 9 May 2021.

14 *Firebase*. Google LLC. Online. <firebase.google.com>. Accessed 9 May 2021.

15 *Firebase Pricing*. Google LLC. Online. <firebase.google.com/pricing>. Accessed 9 May 2021.

16 *Firebase Documentation: Samples*. Google LLC. Online. <firebase.google.com/docs/samples>. Accessed 9 May 2021.

17 *Firebase console*. Google LLC. Online. <console.firebase.google.com/?hl=fi&pli=1>. Accessed 9 May 2021.

18 *Data collection – Firebase Help*. Google LLC. Online. <support.google.com/firebase/answer/6318039?hl=en>. Accessed 9 May 2021.

19 *How Google uses information from sites or apps that use our services – Privacy & Terms*. Google LLC. Online. <policies.google.com/technologies/partner-sites?hl=en>. Accessed 9 May 2021.

20 *Safeguarding your data - Analytics Help*. Google LLC. Online. <support.google.com/analytics/answer/6004245?hl=en>. Accessed 9 May 2021.

21 *Privacy Policy – Privacy & Terms*. Google LLC. Online. <policies.google.com/privacy?hl=en>. Accessed 9 May 2021.

22 *Data retention - Analytics Help*. Google LLC. Online. <support.google.com/analytics/answer/7667196?hl=en>. Accessed 9 May 2021.

23 *Privacy controls in Google Analytics - Firebase Help*. Google LLC. Online. <support.google.com/firebase/answer/9019185?hl=en>. Accessed 9 May 2021.

24 *Firebase integrations*. Google LLC. Online. <firebase.google.com/integrations>. Accessed 9 May 2021.

25 *How Google Makes Money (GOOG)*. About Inc. Online. <www.investopedia.com/articles/investing/020515/business-google.asp>. Accessed 9 May 2021.

26 *Analytics Events*. Google LLC. Online. <firebase.google.com/docs/reference/cpp/group/event-names>. Accessed 9 May 2021.

27 *Google Analytics for Firebase Use Policy*. Google LLC. Online. <firebase.google.com/policies/analytics>. Accessed 9 May 2021.

28 *EU user consent policy*. Google LLC. Online. <www.google.com/about/company/user-consent-policy>. Accessed 9 May 2021.

29 *Releases*. GitHub Inc. Online. <www.github.com/arnesson/cordova-plugin-firebase/releases>. Accessed 9 May 2021.

30 *Configure Analytics Data Collection and Usage*. Google LLC. Online. <firebase.google.com/docs/analytics/configure-data-collection?platform=android>. Accessed 9 May 2021.

31 *What are the GDPR Fines?*. Proton Technologies AG. Online. <www.gdpr.eu/fines>. Accessed 9 May 2021.

32 *Privacy and Security in Firebase*. Google LLC. Online. <firebase.google.com/support/privacy#global_services>. Accessed 9 May 2021.

33 *Frequently Asked Questions on the judgment of the Court of Justice of the European Union in Case C-311/18 - Data Protection Commissioner v Facebook Ireland Ltd and Maximillian Schrems*. European Data Protection Board. Online. <edpb.europa.eu/sites/edpb/files/files/file1/20200724_edpb_faqoncjeuc31118.pdf>. Accessed 9 May 2021.

34 *Best practices to avoid sending Personally Identifiable Information (PII) - Analytics Help*. Google LLC. Online. <support.google.com/analytics/answer/6366371?hl=en>. Accessed 9 May 2021.

35 *Mikä on henkilötieto?*. Tietosuojavaltuutetun toimisto. Online. <www.tietosuoja.fi/mika-on-henkilotieto>. Accessed 9 May 2021.

36 *What is considered personal data under the EU GDPR?*. Proton Technologies AG. Online. <www.gdpr.eu/eu-gdpr-personal-data>. Accessed 9 May 2021.

37 *Configure multiple projects*. Google LLC. Online. <firebase.google.com/docs/projects/multiprojects>. Accessed 9 May 2021.