HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

# Firebase Performance Monitoring

## Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2017


Autoren: Fabian Schwyter & Stefan Diegas
Betreuer: Prof. Dr. Peter Heinzmann


Arbeitsperiode: 18.9.2017 – 22.12.2017
Arbeitsumfang: 240 Arbeitsstunden bzw. 8 ECTS pro Student

# 1 MANAGEMENT SUMMARY

Firebase ist eine Mobile und Web Entwicklungsumgebung, welche von Google zur Verfügung gestellt wird. Mit den angebotenen Produkten vereinfacht Firebase die Entwicklung von Applikationen für die Softwareentwickler.

Da Firebase an der HSR noch nicht ausführlich behandelt wird, sollten im Rahmen der vorliegenden Studienarbeit einerseits Unterlagen für den Einsatz im Unterricht erarbeitet werden. Andererseits sollten mithilfe von Firebase verschiedene Performance Daten aufgezeichnet und ausgewertet werden.

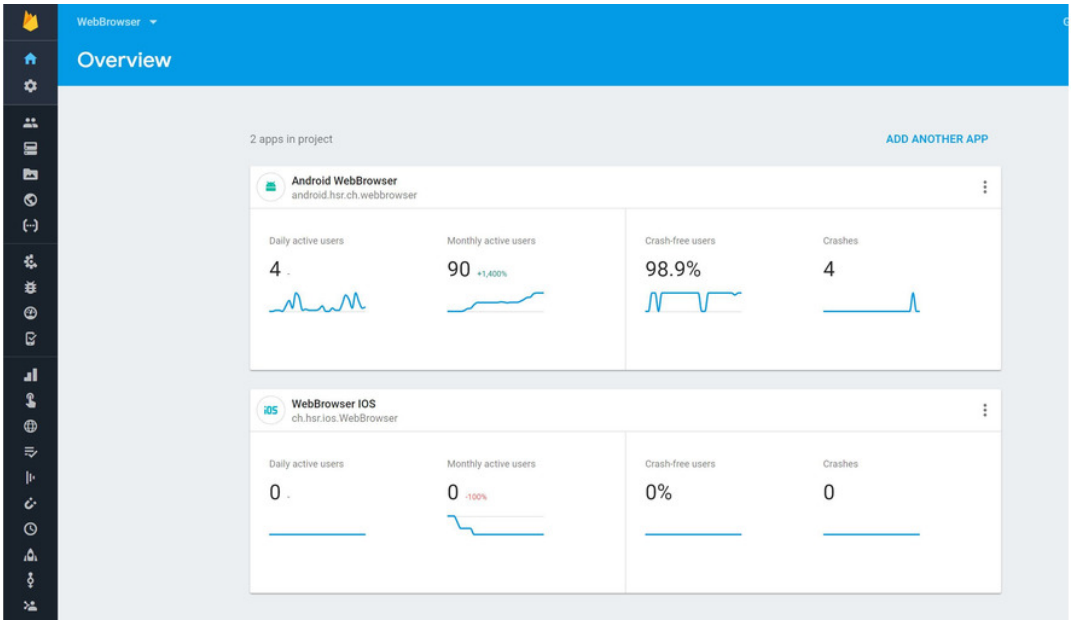



*Abbildung 1: Firebase Konsole*

Nach dem Kennenlernen von Firebase anhand einer Chat-Anwendung war das Wissen für den zweiten Teil der Arbeit gegeben. Es wurde eine Browser App programmiert, welche mit Hilfe von Firebase diverse Daten aufzeichnet. Für die Implementation des Browsers wurde die Android WebView verwendet. Das Aufzeichnen der Daten wurde mit Analytics Events, Performance Traces und automatischen Performance Network Requests realisiert. Jeder einzelne Request der WebView wurde abgefangen und ein neuer Request wurde mit OKHTTP initiiert. Firebase verlangt die Verwendung der OKHTTP Bibliothek für die automatische Aufzeichnung mit Performance Network Requests. Mit Google BigQuery werden die Analytics Events ausgewertet.

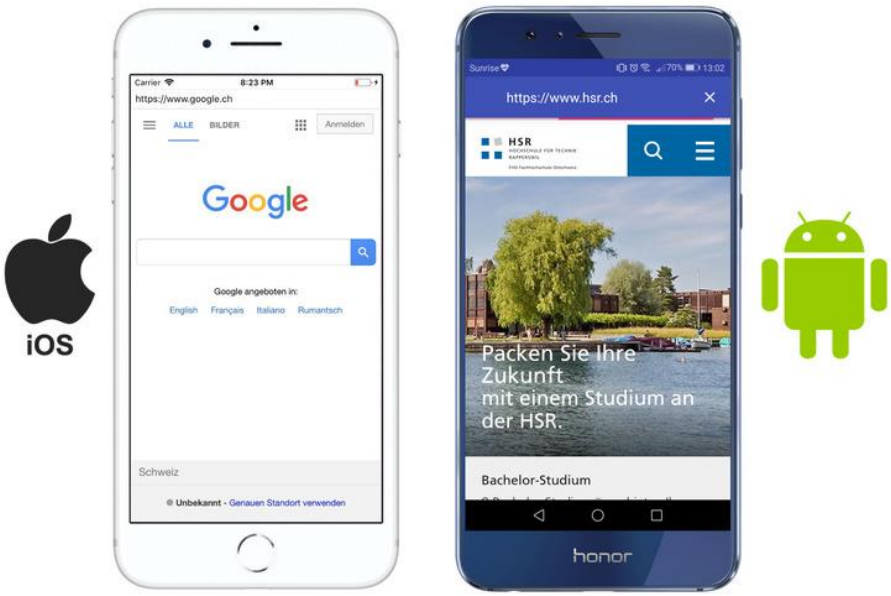



*Abbildung 2: Performance Browser IOS & Android*

Firebase wurde als Ganzes kennengelernt und die verschiedenen Funktionen wurden in kleineren Projekten verwendet. Das Wissen kann anhand einer Schulung weitergegeben werden.

Firebase Performance befindet sich noch in der Beta Phase und dürfte sich in den nächsten Releases noch stark ändern. Die entwickelte Demo-Anwendung ist für die Auswertung der Performance Daten auf die Visuelle Darstellung von Firebase limitiert. Die Performance Daten lassen sich aber auch mithilfe von Google Analytics Events erheben und auswerten. Ferner ist eine Auswertung mit Hilfe von Firebase BigQuery möglich. Aktuell sind Auswertungen in zu Benutzerdaten, Mobilfunkdaten und Webseitendaten verfügbar. So lassen sich Aussagen dazu machen, welche Webseiten mit dem Browser wie oft aufgerufen wurden (Bild 3).

Die vorbereiteten Schulungsunterlagen können an der HSR verwendet werden, um den Studenten die Möglichkeiten mit Firebase aufzuzeigen. Der Performance Web-Browser war als Demonstrator und Studienobjekt für die Arbeit vollkommen ausreichend, für die Zukunft könnte dieser aber noch verbessert werden, indem man den Support für Cookies und eine gewisse Intelligenz bei der Interpretation der Eingaben implementiert.

| Row | websitehost | website_host_count |
|---|---|---|
| 1 | www.bluewin.ch | 289 |
| 2 | www.google.ch | 77 |
| 3 | www.cnlab.ch | 16 |
| 4 | www.nzz.ch | 15 |
| 5 | maps.google.ch | 12 |
| 6 | www.youtube.com | 11 |

*Abbildung 3: BigQuery Auswertung*

# 2 AUFGABENSTELLUNG

## 2.1 AUSGANGSLAGE

Die HSR und sein Spin-Off cnlab sind bekannt für die Aktivitäten im Bereich Internet Performance Monitoring. Neben den Speedtest Programmen sind bei cnlab auch Android und iOS Speedtest Apps verfügbar. Gegenwärtig verschieben sich diese Aktivitäten zunehmend weg von reinen Netzwerk Datenraten und Antwortzeiten Messungen hin zur Erfassung von Kundenerlebnis (User Experience) Faktoren.

Firebase ist eine neue Cloud-Computing Plattform von Google zur Entwicklung, Erweiterung, Verwaltung und Vermarktung von Apps. Firebase umfasst viele Entwicklungs-, Werbe und Verrechnungskomponenten. Dazu gehört die Firebase Analytics Komponente, welche ähnlich wie Google Analytics Informationen über die Nutzer, die Nutzungsart und die Auseinandersetzung der Nutzer mit Apps liefert. Es geht dabei nicht nur um Page Views oder Sessions. Firebase Analytics fokussiert auf Ereignisse (Events) wie beispielsweise App Installation, Deinstallation, App Starts, Abstürze, Antwortzeiten aber auch User Engagement, Retention und Revenue Werte sowie demografische Daten und User Interessen.

## 2.2 ZIEL

Im Rahmen dieser Arbeit geht es zuerst darum, Firebase und vor allem die Performance Monitoring SDK Funktionen genau kennen zu lernen. Firebase soll danach in Android und iOS Apps eingesetzt werden, um Performance- und User Experience Faktoren zu erfassen. Es ist auch aufzuzeigen, wie die Demographie der Speedtest Nutzer aussieht. Die erfassten User Experience Messwerte sollen mit Einschätzungen echter Nutzer verglichen werden können.

## 2.3 AUFGABEN

### 2.3.1 Einarbeitung, Analyse

- Einarbeitung in Firebase App Development Plattform: Grundlagen, Funktionen, SDK, Möglichkeiten
- Vergleich mit Google Analytics
- Analyse der Android Speedtest Anwendung
- Analyse der iOS Speedtest Anwendung

### 2.3.2 Realisierung

- Firebase Anwendungen
- Nutzung/Entwicklung von Apps mit Firebase Performance Monitoring
- Auswertung von Analytics Events
- Analyse/Visualisierung mit Google Cloud BigQuery
- Inbetriebnahme, Testing
- Stunden Kurs über Firebase

## 2.4 REFERENZEN, BEISPIELE


1 Firebase https://firebase.google.com/
2 Firebase YouTube Channel www.youtube.com/user/Firebase
3 What's new in Firebase 2017 www.youtube.com/watch?v=IRk6n3M4d2E
4 Firebase Performance Monitoring firebase.google.com/docs/perf-mon/
5 Introduction to Firebase Analytics www.youtube.com/watch?v=EQJTx1nDr_Q
6 Silke Mimlich, Google Firebase: Universelle App-Plattform für Entwickler und Marketer, 13.07.2016, www.online-marketing-forum.at/experten-blog/2016/google-firebase-universelle-app-plattform-fuer-entwickler-und-marketer.html
7 Michaela Linhart, Google Firebase Analytics » So funktioniert App Tracking NEU, 21. Juni 2016, www.e-dialog.at/blog/webanalyse/app-tracking-neu-mit-google-firebase-analytics/

# 3 INHALTSVERZEICHNIS

# 4 EINLEITUNG

Firebase ist unter den Mobile-Entwicklern seit rund zwei Jahren in aller Munde. An der HSR wird das Thema nur am Rande behandelt. Diese Studienarbeit dient dazu Firebase kennen zu lernen und die Möglichkeiten von Firebase zu erläutern. Dieses Wissen soll die HSR nutzen und in Ihrem Unterricht einbringen können.

Zum Thema Firebase gab es kein Vorwissen, die Studienarbeit fängt bei null an. Weil Firebase auch für Informatik Verhältnisse ein sehr neues Produkt ist und weiterhin ständig ausgebaut wird, gibt es kaum Literatur zum Thema.

Firebase ist eine Mobile und Web Entwicklungsumgebung. Etwa 2011 boten Andrew Lee und James Tamplin im Startup «Envolve» ein Application Programming Internface (API) an, mit dem man einen Chat Service realisieren konnte. Tamplin und Lee fanden heraus, das «Envolve» nicht nur für Chat Nachrichten verwendet wurde, sondern auch um beliebige Applikationsdaten zwischen Geräten zu synchronisieren.[1]

Aus dem Startup «Envolve» gründeten Lee und Tamplin im Mai 2012 die Firma Firebase Inc., welche Google im Oktober 2014 übernahm.[2] Seitdem kauft und entwickelt Google ständig neue Funktionen und implementiert sie in Firebase. Im Oktober 2015 übernahm Google die Firma Divshot und im Januar 2017 Fabric und Crashlytics von Twitter. Diese wurden ins Firebase Team integriert.[3] [4] Divshot war ein Tool um statische Webseiten mit HTML5 zu bauen.[5] Fabric bietet Entwicklern ein grosses Angebot an Entwicklungshilfen, wie zum Beispiel Crashalytics. Crashalytics war die Grundlage von Fabric und ist ein Dienst, welcher Daten sammelt, wenn eine Applikation abgestürzt ist. Mit Funktionen, wie Crash-Reporting und der Möglichkeit zur Verwaltung von Werbung gab es Überlappungen zu den bisherigen Firebase Funktionen.[6]

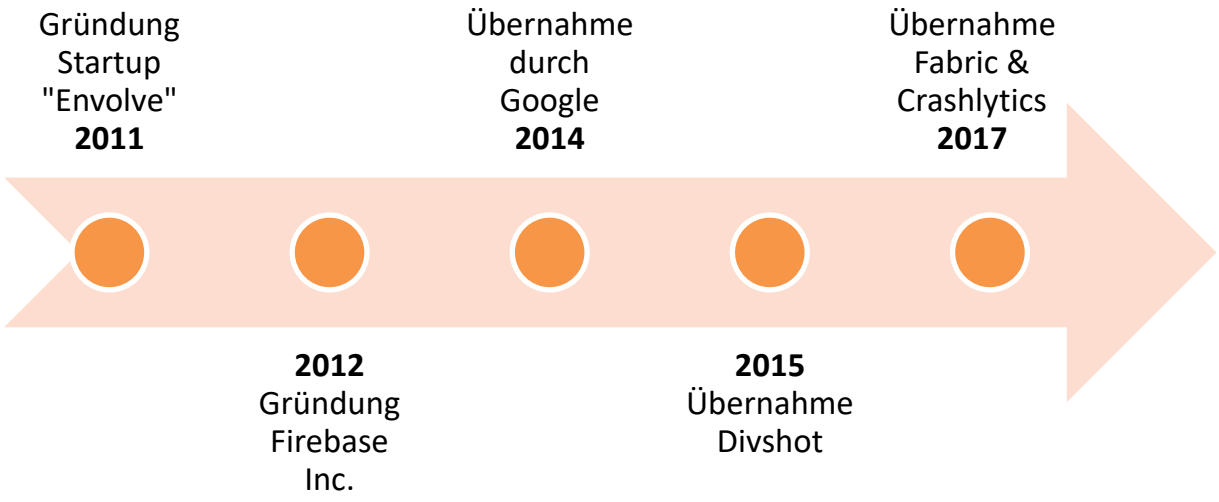



*Abbildung 4: Zeitstrahl Firebase Geschichte*

## 4.1 Produkte im Überblick

Firebase bietet Entwicklern eine Reihe von Produkten an, die immer wieder neu kategorisiert werden. Gegenwärtig (Stand 31.10.2017) unterscheidet Firebase zwischen «Develop» und «Grow» Produkte.[7]

Develop Produkte geben den Entwicklern verschiedene Bausteine, welche normalerweise als Backend genutzt werden. Die «Develop» Produkte bieten Möglichkeiten in Form von Datenpersistierung, Authentisierung, Hosting und Monitoring. Es wird auch eine Plattform Angeboten, welche es erlaubt Android Apps auf verschieden Geräten zu testen.

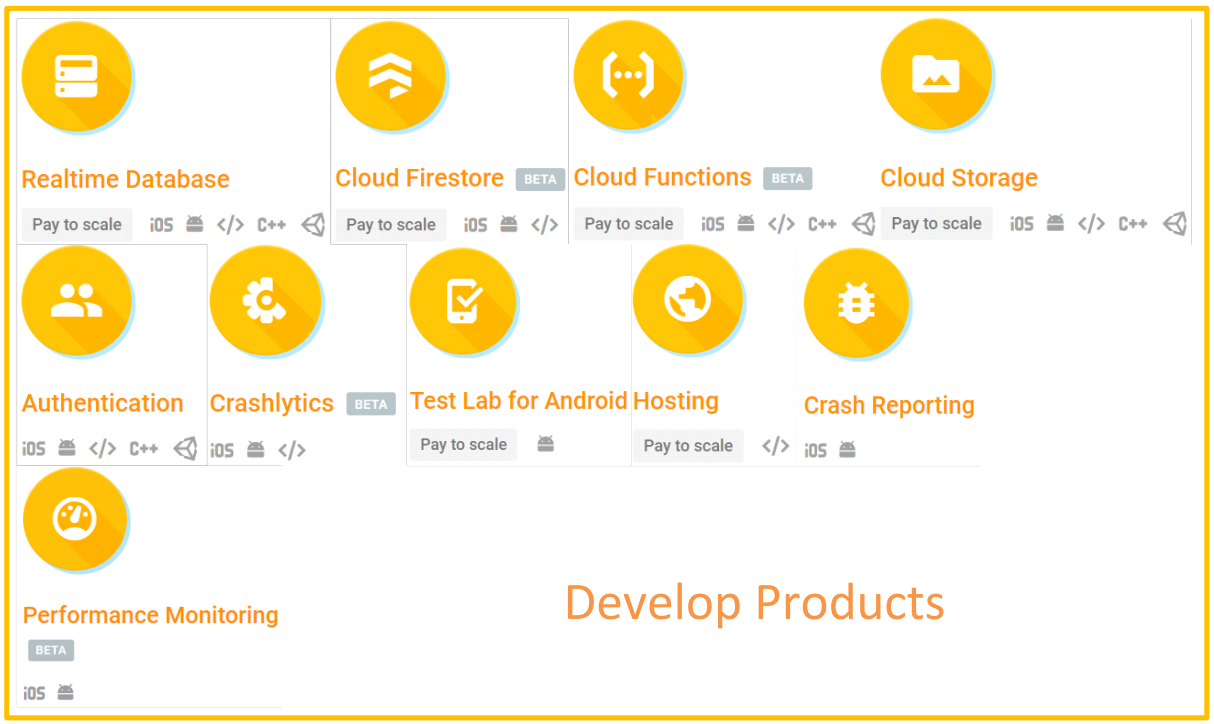



*Abbildung 5: Firebase Develop Produkte*

Grow Produkte sollen die Verbreitung von Anwendungen auf verschiedene Arten ermöglichen. Dabei gibt es Produkte die für die Verbreitung von User zu User gedacht sind, sowie Produkte die dem Unternehmen ermöglichen mit den Benutzern zu kommunizieren und so Ihre Applikation zu forcieren.

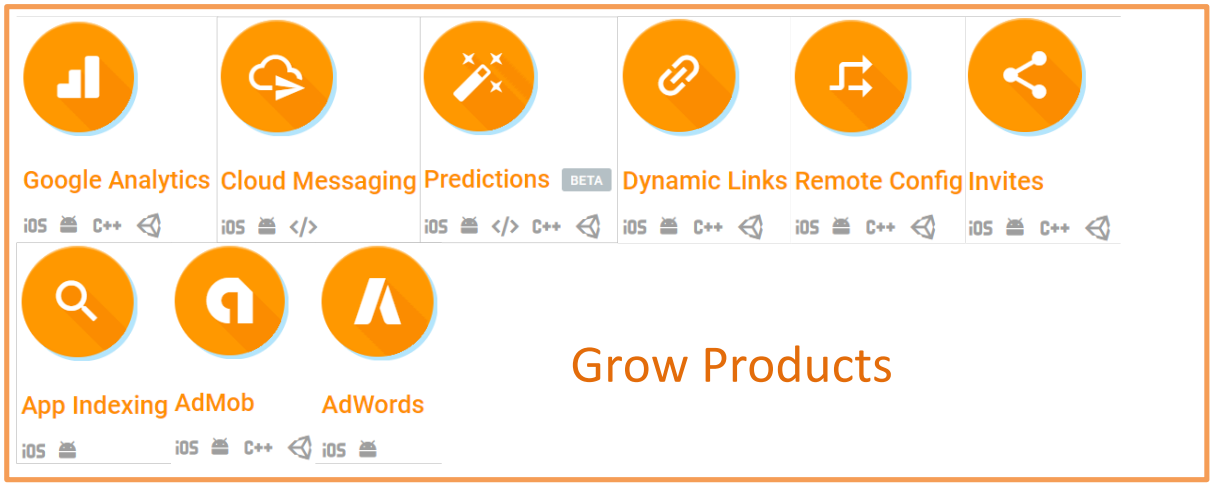



*Abbildung 6: Firebase Grow Produkte*

Anhand des Chats «FriendlyChat» von der Online Kurs Plattform Codelabs[8] werden die Firebase Produkte gut beschrieben und eingesetzt. Im Hauptabschnitt wird FriendlyChat verwendet um die Implementierung und Anwendung von Firebase zu erläutern.

# 5 FIREBASE ERKLÄRT MIT FRIENDLYCHAT

Mit FriendlyChat[8] gibt es ein Tutorial, in welchem eine einfache Chat App programmiert wird, welche viele Firebase Funktionen benutzt. FriendlyChat gibt es für Android, IOS und Web. Für die komplette App wird auf einen grossen Teil der Firebase Features zurückgreifen und Anhand dieser Beispiele Firebase erklärt.

Im Rahmen der Studienarbeit wurde eine zweistündige Vorlesung erarbeitet, die anhand FriendlyChat die Firebase Funktionen Schritt für Schritt erklärt.

## 5.1 KONFIGURATION KONSOLE

Der Einstieg für die Firebase Konfiguration ist die Firebase Konsole, welche über einen Browser erreichbar ist. Unter https://console.firebase.google.com kann man das erste Projekt anlegen, jedoch ist die Vorbedingung, dass man einen Google Account hat. Nach der Erstellung des Projektes, kann man direkt beginnen das SDK in einer App für IOS oder Android einzurichten.

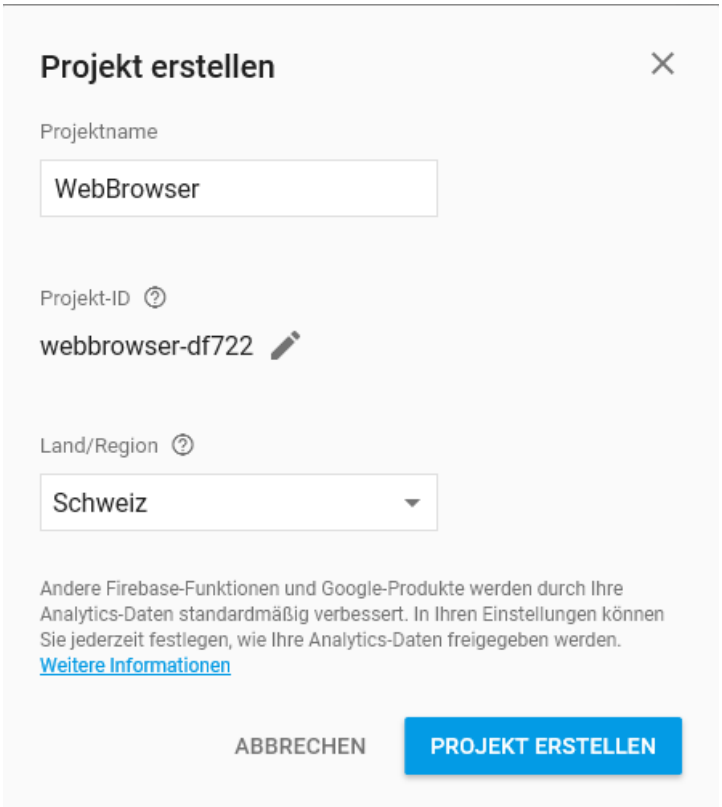



*Abbildung 7: Neues Projekt erstellen in der Konsole*

## 5.2 FIREBASE SDK

Für die Verwendung von Firebase muss jeder App die Firebase SDK hinzugefügt werden. Das Software Development Kit bietet den lokalen Zugang zu Firebase an. Um Firebase mit allen seinen Produkten verwenden zu können, muss der Client ausserdem über eine Internetverbindung verfügen. Firebase selber läuft auf Google Servern, die auf der ganzen Welt verstreut sind.

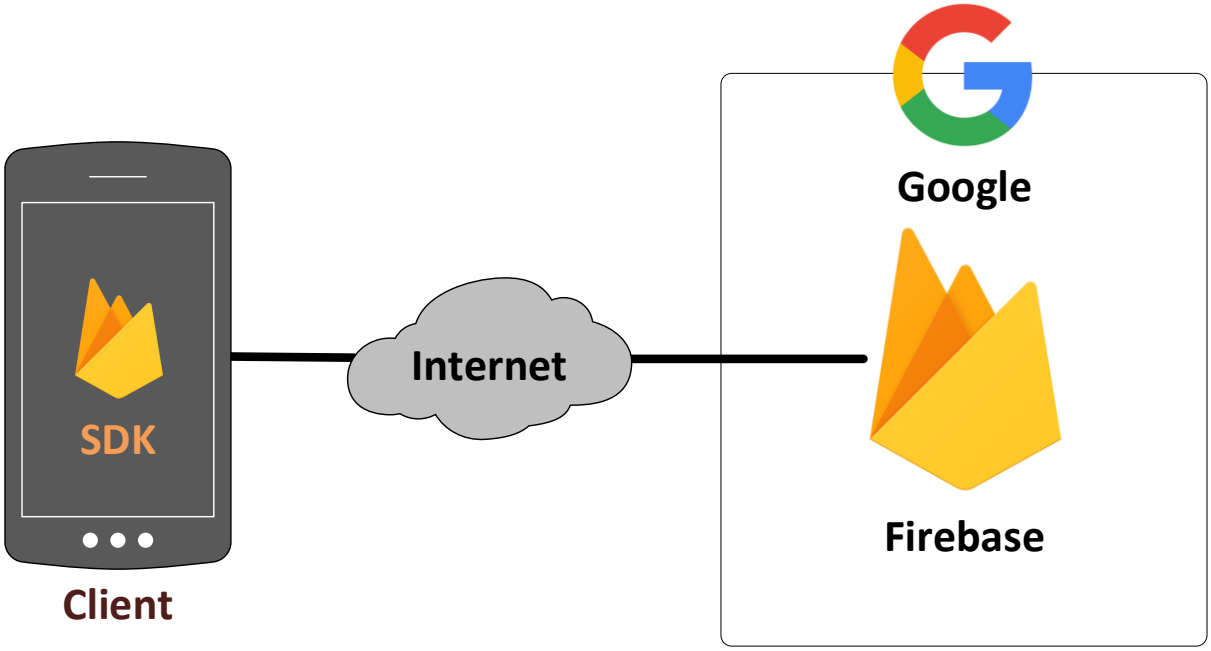



*Abbildung 8: Firebase Systemübersicht*

### 5.2.1 Android Voraussetzungen

Firebase funktioniert nur auf Geräten ab Android 4.0 (Ice Cream Sandwich). Zusätzlich müssen die Google Play Services Version 11.4.2 oder neuer installiert sein. Google Play SDK muss installiert sein und Android Studio Version 2.2 oder höher als Entwicklungsumgebung. Android Studio sollte zu Firebase verbunden sein.[9]

### 5.2.2 IOS Voraussetzungen

Firebase funktioniert bei IOS nur noch ab IOS 8.0 und es wird verlangt, dass mindestens Xcode 8.0 installiert wurde. Swift muss mindestens ab dem Standard 3.0 verwendet werden.

## 5.3 IMPLEMENTATION SDK ANDROID

### 5.3.1 SDK Installation

Mit Hilfe des Android SDK Managers müssen die «Google Play Services» installiert werden. Nach erfolgreicher Installation findet man die Firebase Plugins über die Suche in den Plugins und können direkt installiert werden. Die Konfiguration des Firebase Dienstes erfolgt über die IDE «Connect to Firebase».

### 5.3.2 Google Service JSON File

Beim Hinzufügen der App zur Firebase Konsole wird eine «google-services.json» Konfigurations-Datei heruntergeladen, welche die erforderlichen Firebase Metadaten für die App enthält. Dieses File muss im gleichem Ordner wie das App-Level build.gradle File abgelegt werden.

### 5.3.3 Firebase Implementation Android

**Im Project-Level build.gradle File:**

```
dependencies {
  classpath 'com.google.gms:google-services:3.1.1' // google-services
}

maven {
  url "https://maven.google.com" // Google's Maven repository
}
```

*Code 1:Project-Level build.gradle*

**Im App-Level build.gradle File:**

```
dependencies {
  compile 'com.google.firebase:firebase-core:11.4.2'
}
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services
```

*Code 2- App-Level build.gradle*

## 5.4 IMPLEMENTATION SDK IOS

### 5.4.1 SDK Installation Cocoapods

Cocoapods ist ein Dependency Manager für Swift und Objective C Projekte. Mit Cocoapods kann man Libraries zu seinen XCode Projekten hinzufügen. Firebase kann man über Cocoapods beziehen, oder es gibt die Möglichkeit die Library manuell zu downloaden.[10]

Im Projekt Ordner wird mit Cocoapods ein neues Projekt initialisiert. Über das generierte Podfile gibt man an, welche Firebase Libraries man installiert haben möchte. Die Firebase/Core Library bildet die Core Funktionalitäten von Firebase ab. Weitere Firebase Libraries können später noch hinzugefügt werden.[11]

### 5.4.2 Google Service JSON

Wie bei Android, gibt es auch bei IOS ein File mit Metadaten für Firebase. Dieses File heisst bei IOS GoogleService-Info.plist. Dieses File enthält wichtige Daten, um eine Verbindung zum Firebase Projekt zu erstellen.

Vom GoogleService-Info.plist. File muss der Value vom Key REVERSED_CLIENT_ID in die URL «Schemes» kopiert werde, welche in den Projekteinstellungen unter URL Types zu finden sind.[12]

Das File muss zusätzlich als Target angegeben werden, damit es mitkompiliert wird, ansonsten kann das File nicht gefunden werden.[13]

### 5.4.3 Firebase Implementation IOS

Firebase kann über die Klasse FirebaseApp konfiguriert und gestartet werden, welche sich im AppDelegate.swift befindet.

**Im AppDelegate.swift File:**

```
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
    func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey:
Any]?) -> Bool {
        FirebaseApp.configure()
        return true
    }
}
```

*Code 3: AppDelegate.swift*

## 5.5 AUTHENTICATION

Für die bessere Erklärung der einzelnen Produkte, wird FriendlyChat als Hilfe genommen. Schritt für Schritt werden die Produkte anhand mögliche Verwendungen in der Chat App erklärt.

Damit Benutzer in der FriendlyChat App etwas lesen oder schreiben können, müssen sie sich zuerst einloggen. Firebase bietet mit Authentication verschiedene Möglichkeiten sich anzumelden:

- E-Mail und Passwort
- Identity Provider
    - Google
    - Facebook
    - Twitter
    - GitHub
- Telefonnummer
- Anonym

Mit Authentication entfällt Arbeit beim Entwickler. Dieser muss nicht mehr Sicherstellen, das die Benutzer Anmeldedaten sicher vor Zugriff von Dritten abgespeichert werden.

Die FriendlyChat App bietet als Login E-Mail und Passwort, sowie die Möglichkeit sich mit einem Google Account anzumelden. Diese Optionen muss man in der Konsole aktivieren und in der App implementieren.

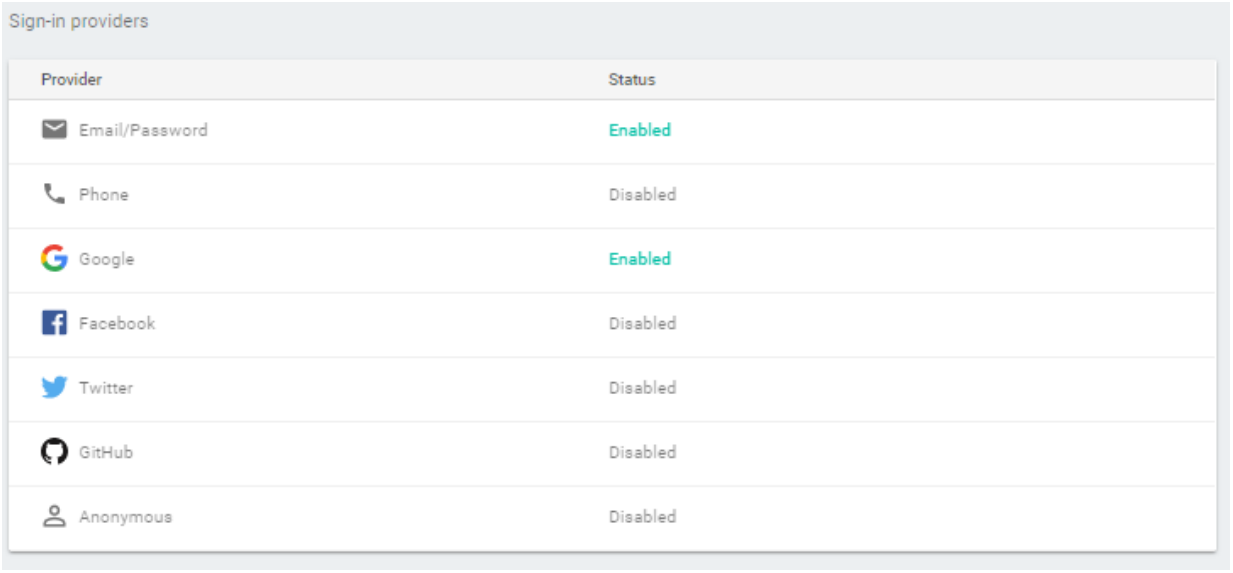



*Abbildung 9: Verfügbare Authentifizierungsmethoden*

## 5.6 REALTIME DATABASE

Firebase bietet eine Realtime Datenbank an, welche JSON-Values in einer NOSQL Datenbank abspeichert. Bei Änderungen in der Datenbank werden die Daten bei allen Clients innerhalb von Millisekunden automatisch aktualisiert. Zusätzlich kann ein Cache aktiviert werden, falls einmal ein Gerät offline ist. Die Realtime Database wurde als Ablage für Nachrichten in der FriendlyChat App ohne Cache genutzt. Nicht authentifizierte Benutzer dürfen keine Nachrichten sehen, dafür werden folgende Regeln in Firebase definiert:

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

*Code 4: Realtime Database Beispiel Konfiguration*

Bei bereits bestehenden Daten im JSON Format, besteht die Möglichkeit diese über die Konsole zu importieren. Über die API können alle authentifizierten Benutzer neue Nachrichten in der Datenbank schreiben und lesen.

## 5.7 STORAGE

Zu einer modernen Chat-Anwendung sollte das Verschicken von Bildern unterstützt werden. Bilder müssen nicht in einer Datenbank gespeichert werden, sondern sie können in Firebase Storage abgelegt werden. Im Storage können beliebige Files gespeichert werden, in diesem Fall macht FriendlyChat ein Upload eines Bildes und erstellt einen Datenbank-Eintrag mit einer Referenz auf das Bild.

## 5.8 HOSTING

Da der Chat auch übers Web erreichbar sein soll, kann das Web User Interface mit Firebase Hosting zur Verfügung gestellt werden. Das Hosting steht bis zu 1 GB statischen Content und 10 GB/Monat Transfer Daten kostenlos zur Verfügung. Am Ende des Deploy-Vorgangs wird unter Hosting ein Link zur Verfügung gestellt, mit dem auf die Weboberfläche zugegriffen werden kann.

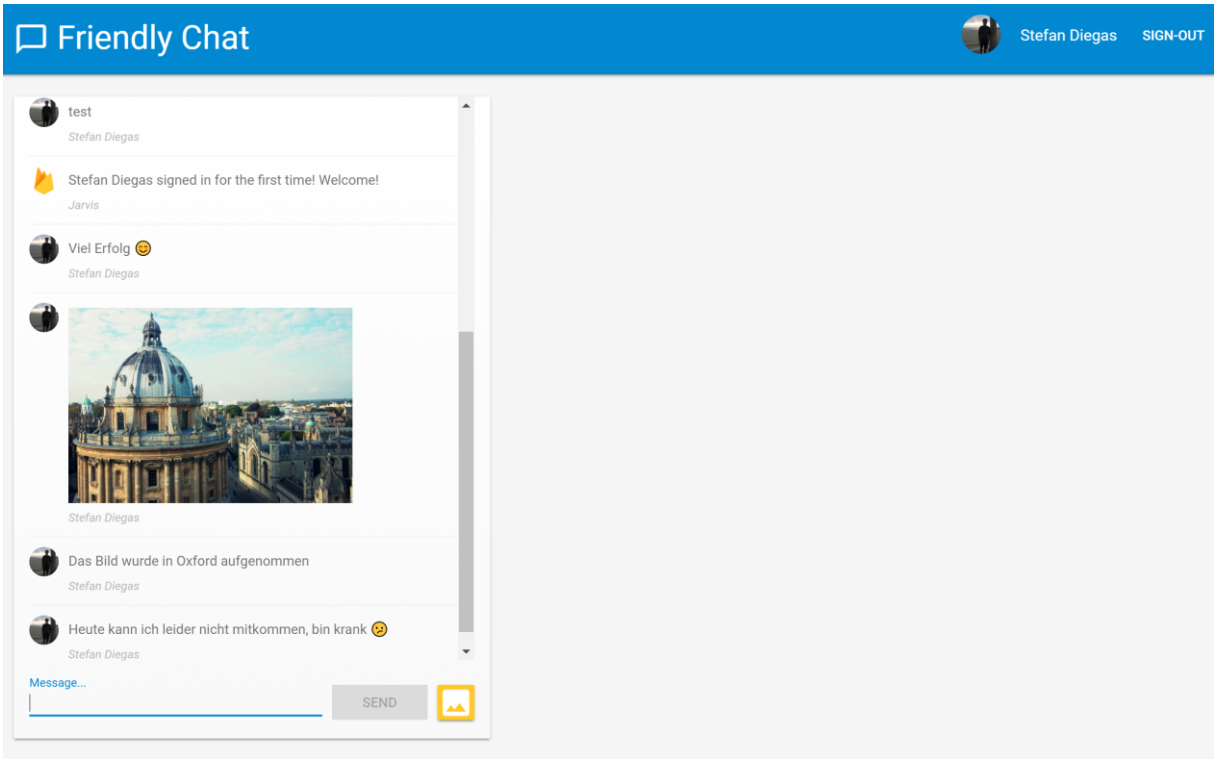



*Abbildung 10: Webanwendung FriendlyChat*

## 5.9 FUNCTIONS

Cloud Functions ermöglicht es Funktionen aufgrund von Firebase Events auszuführen, welche wiederum auf Firebase Produkte, wie zum Beispiel Cloud Messaging zugreifen. Rechenintensive Tasks können anstatt auf dem Client direkt auf Cloud Functions ausgeführt werden.

Für die FriendlyChat App sind zwei Cloud Functions definiert. Die erste Funktion dient dazu, die anderen Benutzern zu informieren, wenn jemand eine Nachricht geschrieben hat. Dazu wird auf den Event gewartet, welches durch das Schreiben auf der Datenbank ausgelöst wird. Der Inhalt und die Urheber werden dann per Firebase Notifications den anderen Benutzern mitgeteilt.

Die zweite Funktion dient dazu, die neuen Benutzer willkommen zu heissen, ein Bot namens «Jarvis» soll die neuen Benutzer durch Schreiben einer Nachricht begrüssen. Es wird darauf gewartet, dass ein Event «User Create» geworfen wird, die Willkommens Nachricht wird in die Datenbank geschrieben.

### 5.9.1 Ablauf Function

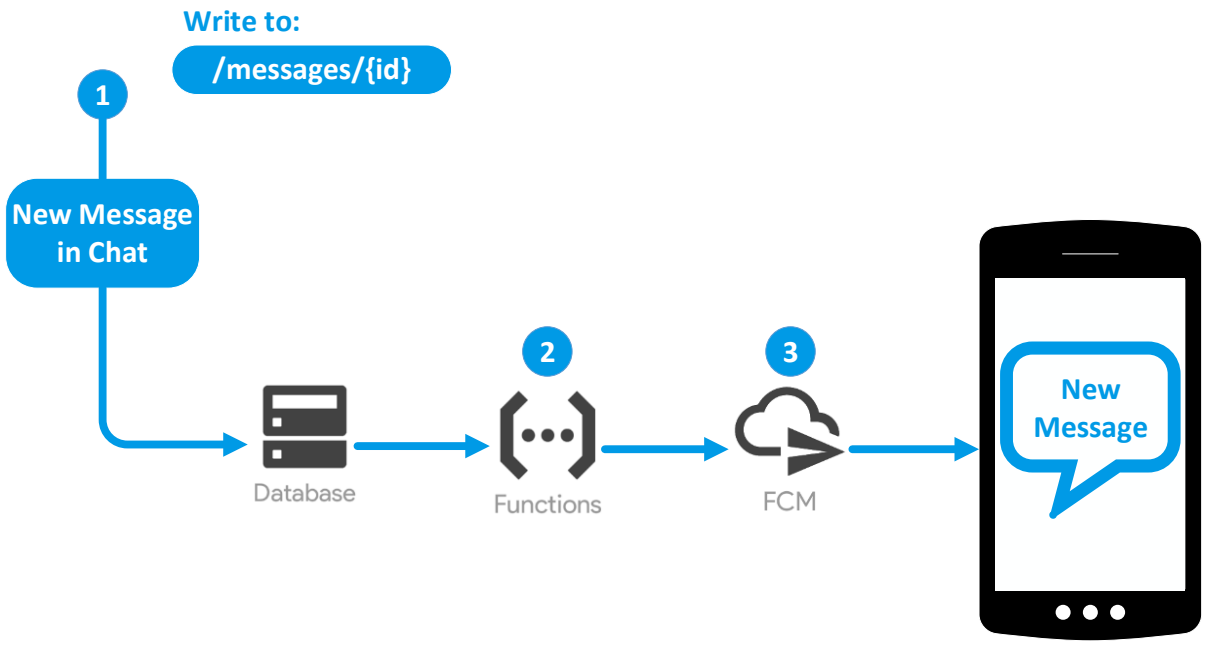



*Abbildung 11: Cloud Function Benachrichtigung bei neuen Nachrichten*

1. Eine neue Nachricht wird auf die Datenbank geschrieben.
2. Cloud Functions triggert den Event, wenn auf der Datenbank geschrieben wird.
3. Cloud Functions verwendet Firebase Cloud Messaging um alle Benutzer über die neue Nachricht zu informieren.

## 5.10 App Indexing

Mit «App Indexing» wird erlaubt, dass man mit der Google App die FriendlyChat durchsuchen kann. Dabei wird jedes einzelne Wort, welches als Nachricht verschickt wird indexiert. Sucht man nach einem bestimmten Wort in der «Google In App Suche», dann werden die Treffer der eigenen App angezeigt.

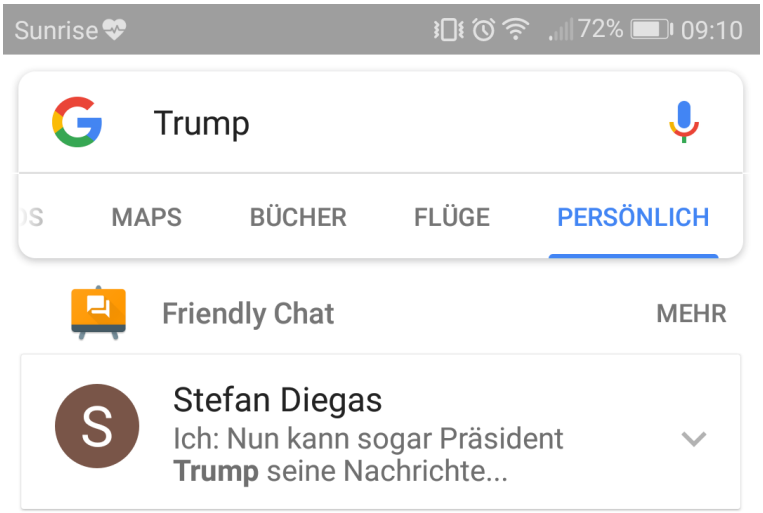



*Abbildung 12: App Indexing Android*

## 5.11 Cloud Messaging

Firebase Cloud Messaging erlaubt es Plattformübergreifend Nachrichten zu verschicken. Im FriendlyChat sollten alle benachrichtigt werden, wenn jemand eine neue Nachricht publiziert. Egal ob im Android, IOS oder im Web, die Benachrichtigung sollte überall bei einer neuen Nachricht erscheinen. Hierzu wird Firebase Functions verwendet, welche FCM benutzt um Nachrichten mit den jeweiligen Informationen zu verschicken. Auch über die Konsole können Messages verfasst und verschickt werden.

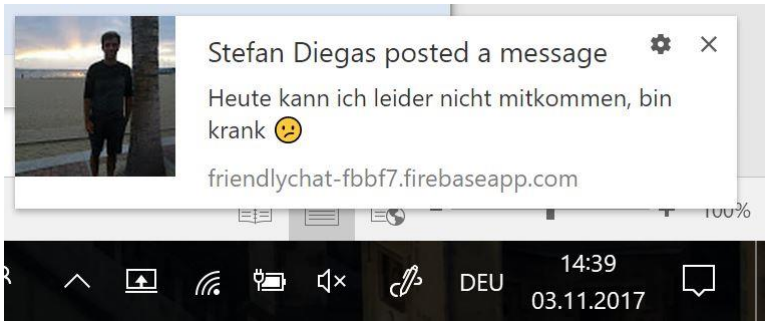



*Abbildung 13: Chrome Message*

## 5.12 Remote Config

Mit Remote Config kann man das Verhalten oder Darstellung einer App verändern ohne ein Update zu veröffentlichen. In den Applikationen können verschiedene Parameter als Variablen definiert werden, welche das Applikationsverhalten beeinflussen. In der Friendly Chat App kann somit die Nachrichtenlänge per Remote Config angepasst werden. Es besteht die Möglichkeit die Parameter über die Konsole beliebig zu verändern.

In Remote Config kann man verschiedene «Conditions» definieren, so dass die Parameter nur in einer definierten Zielgruppe verändert werden.

Es gibt folgende definierbare «Conditions»

- App Version
- OS Type
- Device Language
- Country Region
- User Audiences
- User Property
- Prediction

## 5.13 Dynamic Links

Mit Dynamic Links werden Benutzer direkt zum Inhalt in der App oder Webseite weitergeleitet. Falls die App nicht installiert wurde, dann gelangt man in den Store, wo man die App direkt installieren kann und kann man den Link öffnen und gelang auf den Inhalt.

## 5.14 Predictions

Firebase Predictions erlaubt es Vorhersagen über das Benutzerverhalten zu machen, dafür werden mittels Machine Learning Analytics Daten ausgewertet. Predictions gruppiert die Benutzer nach der gemachten Vorhersage.

Mit Predictions kann z.B. festgestellt werden, welche Benutzer wahrscheinlich in naher Zukunft die FriendlyChat App deinstallieren werden. Den Benutzern kann man mit «Remote Config» gezielt die maximale Nachrichtenlänge nach oben anpassen, um diese zum Bleiben zu überzeugen. Mit «Notifications» besteht auch die Möglichkeit, den Benutzern darauf aufmerksam zu machen, dass die App angepasst wurde.

## 5.15 Invites

Falls FriendlyChat in einem App Store zur Verfügung gestellt werden sollte, dann besteht die Möglichkeit mehr Benutzer zu gewinnen. Firebase Invites baut auf Firebase Dynamic Links. Zahlreiche Benutzerbefragungen belegen, dass Empfehlungen von anderen Benutzern einen wichtigen Einfluss darauf haben, ob eine App installiert wird oder nicht. Mit Firebase Invites bietet Google eine einfache Möglichkeit anderen Benutzern die App zu empfehlen. Dies kann entweder per SMS oder per E-Mail geschehen.

## 5.16 Crash Reporting

Eine App die regelmässig Crasht wird von einem Benutzer schnell deinstalliert und der Benutzer macht sich auf der Suche nach einer Alternative. Da man ungern User an die breite Konkurrenz verliert, wurde in FriendlyChat Firebase Crash Reporting implementiert. Crash Reporting gibt die Möglichkeit alle Exceptions der Clients zu sammeln. Auf der Konsole werden alle Exceptions zu Issues gruppiert und dann grafisch dargestellt. Falls eine Exception eintrifft, ist es möglich einen E-Mail-Benachrichtigung zu erhalten. Auch ist es möglich eigen definierte Daten zu loggen, um die Entstehung der Crashes zu identifizieren.[14]

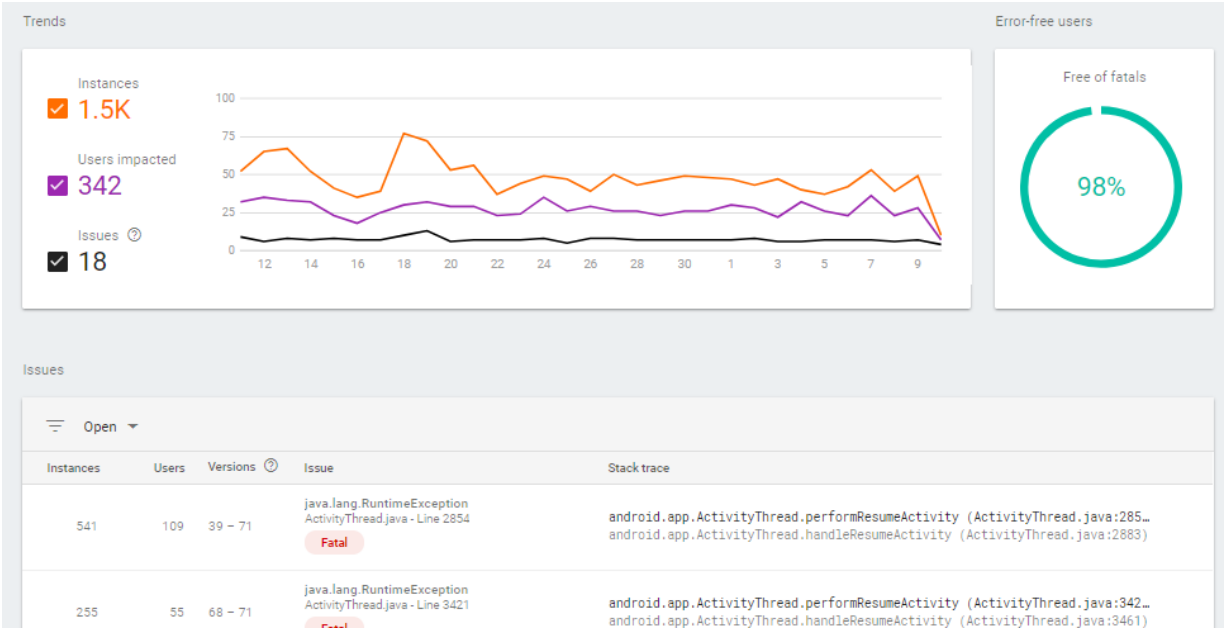



*Abbildung 14: Crash Reporting Dashboard*

## 5.17 Test Lab für Android

Für FriendlyChat wurde erfolgreich auf einem Gerät getestet, doch wie verhalten sich die Tests auf anderen Geräten mit einer anderen Android Version und einem anderen Standort? Um diese Frage zu klären, wird Test Lab verwendet. Firebase Test Lab stellt eine Cloud basierte Infrastruktur für automatisierte Test bereit. Die Tests werden auf verschiedenen physikalischen Geräten, sowie auch virtuellen Geräten ausgeführt. Neben der Konfiguration der Geräte, kann man auch die Android Version, Lokation und Orientierung auswählen. Die Testresultate mit Logs, Videos und Screenshots werden auf der Firebase Konsole zur Verfügung gestellt. Falls kein Test Code geschrieben wurde, prüft Test Lab ob Crashes produzierbar sind, dafür wird «Robo Test» verwendet. «Robo Test»[15] analysiert die Struktur des User Interfaces und simuliert anschliessend Benutzergesten.[15]

## 5.18 Performance Monitoring

Firebase Performance Monitoring dient dazu die Performance von Apps zu messen und auf einen Blick Engpässe zu erkennen. Performance Monitoring bietet die Möglichkeit, in der App Traces zu machen um Beispielsweise die Startup Time zu messen.

Ein weiterer Ansatz ist, das automatische Messen von «Network Requests» zu Services, auf welche die App zugreift. So können langsame oder nicht funktionierende Services identifiziert werden. Besonders in Regionen mit langen Latenzzeiten, können so einfach Flaschenhälse identifiziert werden.

Firebase Performance ist stand Heute (23.10.2017) erst in der Beta Release erhältlich. Es wurde bis jetzt von Google noch nicht alles dokumentiert. Die Dokumentation von IOS in der API ist nicht vorhanden.

### 5.18.1 Dashboard

Das Dashboard dient dazu sich einen Überblick über die Performance einer App zu verschaffen. Neben der «Network success rate» werden auch andere Kennzahlen wie die «Network response latency» nach Land angezeigt, Traces und vieles mehr.[16]

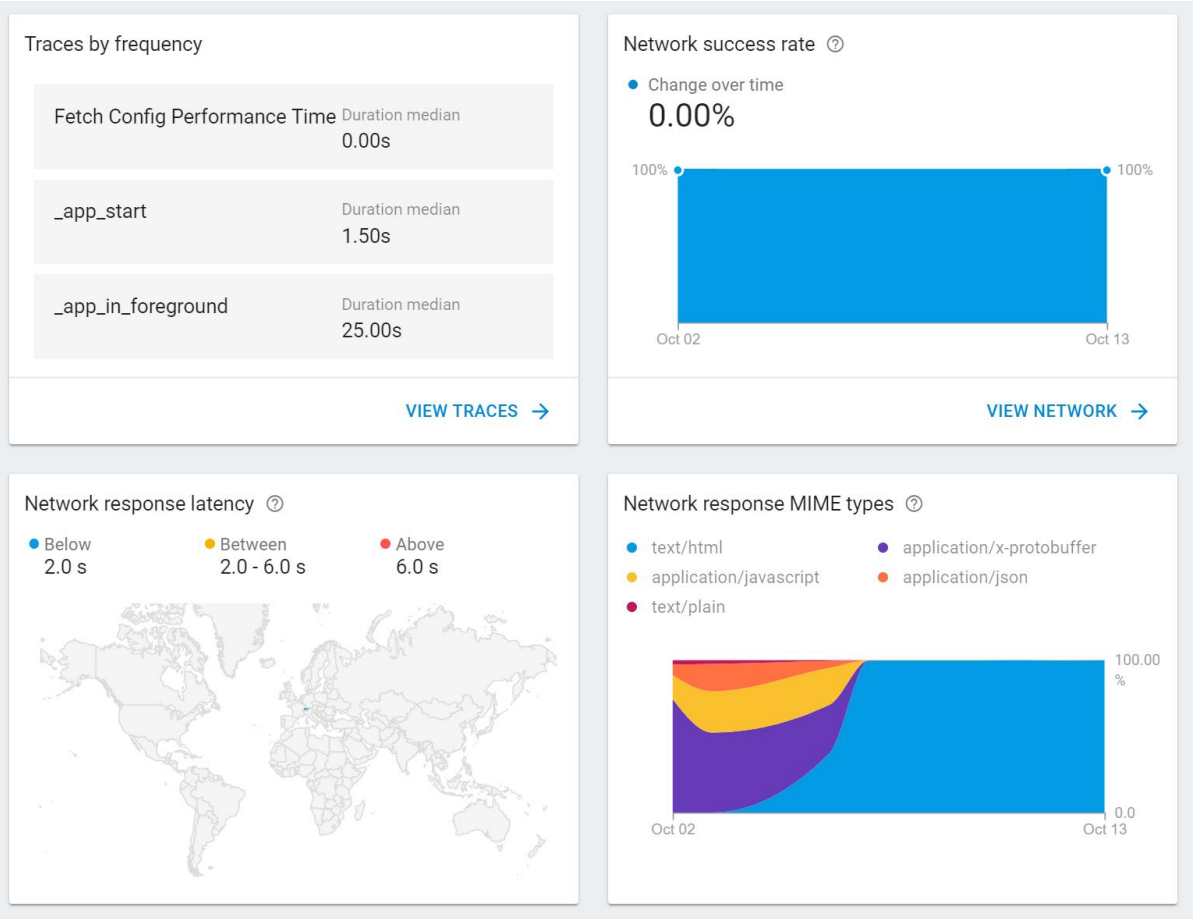



*Abbildung 15: Firebase Performance Dashboard Übersicht*

### 5.18.2 Traces

Traces erlauben die Performance einer App zu messen. Traces haben 2 Methoden. Es kann die Zeit gemessen werden und es gibt einen Counter. Der Counter kann zwischen den Start und Stop Zeiten inkrementiert werden. Firebase hat schon folgende drei automatischen Traces implementiert:[17]

| Trace Name | IOS | Android |
|---|---|---|
| App start | Der Trace startet, wenn die App das erste Objekt ins Memory lädt und stoppt nach dem ersten erfolgreichem loop nachdem die App die UIApplicationDidBecomeActive-Notification erhält. | Startet wenn die Apps FirebasePerfProvider ContentProvider seine onCreate() Methode abschliesst und stoppt wenn die erste Activity onResume() aufruft. |
| App in background | Startet wenn die App die UIApplicationWillResignActive-Notification erhält und stoppt wenn UIApplicationDidBecomeActive-Notification erhalten wurde. | Startet wenn die letzte Activity die Methode onStop() aufruft und stoppt wenn die erste Activity onResume() aufruft. |
| App in foreground | Startet wenn die App die UIApplicationDidBecomeActive-Notification erhält und stoppt, wenn UIApplicationWillResignActive-Notification erhalten wurde. | Startet wenn die erste Activity onResume() aufruft und stoppt wenn die letzte Activity onStop() aufruft. |

*Tabelle 1: Standard Traces*

Die Traces besitzen alle einen Namen und einen Mittelwert der Stoppuhr. In der letzten Spalte ist angegeben, wie oft dieser Trace ausgeführt wurde.

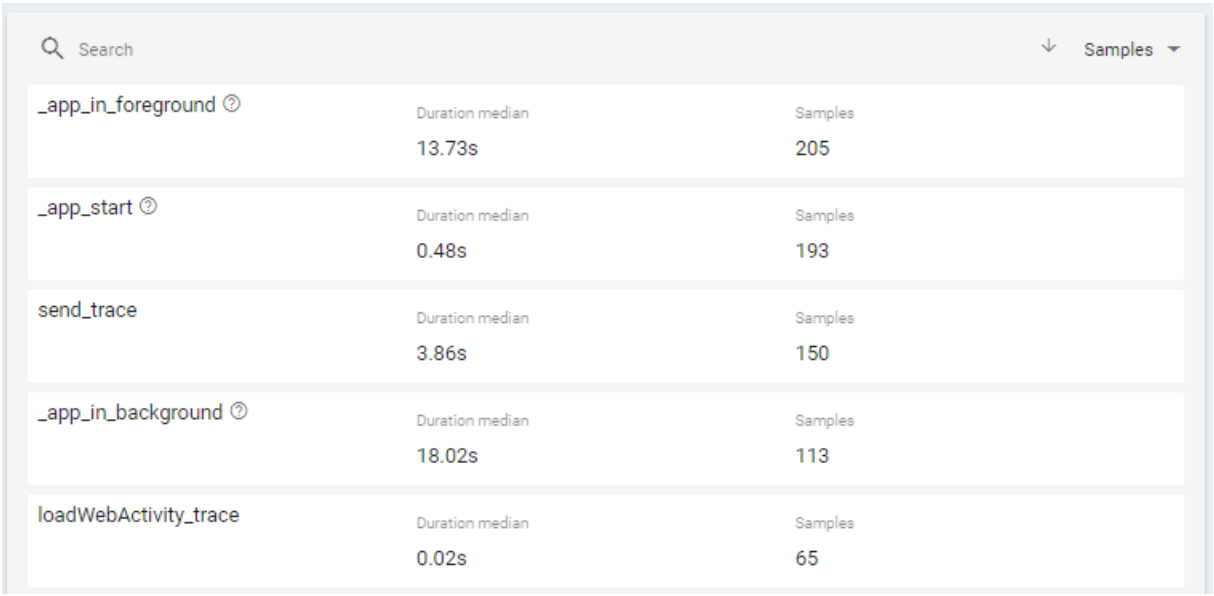



*Abbildung 16: Trace Übersicht*

### *5.18.2.1 Implementation Trace*

Für die FriendlyChat App kann ein Trace definiert werden, der beim Klicken des Sende-Buttons eine Stoppuhr startet. Sobald die View mit den aktuellsten Daten aktualisiert wird, dann wird der Trace gestoppt. Dadurch wird gemessen, wie lange es dauert, bis die eigene Nachricht angezeigt wird. Man könnte auch einen Counter beim Klicken des Absende-Buttons implementieren um festzustellen wie viele Male ein Benutzer auf senden klickt.

```
//Declaration on top of Activity
public Trace sendTrace;
//Initialisation in the onCreate() method
sendTrace = FirebasePerformance.getInstance().newTrace("send_trace");
//Start Trace and Stop Trace(Timer)
sendTrace.start();
sendTrace.stop();
//Increment Trace(Counter)
myTrace.incrementCounter("send_button_hit");
```

*Code 5: Beispiel Trace Android*

### 5.18.3 Network Requests

Google verspricht mit Firebase automatisch alle HTTP und HTTPS Requests zu erfassen und Performance Daten auszuwerten. In den Issues wird jedoch auf einige Einschränkungen hingewiesen. Bei Android werden die automatischen Network Traces nur gemacht, falls der OKHTTP Client ab Version 3.0.0 verwendet wird.[18] Bei IOS werden Requests mit URLConnection nicht getraced. In den aktuellen IOS Performance Issues (19.12.2017) ist dieser Issue nicht mehr vorhanden.[19]

Die Auswertungen der Netzwerkanfragen enthalten als Titel den Host der URL, eine mittlere Response Time und eine Success Rate. Zusätzlich wird angegeben, wie häufig eine bestimmte URL aufgerufen wurde.

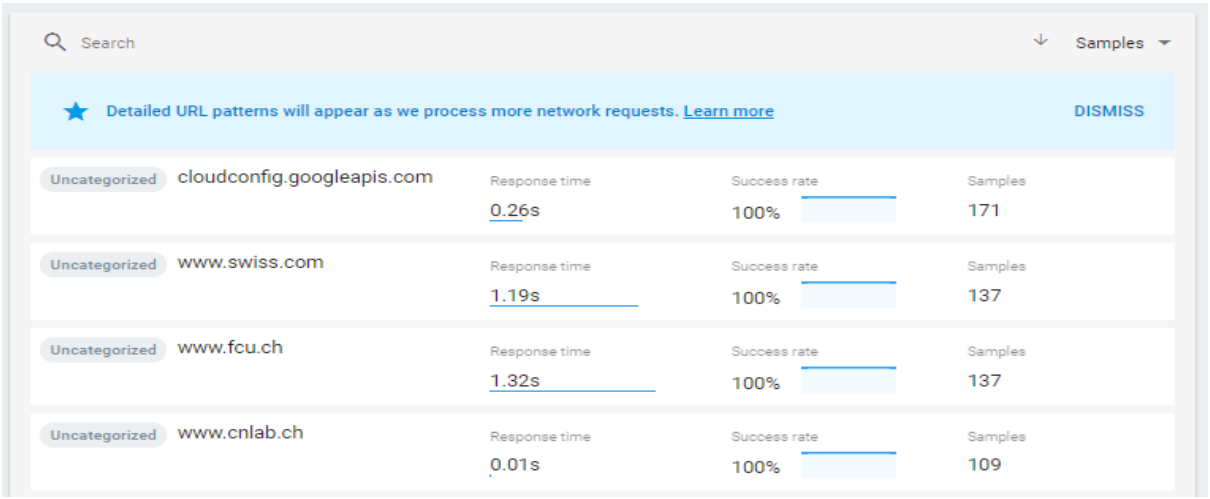



*Abbildung 17: Network Requests Übersicht*

In der detaillierten Ansicht werden die folgenden Auswertungen grafisch angezeigt.

- Durchschnittliche Response Time
- Durchschnittliche Response Payload Size
- Durchschnittliche Request Payload Size
- Success Rate

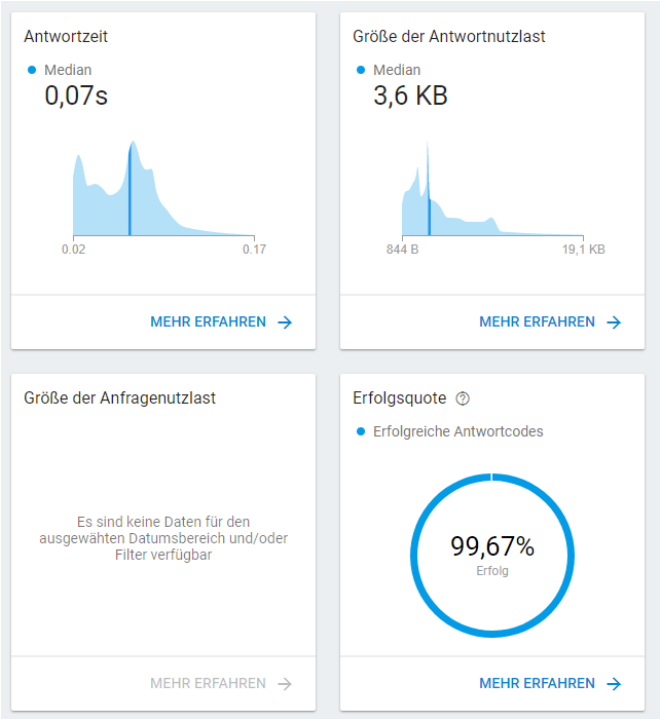



*Abbildung 18: Erweiterte Ansicht Netzwerk Anfragen*

Die Success Rate gibt an, ob die Requests erfolgreich waren. Ob die Requests erfolgreich waren hängt vom Indikator Response Code ab. Die HTTP Response Codes 4xx und 5xx werden als Fehler angezeigt.

Diese Werte kann man genauer anschauen und nach folgenden Dimensionen kategorisieren:

- App Version
- Country
- OS Version
- Device
- Radio
- Operator
- MIME Type

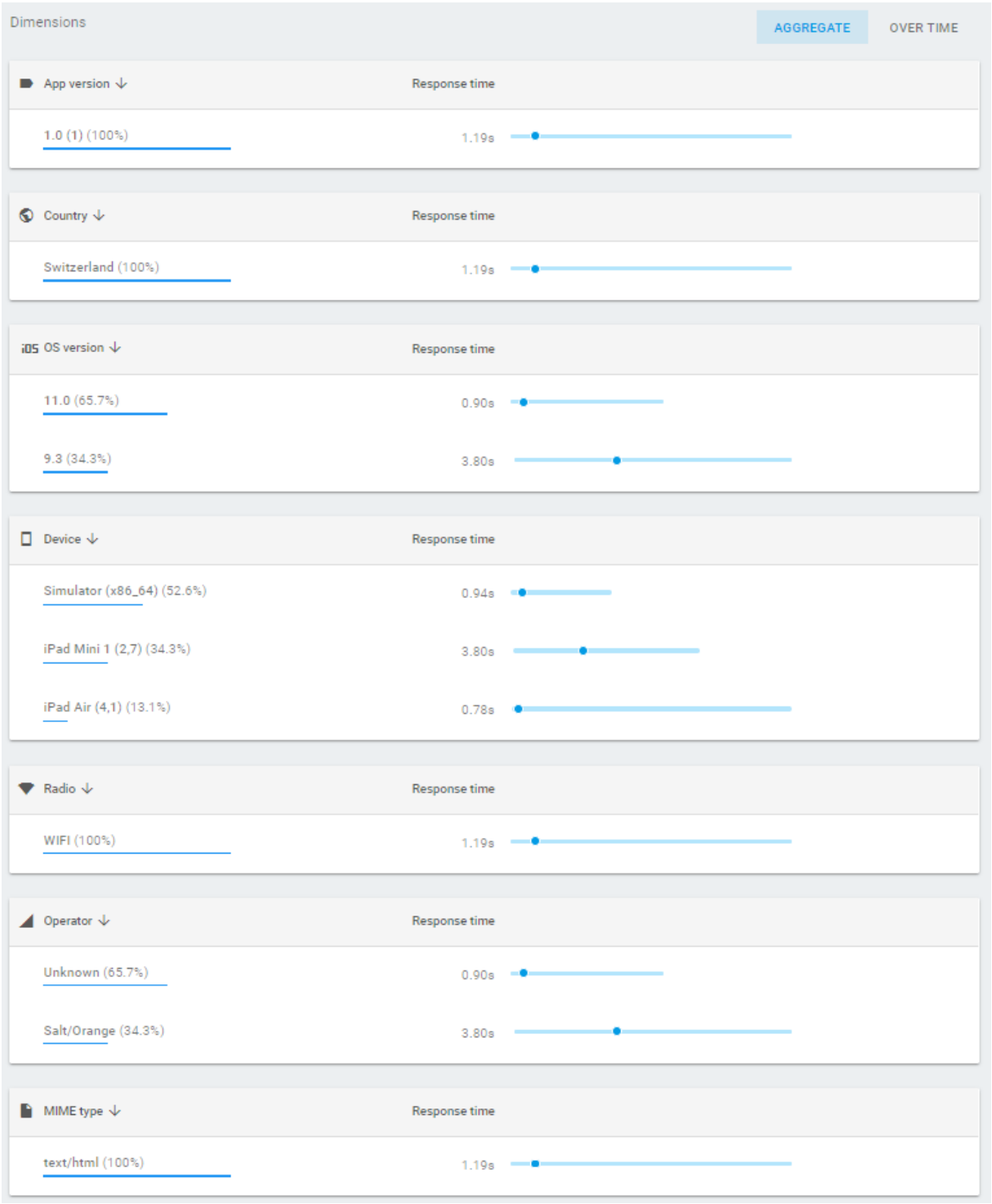



*Abbildung 19: Network Request Response Time Detailansicht*

## 5.19 Analytics

Google Analytics wurde in Firebase integriert und ist seitdem eine Kernfunktion von Firebase. Damit ist es möglich in Apps das Kundenverhalten aufzuzeichnen und anschliessend auszuwerten. In der Konsole gibt es für jede App, wie in den anderen Firebase Produkten, eine getrennte Ansicht der Daten. Die einzelnen Funktionen werden unten genauer beschrieben.[20]

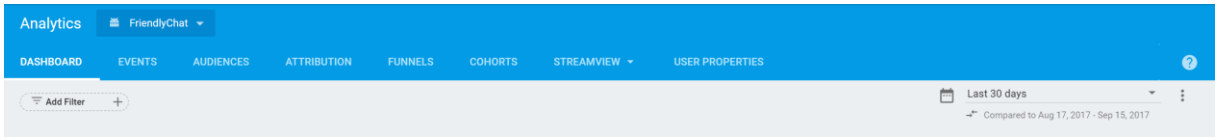



*Abbildung 20: Analytics Tabs*

### 5.19.1 Dashboard

Das Dashboard dient dazu sich einen Überblick der App zu verschaffen. Die wichtigsten Kennzahlen wurden bereits aufbereitet und grafisch dargestellt. Für einen vertieften Blick in die Daten, kann je nach Angebot, auf die passenden Links geklickt werden.

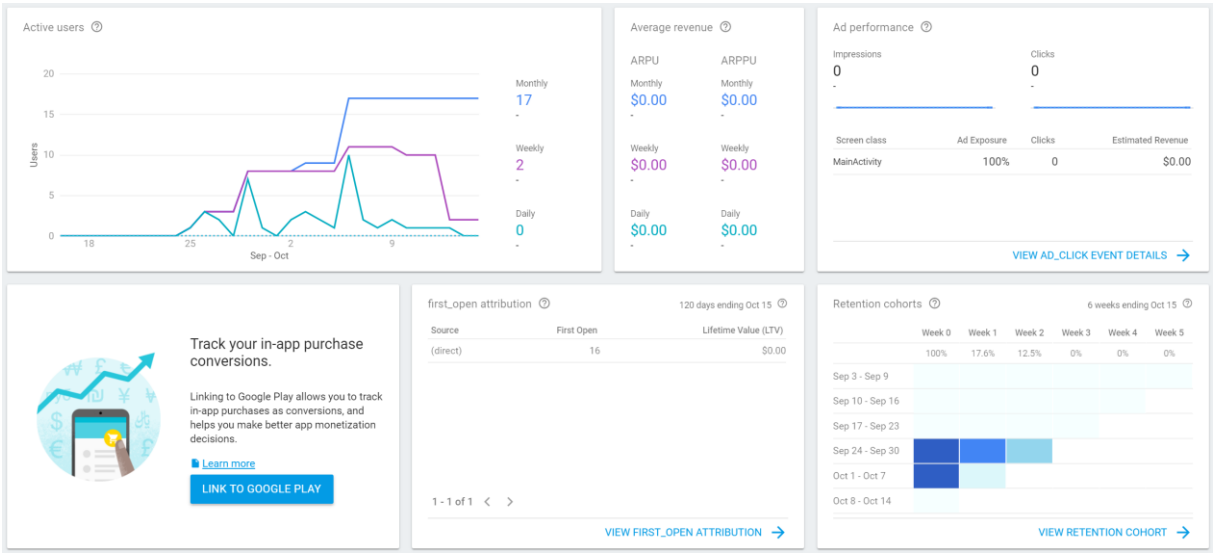



*Abbildung 21: Dashboard Übersicht*

### 5.19.2 Events

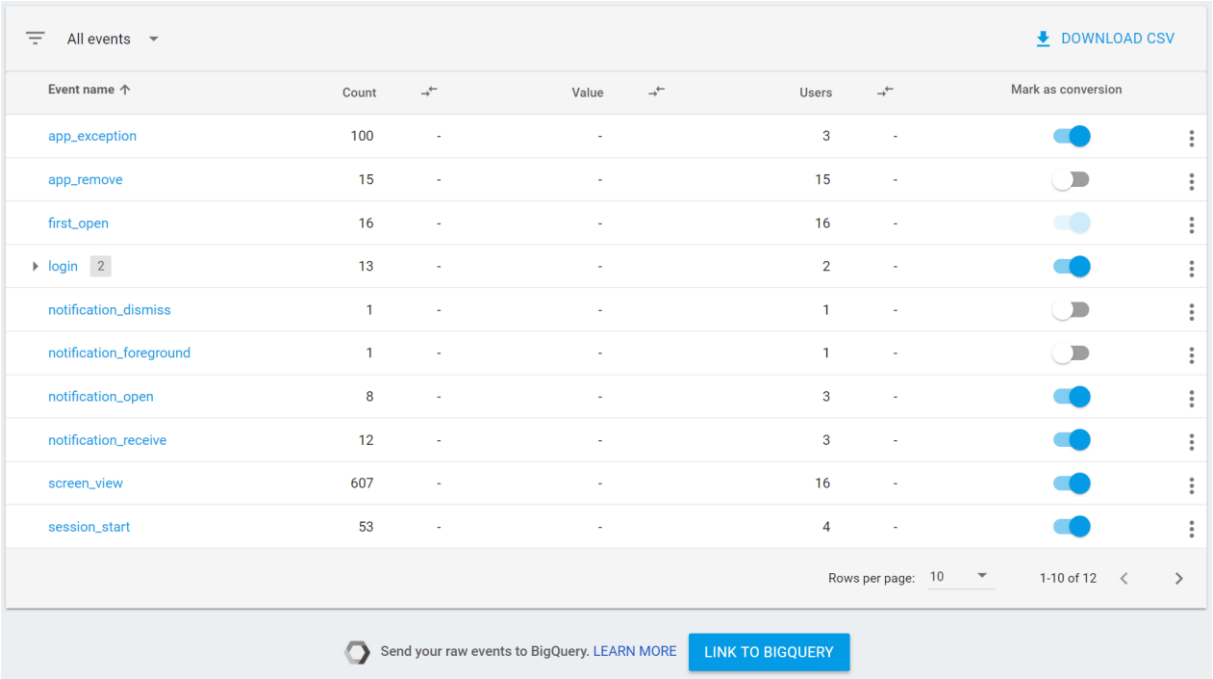



*Abbildung 22: Events Übersicht*

Events ermöglichen einen Einblick in die App und können Benutzereingaben, System Ereignisse oder Fehler aufzeichnen. Google Analytics loggt folgende Events automatisch für den Entwickler: [21]

| Event Name | Auslösung | Parameter |
|---|---|---|
| first_open | Das erste Mal in dem ein User die App nach der Installation öffnet. | |
| in_app_purchase | Jedes Mal, wenn ein Benutzer einen in-App Einkauf tätigt, welcher entweder über den App Store in iTunes oder über Google Play verarbeitet wird. | • Produkt ID<br>• Produkt Name<br>• Währung<br>• Quantität |
| user_engagement | Periodisch während die App im Vordergrund ist. | |
| session_start | Wenn ein Benutzer mit der App arbeitet. | |
| app_update | Wenn die App auf eine neue Version upgedatet wird und anschliessend gestartet wird. | • Vorherige App Version |
| app_remove | Wenn die App auf dem Gerät deinstalliert wird. | |
| os_update | Wenn das OS upgedatet wird. | • Vorherige OS Version |

| | | |
|---|---|---|
| app_clear_data | Wenn der Benutzer die App Daten löscht, alle Einstellungen und Login Daten entfernt. | |
| app_exception | Wenn die App abstürzt oder eine Exception wirft. | |
| notification_foreground | Wenn eine Benachrichtigung von Firebase Notifications erhalten wird während die App im Vordergrund ist. | |
| notification_receive | Wenn eine Benachrichtigung von Firebase Notifications erhalten wird. | |
| notification_open | Wenn eine Benachrichtigung von Firebase Notifications geöffnet wird. | |
| notification_dismiss | Wenn eine Benachrichtigung von Firebase Notifications abgewiesen wird. | |
| dynamic_link_first_open | Wenn der Benutzer die App zum ersten Mal über einen dynamic link öffnet. | |
| dynamic_link_app_open | Wenn der Benutzer die App über einen dynamic link öffnet. | |
| dynamic_link_app_update | Wenn der Benutzer die App nach einem Update zum ersten Mal über einen dynamic link öffnet. | |

*Tabelle 2: Standard Analytics Events* [22]

#### *5.19.2.1 Eigenschaften*

Standardmässig werden zu jedem Event Systeminformationen geschickt. Jedoch erlauben Events auch die Mitgabe beliebiger Parameter, welche dann in der Konsole auch ersichtlich sind. Die mitgeschickten Systeminformationen sind mit BigQuery abrufbar.

#### *5.19.2.2 Beschränkungen*

- Maximal 500 Events pro App
- Maximal 25 Parameter pro Event
- Maximal 100 Zeichen Value pro Parameter

#### *5.19.2.3 Implementation Event*

Für FriendlyChat wurde ein Event deklariert, welches ausgelöst wird, wenn ein Benutzer sich einloggt. Dieser Event sollte am Ort im Code eingefügt werden, an welchem sich der Benutzer einloggt. Als Namen des Events wurde «login» gewählt, zudem werden «login_name» und «login_email übermittelt».

```
//Declaration on top of Activity
private FirebaseAnalytics mFirebaseAnalytics;
//Initialisation in the onCreate() method
mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);
//Bundle creation, adding some parameters and send Event
Bundle params = new Bundle();
params.putString("login_name", personName);
params.putString("login_email", personEmail);
mFirebaseAnalytics.logEvent("login", params);
```

*Code 6: Beispiel Event Android*

### 5.19.3 Audiences

Audiences ermöglicht es Benutzer anhand von Events und Benutzereigenschaften zu gruppieren. Bereits implementiert sind die Gruppen «Purchasers» und «All Users». Mit Audiences besteht die Möglichkeit beliebige Untergruppen zu bilden und diese können für folgendes gebraucht werden:[23]

- Benachrichtigung der Gruppe über Firebase Notification.
- Gezielte Remote Konfiguration von Parametern über Firebase Remote Config.
- Anhand von Untergruppen gefilterte Reports erstellen.

### 5.19.4 Attribution

Attribution ist ein umfangreiches Thema, hier wird grob beschrieben was der Nutzen von Attribution ist. Angenommen eine App stellt einen Werbebanner zur Verfügung, wenn die App gestartet wird und der Webebanner initialisiert wurde, fragt das Handy nach personalisierter Werbung.[24]

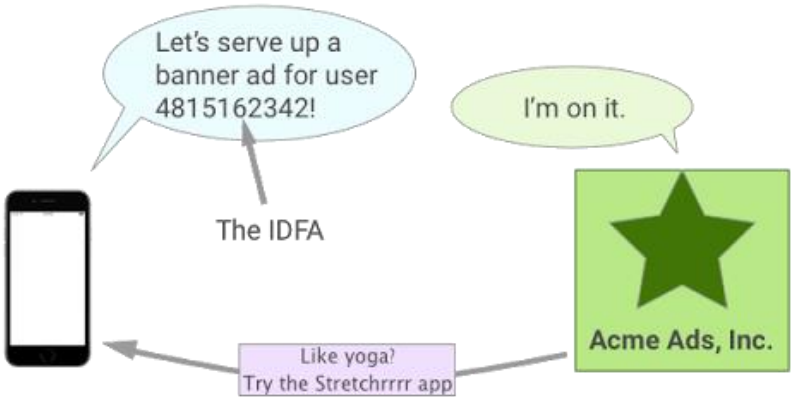



*Abbildung 23: Attribution Werbung anfordern (Quelle: firebase.googleblog.com )*[24]

Angenommen der Benutzer klickt nun auf die Werbung und installiert die App, dann wäre es hilfreich zu wissen, welche Benutzer aufgrund der Werbekampagne die App installiert haben. Hier kommt Firebase Analytics mit Attribution ins Spiel. Sobald ein Benutzer auf den Werbebanner eines Werbe-Netzwerks klickt, wird Firebase darüber informiert. Bereits über 50 grosse Werbefirmen sind bereits registriert. Diese stellen Firebase anschliessend Ihre Daten zur Verfügung.

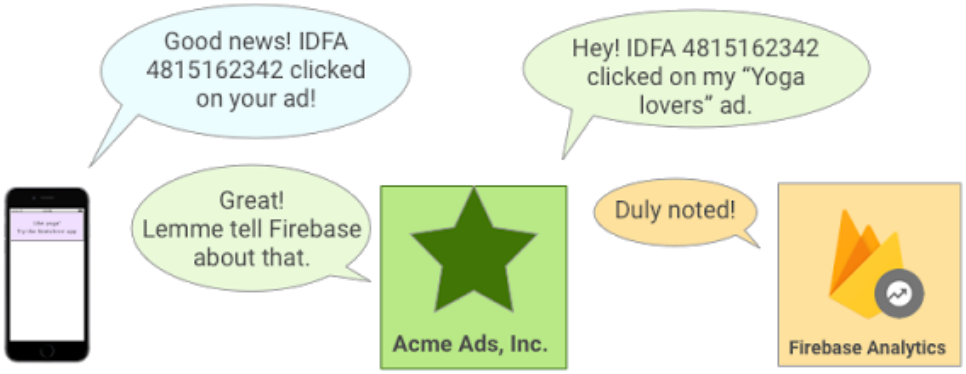



*Abbildung 24: Attribution Benachrichtigung Firebase (Quelle: firebase.googleblog.com)*[24]

Wenn jemand in der installierten App aufgrund von Werbung in der er in einer anderen App einen in-App Einkauf tätigt, dann wäre es Sinnvoll zu wissen, dass der Benutzer aufgrund einer bestimmten Kampagne den Event ausgelöst hat. Diese Events muss man unter Attribution als «Conversion» markieren. Man sieht anschliessend, welche Kampagne einen Einfluss darauf hatte, dass dieser Event ausgelöst wurde.


*Abbildung 25: Attribution Zusammenhänge erstellen (Quelle: firebase.blog.com )*[24]

### *5.19.4.1 Postbacks*

Die Werbe-Netzwerke nehmen war, dass auf ihre Werbung geklickt wurde, jedoch haben Sie keine Chance nachzuvollziehen ob Aufgrund der Werbung auch jemand z.B. die App installiert hat. Mit Postbacks kann man die Werbe-Netzwerke benachrichtigen, wenn die Werbung Erfolg hatte. Werbung wird somit effizienter geschaltet und das Unternehmen, welches die Kampagne finanziert kann dadurch viel Geld sparen, indem seine Werbung welche bereits Erfolgreich war, bei dem Benutzer nicht mehr gezeigt wird.

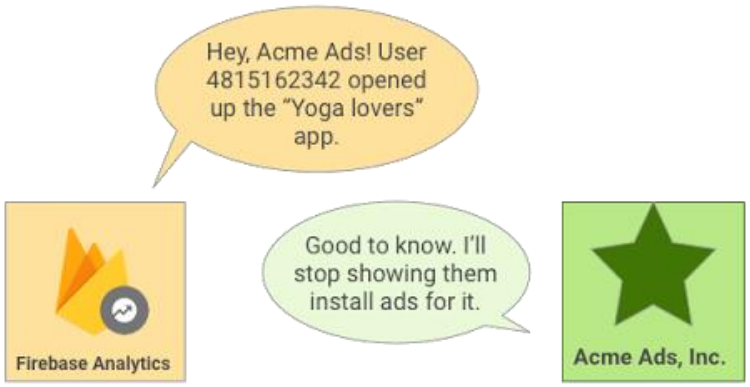



*Abbildung 26: Daten Web-Netzwerk zur Verfügung stellen (Quelle: firebase.blog.com)*[24]

Auch der Erfolg von Firebase Dynamic Links kann somit einfach nachvollzogen werden.

#### *5.19.4.2 Identifikatoren*

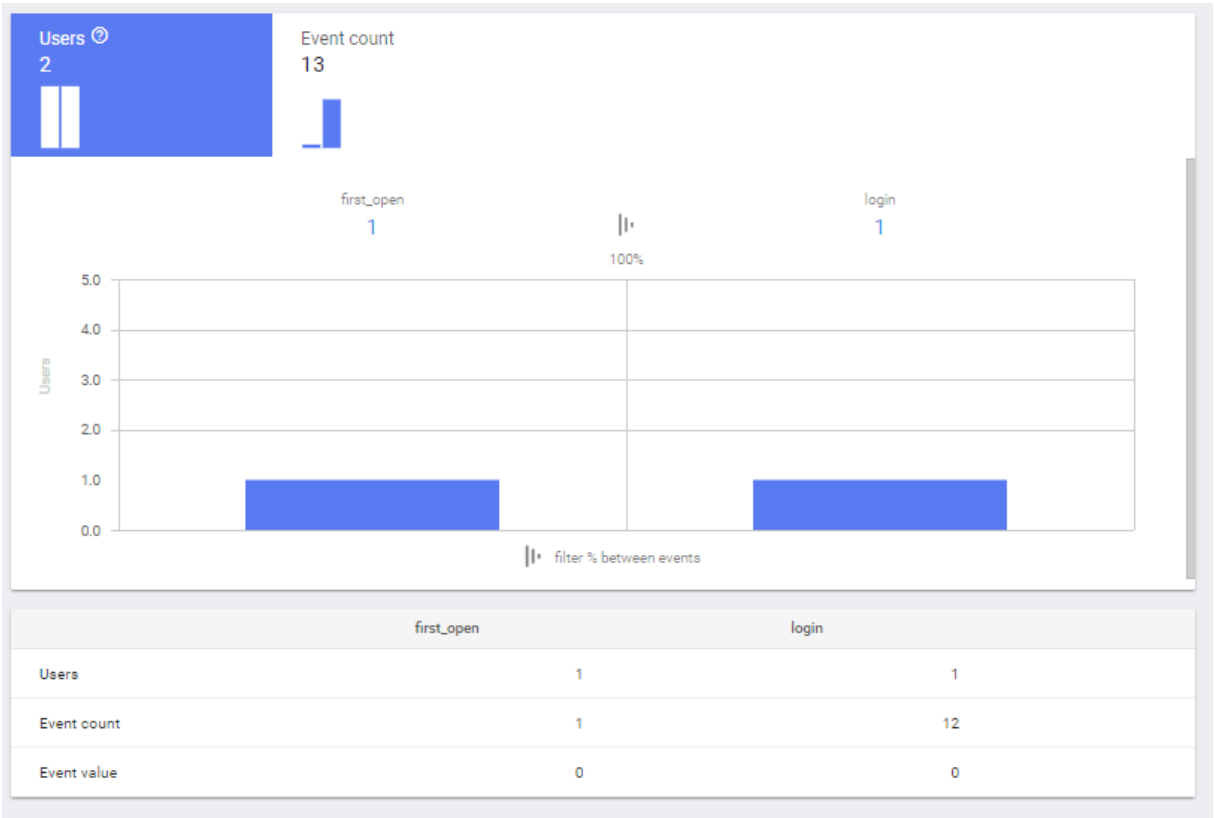
Als Identifikatoren werden sowohl für IOS wie auch für Android eine Advertising ID verwendet. Jedes Gerät hat dafür eine eigene ID, diese kann auf Android Geräten in den Einstellungen nachgeschaut werden.[25]

### 5.19.5 Funnels

Funnels dienen dazu nachzuvollziehen wie Benutzer sich in der App verhalten. Z.B. wäre es gut zu wissen, wie viele Benutzer, welche die App installiert haben sich auch in der App registriert haben.

Wenn viele die App installieren, sich jedoch nur 30% davon registrieren, weisst das auf einen schwierigen Registrations-Prozess hin. Genau dieser Fall ist mit Funnels nachvollziehbar, definierte Events werden in Serie geschaltet und es kann nachvollzogen werden, wie viele mit der Zeit abspringen und welche den nächsten Event auslösen.[26]

Für ein Projekt wie FriendlyChat, soll herausgefunden werden, wie viele der Benutzer sich nach der Installation registriert haben. Dafür definiert man zwei Events «first_open» und «login», beide Events werden miteinander verknüpft. Im Fall unten wurde die App sehr selten installiert, wodurch Daten fehlen. Von denen Benutzern welche die App installiert haben, haben sich jedoch 100% auch registriert.




| | first_open | login |
|---|---|---|
| Users | 1 | 1 |
| Event count | 1 | 12 |
| Event value | 0 | 0 |

*Abbildung 27: Funnel FriendlyChat*

### 5.19.6 StreamView

StreamView erlaubt es in Echtzeit nachzuvollziehen, in welcher Region in den letzten 30 Minuten Benutzer aktiv waren. Das Nachverfolgen von Events nach Region ist möglich und welche Events wievielmal aufgerufen wurden. [27]

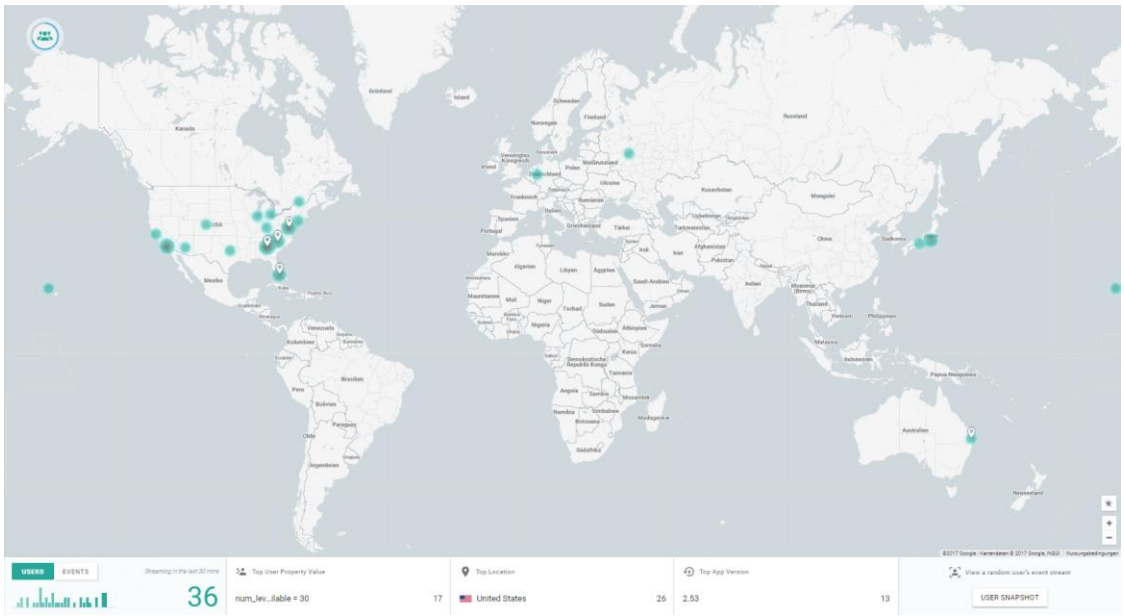



*Abbildung 28: StreamView Google Demo Projekt*

### 5.19.7 DebugView

Die DebugView ermöglich es in Echtzeit nachzuvollziehen welche Events geworfen werden und was für Parameter und User Properties mitgegeben wurden. Dies ist vor allem für die Entwickler gedacht, welche schnell Feedback benötigen um nachzuvollziehen, ob Ihre Implementation funktioniert. Das Gerät muss mit dem Computer Verbunden sein und USB-Debugging unterstützen. Anschliessend muss unter Android eine sogenannte Android Debug Bridge erstellt werden.[28]

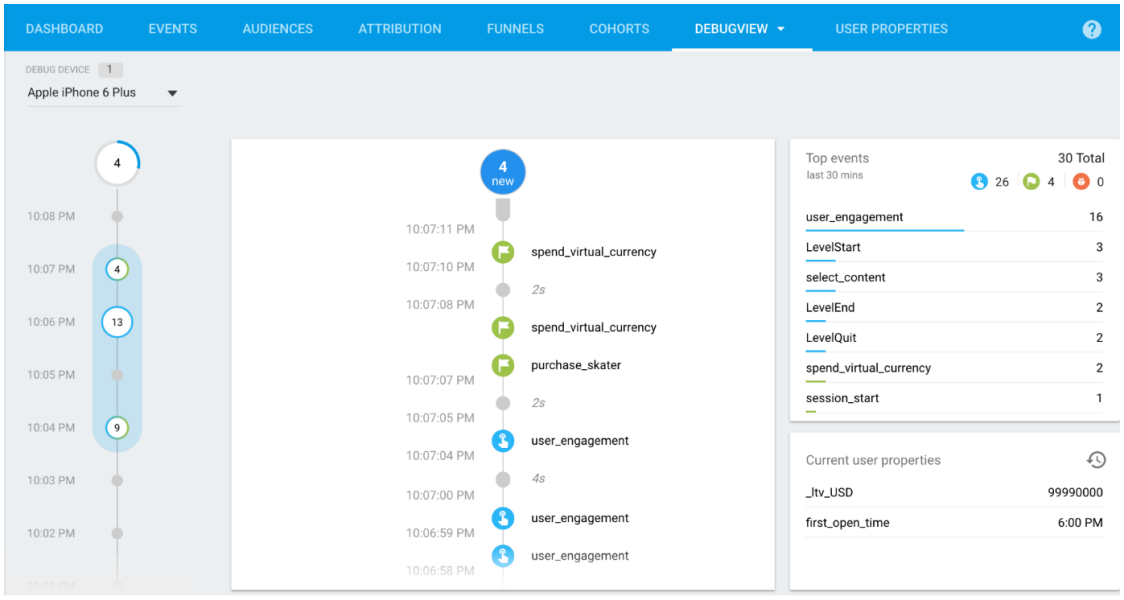



*Abbildung 29: DebugView Übersicht*

### 5.19.8 User Properties

User Properties ermöglicht es Benutzer anhand Ihren Eigenschaften zu kategorisieren. So kann ein User Property «Geschlecht» angegeben werden, welche in der App gesetzt wurde. Die User Properties können manuell im Code gesetzt werden. Firebase selber erhebt ebenfalls ganz automatisch User Properties, wie z.B. die Interessen der Benutzer.[29]

### 5.19.9 Vergleich Firebase Analytics & Google Analytics

Mit Firebase werden verschiedene Tools zusammengefasst und unter einem Hut gebracht, darunter wurde auch Analytics hinzugefügt.

| | Firebase Analytics | Google Analytics |
|---|---|---|
| **Preis** | Gratis, unlimitiert | Gratis, bis Volume erreicht |
| **Messform** | Event basierte Datenerhebung, speziell für Apps | Screenview/Pageview Datenerhebung |
| **Installation** | Key Events werden automatisch implementiert | Entwickler muss explizit Tracking implementieren. |
| **Unterstütze Plattformen** | Android & IOS | Android, IOS & Web |
| **Support** | 10 Email Support Anfragen pro Firebase Benutzer. | Kein persönlicher Support bei Google |
| **Umfang** | Grobe Datenerfassung, wenige Daten werden angezeigt | Umfassende Datenerfassung, umfangreiche Darstellung der Daten. |

*Tabelle 3: Unterschiede Firebase Analytics und Google Analytics*

Firebase Analytics wird für App-Entwickler kostenlos zur Verfügung gestellt, durch automatisch implementierte Events kann ein Überblick über verschiede Daten wie; Aktive Nutzer, App Version, Geräte, Lokation und mehr erhalten. Das Dashboard beschränkt sich auf die Wichtigsten Informationen. Eine umfangreiche Datendarstellung wie mit Google Analytics ist nicht gegeben.

Die Datenerhebung in Firebase Analytics unterscheidet sich grundsätzlich von Google Analytics. Firebase setzt auf eine speziell für Apps entwickelte Event basierte Datenerhebung, Google Analytics jedoch setzt auf eine Screenview/Pageview Datenerhebung. Dadurch ist Firebase Analytics nicht für Web-Plattformen geeignet. Wenn eine Web-Plattform verwendet wird, ist Google Analytics vorzuziehen.

Google hat angekündigt kostenlos Support für Firebase anzubieten, diese beschränkt sich jedoch auf ein FAQ, Release Notes, Foren und Support per Mail (Maximal 10 Anfragen pro Benutzer). Besserer Support ist bei Google Analytics zu erwarten, da meistens ein Reseller Support anbietet.[30]

## 5.20 AdMob & AdWords

AdMob erlaubt es Mobile Entwicklern Ihre App als Werbeplattform zur Verfügung zu stellen. Zusammen mit Google Analytics erhält man grundlegende Informationen zum Umsatz der App.

Mit AdWords kann man Werbung auf den in AdMob definierten Bereichen schalten und Kampagnen lancieren.

# 6 PERFORMANCE BROWSER

Der Performance Browser ist ein Prototyp für Android und IOS. Mithilfe von Firebase sollen Performance Daten in einem Browser erhoben werden. Für die Messungen eignen sich alle Requests, welche von einem Browser gemacht werden. Messbar bei einem einzelnem Request sind Payload und Response Time. Mitgegeben werden ausserdem auch andere Daten, welche Informationen zu Umgebungsmesswerten und Smartphone Daten beinhalten. In einem Prototyp soll gezeigt werden, wie ein Einsatz von Firebase für Performance Messungen angewendet werden kann. Im Performance Browser wird Firebase Performance Monitoring und Firebase Analytics verwendet.

## 6.1 WIE WERDEN WEBSITES GELADEN?

Eine Website besteht normalerweise aus mehreren Komponenten, wie HTML, CSS, Java Script, Bilder und weiteren Document Types. Diese werden nicht alle zusammen mit dem erstem Request geladen, sondern werden alle einzeln geladen. Die initiale Komponente wird als erstes geladen, gefolgt von den weiteren Komponenten.

| Webseite | https://www.cnlab.ch/ |
|---|---|
| HTML | https://www.cnlab.ch/ |
| CSS | https://www.cnlab.ch/sites/default/files/css/css_xE-rWrJf-fncB6ztZfd2huxqgxu4WO-qwma6Xer30m4.css |
| | https://www.cnlab.ch/sites/default/files/css/css_rByfYimIPIeyY2KfJfaS44xO6OyWvk_pVWhCVhrVu64.css |
| | https://www.cnlab.ch/sites/default/files/css/css_9ztyeJ4JnlQly8hNz02bZ0vAlyATlMyUOq_Z5_g2B1g.css |
| | https://www.cnlab.ch/sites/default/files/css/css_wzZifTkRAzYhzQaPy92ICNjmm9Tu8GiiU5m1oWF4CV0.css |
| JS | https://www.cnlab.ch/sites/default/files/js/js_tm1gahPj5RurExr4Zj4GdGvvL7W4-u6_XRVeEsXyp3Q.js |
| | https://www.cnlab.ch/sites/default/files/js/js_U_eJMmIXTb-4_O6i3LnzCBRzm8AS1mdIzXzGAyFxzfw.js |
| | https://www.cnlab.ch/sites/default/files/js/js_2vOiMWT0yKRU5hc9iWlzMa6eD41cU5Bze8WRNve3_n4.js |
| | https://www.cnlab.ch/sites/default/files/js/js_nNvMq5QCEM4ogYqsWM_0iSoAa_gGqoIuJ7MZZI9X4hs.js |
| | https://www.google-analytics.com/analytics.js |
| | https://www.cnlab.ch/piwik/piwik.js |
| JS Events | https://www.cnlab.ch/piwik/piwik.php?action_name=information%20technology%20research%20%7C%20cnlab&idsite=1&rec=1&r=796374&h=14&m=55&s=50&url=https%3A%2F%2Fwww.cnlab.ch%2F&_id=cf9dc5fbd7c20406&_id |

| | |
|---|---|
| | ts=1509803676&_idvc=1&_idn=0&_refts=0&_viewts=1509803676&send_image=1&cookie=1&res=360x640&gt_ms=367&pv_id=pVWayM |
| | https://www.google-analytics.com/r/collect?v=1&_v=j65&a=60892155&t=pageview&_s=1&dl=https%3A%2F%2Fwww.cnlab.ch%2F&ul=de-ch&de=UTF-8&dt=information%20technology%20research%20%7C%20cnlab&sd=32-bit&sr=360x640&vp=360x513&je=0&_u=AACAAEABI~&jid=1527242407&gjid=208493192&cid=1855070110.15098003676&tid=UA-3688268-2&_gid=224981680.1509803676&_r=1&z=883846005 |
| IMG | https://www.cnlab.ch/sites/all/themes/cnlab/graphik/base/active_itr.png |
| | https://www.cnlab.ch/sites/all/themes/cnlab/graphik/base/security.png |
| | https://www.cnlab.ch/sites/all/themes/cnlab/graphik/base/software.png |
| | https://www.cnlab.ch/sites/all/themes/cnlab/graphik/base/performance.png |
| | https://www.cnlab.ch/sites/all/themes/cnlab/graphik/base/bg_itr.png |
| | https://www.cnlab.ch/sites/all/themes/cnlab/graphik/base/gradient_wide52.png |
| | https://www.cnlab.ch/sites/default/files/favicon.ico |

*Tabelle 4: Ladevorgang Webseite*

## 6.2 PROTOTYP

Für den Prototyp besteht eine Android sowie auch eine IOS App, welche beide Firebase Performance Monitoring und Firebase Analytics verwenden um diverse Daten zu erheben.

Beide Browser Varianten wurden mit einer WebView implementiert, welche in beiden Apps zum Einsatz kommen. Unter einer WebView kann man sich einen In-App View vorstellen, welche Webseiten darstellen kann.

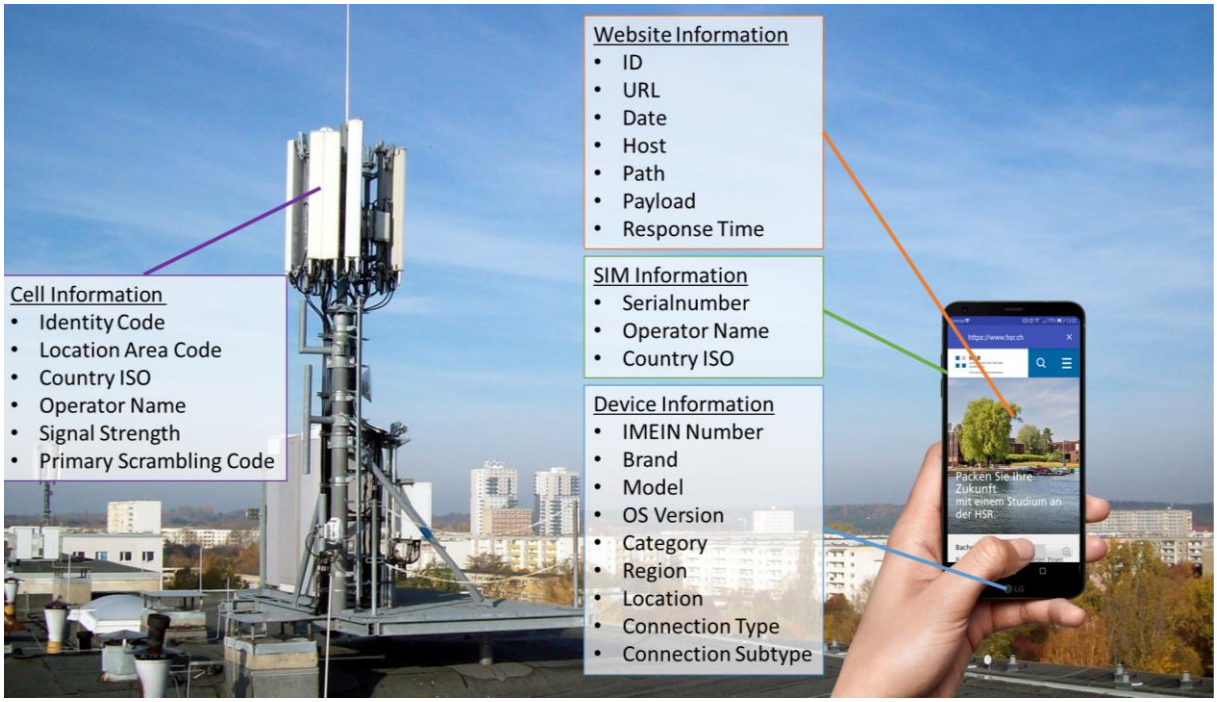



*Abbildung 30: Big Picture Protoyp*

## 6.3 WEBVIEW

### 6.3.1 Android

Die Android WebView beinhaltet eine Klasse WebViewClient. Während einem Ladeprozess einer Webseite, werden verschiedene Events ausgelöst, auf die man Einfluss nehmen kann. So kann man auf den Ladeprozess einer Webseite Einfluss nehmen.

Der OnPageStarted Event, wird jeweils aufgerufen, wenn die Website beginnt zu laden. Bei IFrames oder andere Frames die im Main Frame eingefügt sind, wird der Event nicht nochmal aufgerufen.

Der OnPageFinished Event wird nur aufgerufen, wenn der Main Frame geladen wurde. Jedoch kann dieser Event bereits ausgelöst worden sein, wenn Inhalte wie Bilder oder Videos noch mit JavaScript geladen werden.

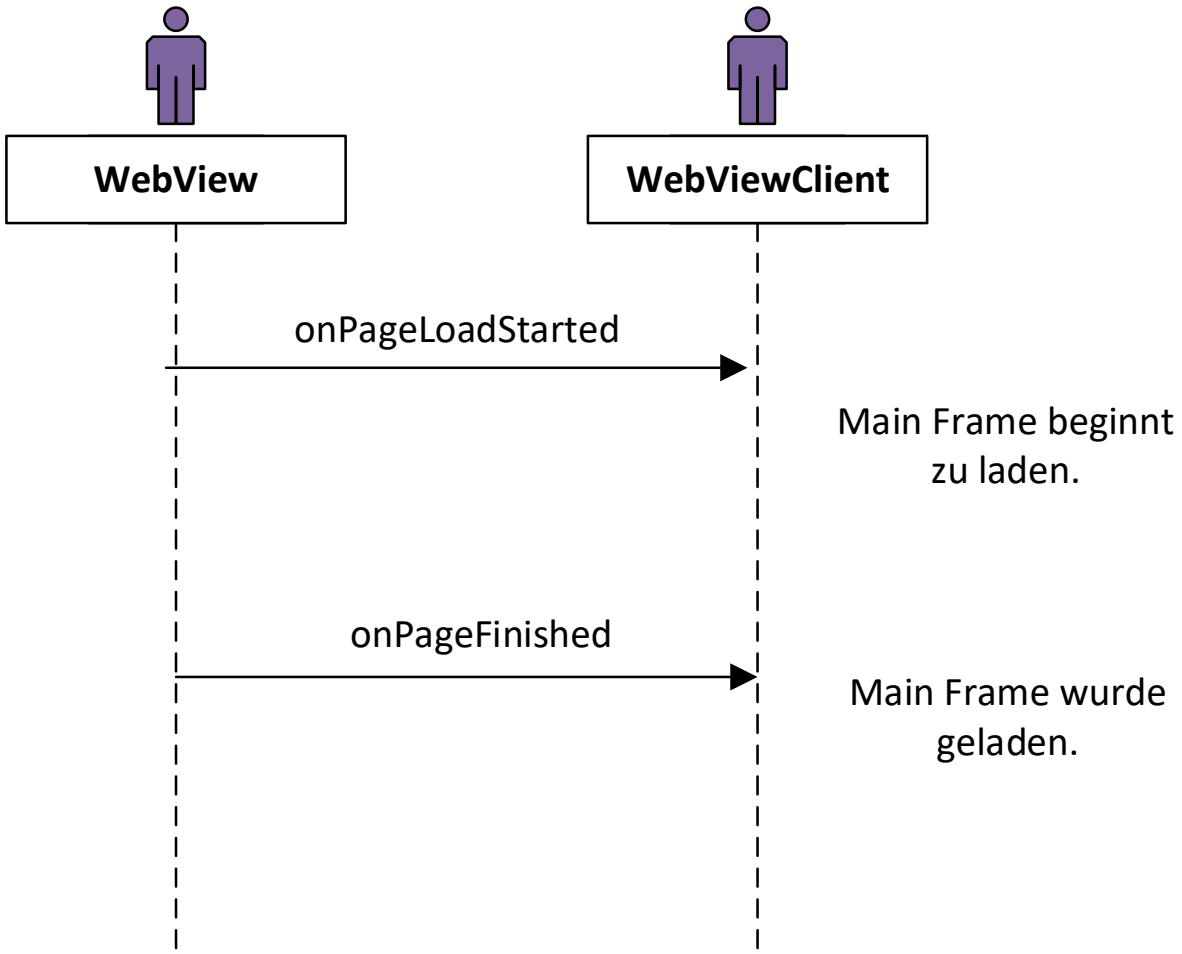



*Abbildung 31: WebView Android*

ShouldInterceptRequest ist die Methode, welche es erlaubt jeden einzelnen Request von der WebView ausgehend abzufangen. Man hat an dieser Stelle die Möglichkeit Anpassungen am Request vorzunehmen. Für den Prototyp wurde ein Request mit OKHTTP neu initiiert und die Response der WebView zurückgegeben, dadurch ist Firebase Performance in der Lage die Requests automatisch zu messen.

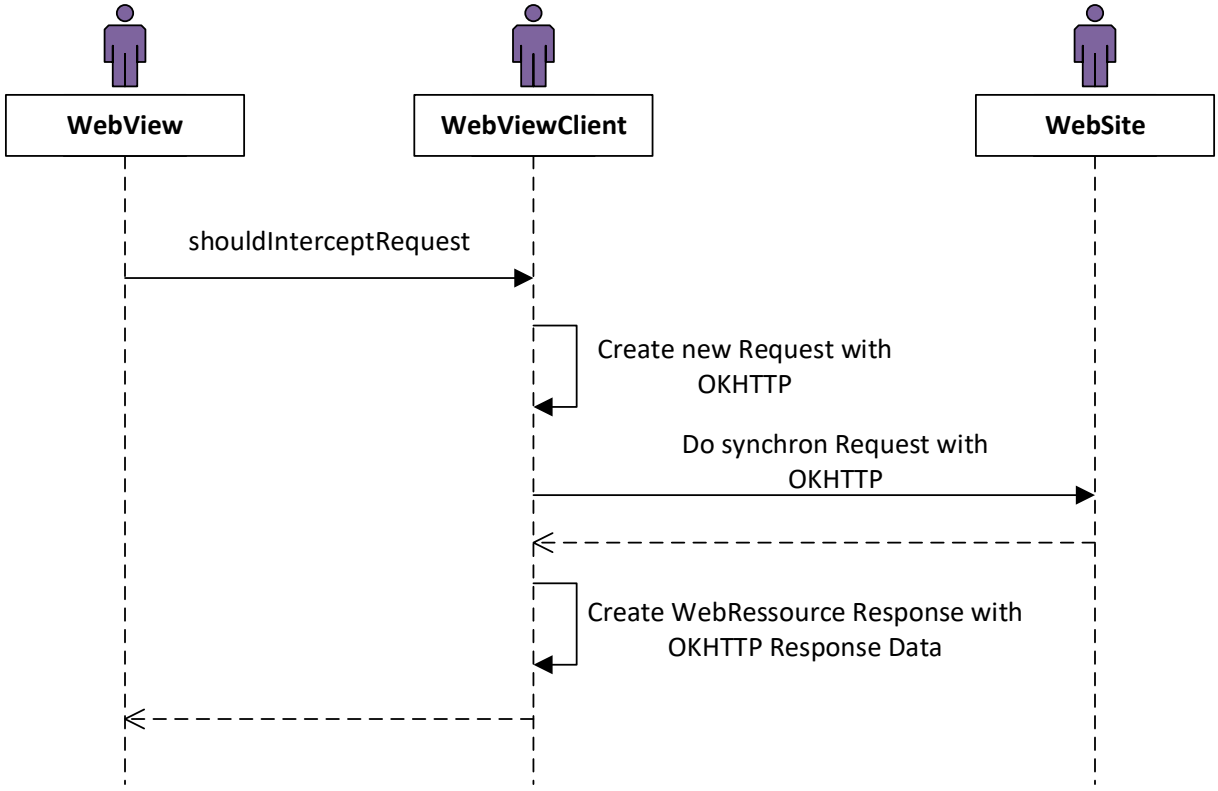



*Abbildung 32: WebView Android Intercept*

### 6.3.2 IOS WebView

Die IOS WKWebView ist die WebView, welche für IOS Apps benutzt wird. Sie bietet einige ähnliche Funktionen wie die Android WebView. Es gibt keine analoge Methode zu ShouldInterceptRequest. Ein Unterschied zwischen IOS und Android ist, dass die Methoden als Delegate definiert werden und diese in der WebView registriert werden.

Es eignen sich zwei Events um eine Messung zu machen. Ein WebView Event welcher zu Beginn des «Loading» Prozesses ausgeführt wird und ein Event, der am Schluss ausgelöst wird.

Das webView(_:didCommit:) Delegate wird ausgeführt, wenn die Website beginnt Web Content zu Empfangen.

Das webView(_:didFinish:) Delegate wird ausgeführt, wenn die Navigation komplett ist.

Die Funktionen werden in der Apple Reference nicht genauer beschrieben. Es ist nicht ersichtlich, wann diese Events genau aufgerufen werden.[31]

## 6.4 PERFORMANCE TRACES

### 6.4.1 Android

Traces werden im Android Prototyp bei den einzelnen Request und über den ganzen «Loading Process» gemacht. Wenn die Webseite beginnt zu laden, wird ein Trace gestartet, welcher die Zeit misst, bis die Website komplett geladen wurde.

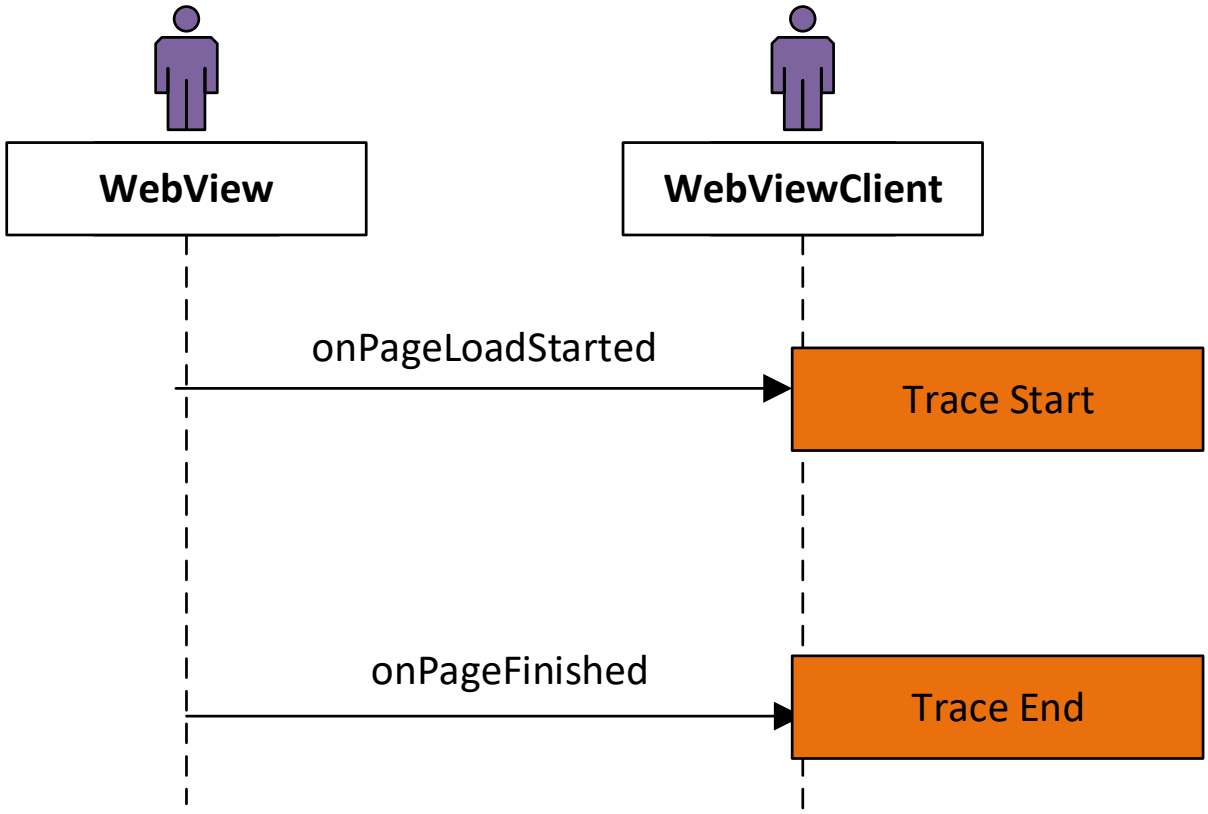



*Abbildung 33: Trace Android Main Request*

Vor den OKHTTP Request wird ein Trace gestartet. Sobald die Response erhalten wurde, dann wird der Trace gestoppt. Als Trace Name wird jeweils die URL mitgegeben, welche aufgerufen wurde. Zusätzlich macht Firebase die eigene Aufzeichnung der Netzwerkanfragen, wenn OKHHTP benutzt wird.

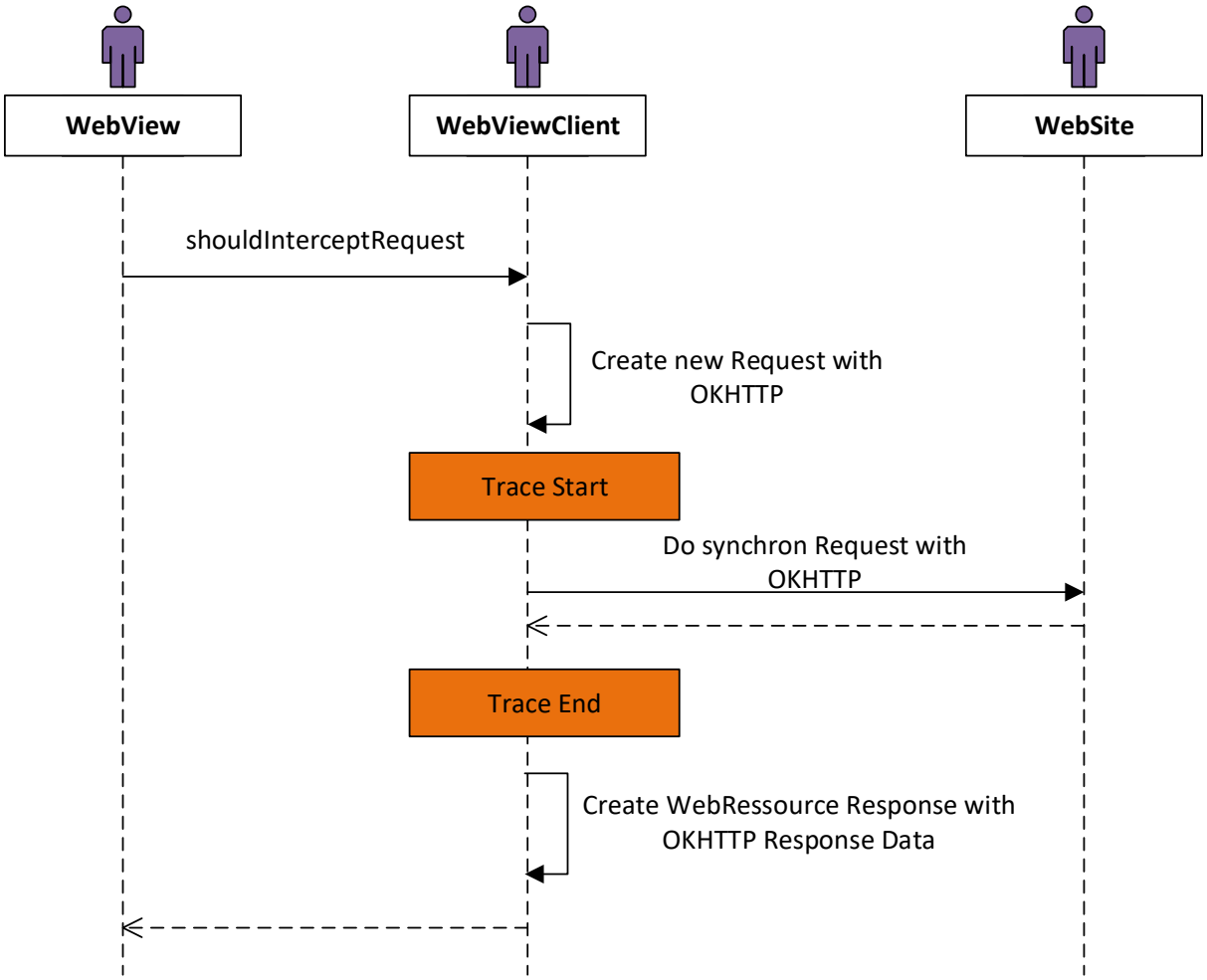



*Abbildung 34: Trace Android Component Requests*

### 6.4.2 IOS

In der IOS Version werden Traces von Firebase an denselben Stellen eingesetzt. Wenn die WebView beginnt zu laden und die Website komplett navigiert hat.

Bei den Traces wird die Zeit zwischen den beiden Events gemessen. Als Trace Name dient die aufgerufene URL. Diese wird dann als Trace zu Firebase Performance gesendet.

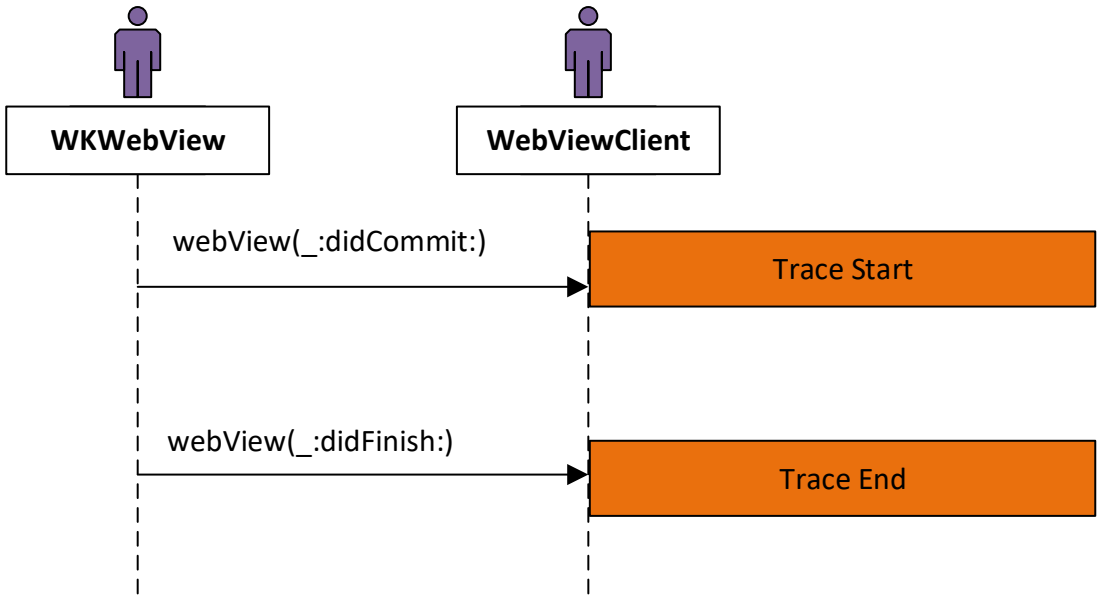



*Abbildung 35: Trace IOS Main Request*

## 6.5 Firebase Analytics Events

### 6.5.1 Android

Während dem Ladeprozess einer Website werden mehrere Events ausgelöst, welche mit Firebase Analytics erfasst werden. Bei jeder Response wird ein einen Analytics Event mit dem Name «ResponseEvent» ausgelöst. Dieser enthält Zeit, Payload, Mobilfunkdaten und Smartphone Daten.

Wenn bei einem Request die Webview der Event «onPageStarted» ausgelöst wird, dann wird eine «NetworkRequestID» gesetzt. Diese wird bei allen folgenden angefragten Ressourcen beibehalten. Dadurch wird die initiale Anfrage mit den folgenden Requests verknüpft, wodurch festgestellt werden kann, welche Request dazugehören.

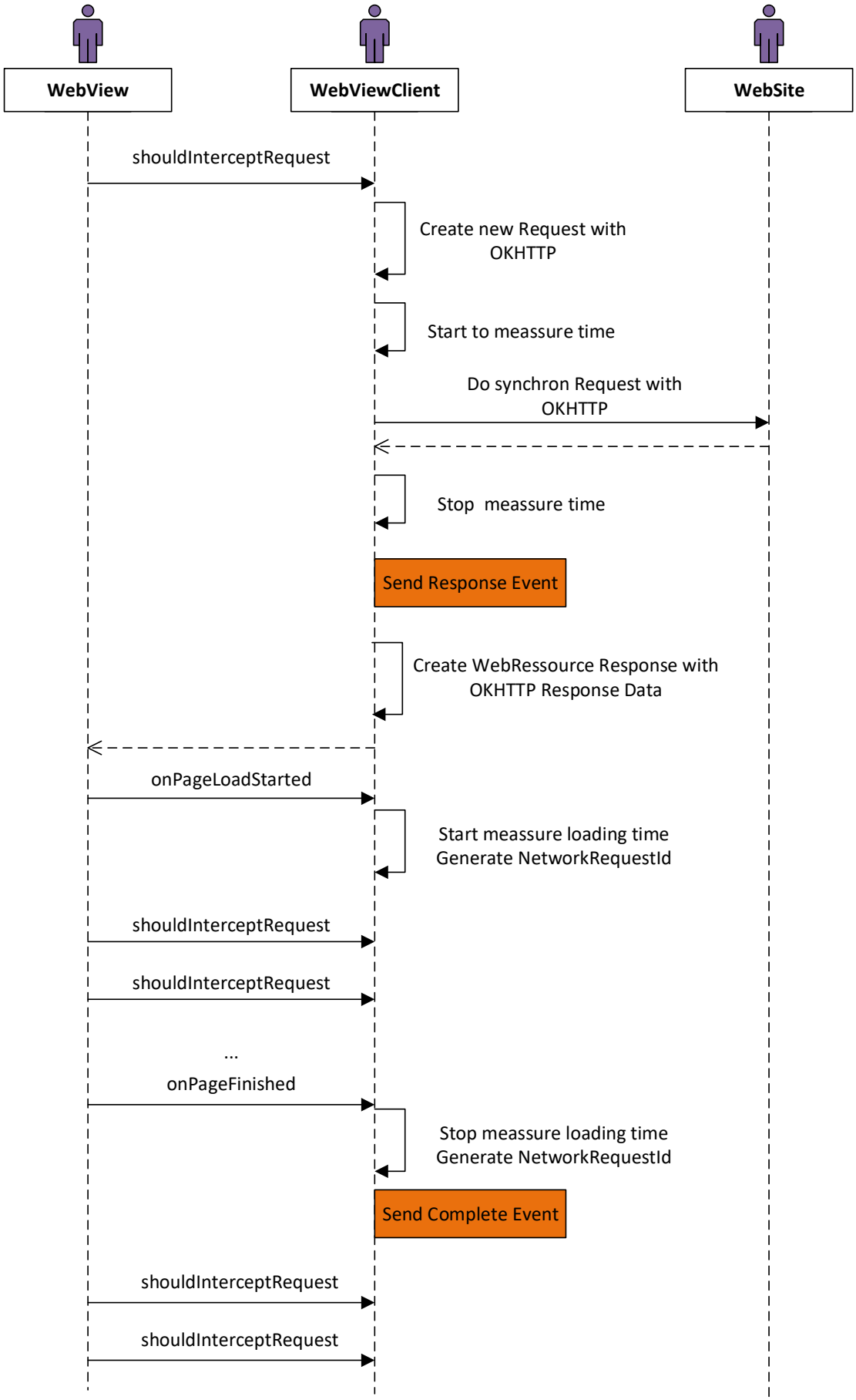



*Abbildung 36: Events Android*

Der komplette Ablauf eines Webseitenaufrufs wird in der Grafik Angezeigt. Zu beachten ist jedoch, dass onShouldInterceptRequest auf einem anderen Thread läuft. OnPageStarted und OnPageFinished laufen auf dem Main Thread. OnShouldInterceptRequest wird jeweils zuerst ausgeführt und wartet auf Response. In dieser Zeit wird die OnPageStarted Methode ausgeführt, welche eine NetworkRequestID erstellt. Wenn die Response da ist, wird ein Response Event für Analytics ausgelöst.

Jeder Analytics Events hat statische Daten, welche immer mitgeschickt werden. Die statischen Daten enthalten Umgebungsmesswerte und Smartphone Daten. Bei OnPageStarted werden diese neu generiert. Die folgenden Events haben immer dieselben statischen Daten.

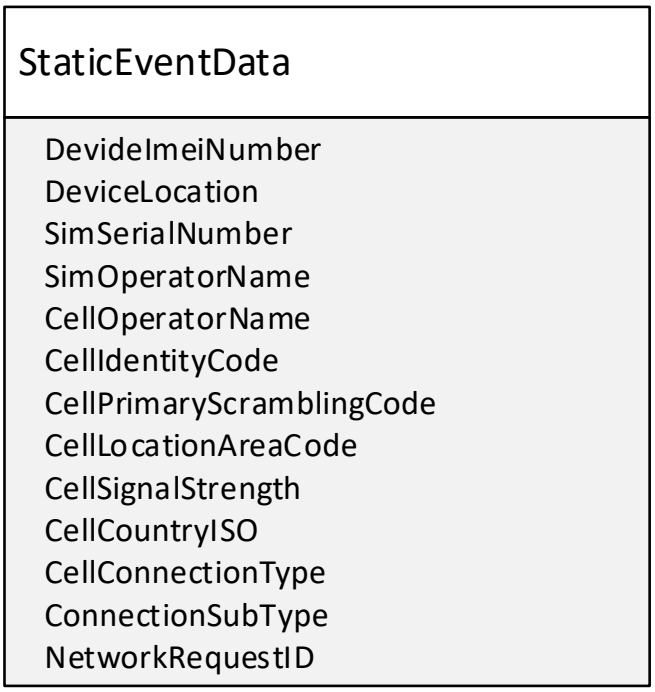



*Abbildung 37: Static Event Data Android*

Die dynamischen Werte werden jeweils Event spezifisch hinzugefügt.

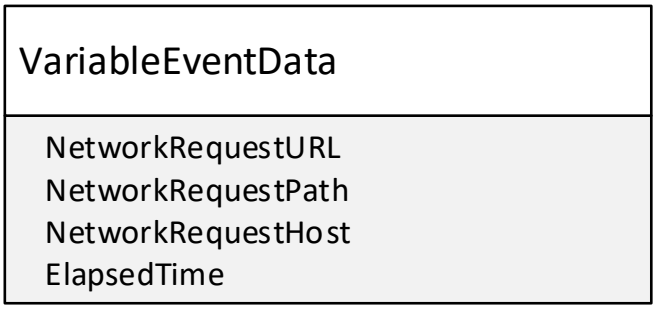



*Abbildung 38: Variable Event Data Android*

### 6.5.2 IOS

Es wird die Zeit zwischen den beiden Events gemessen. Die gemessene Zeit wird dann als Parameter einem Firebase Analytics Event mitgegeben.

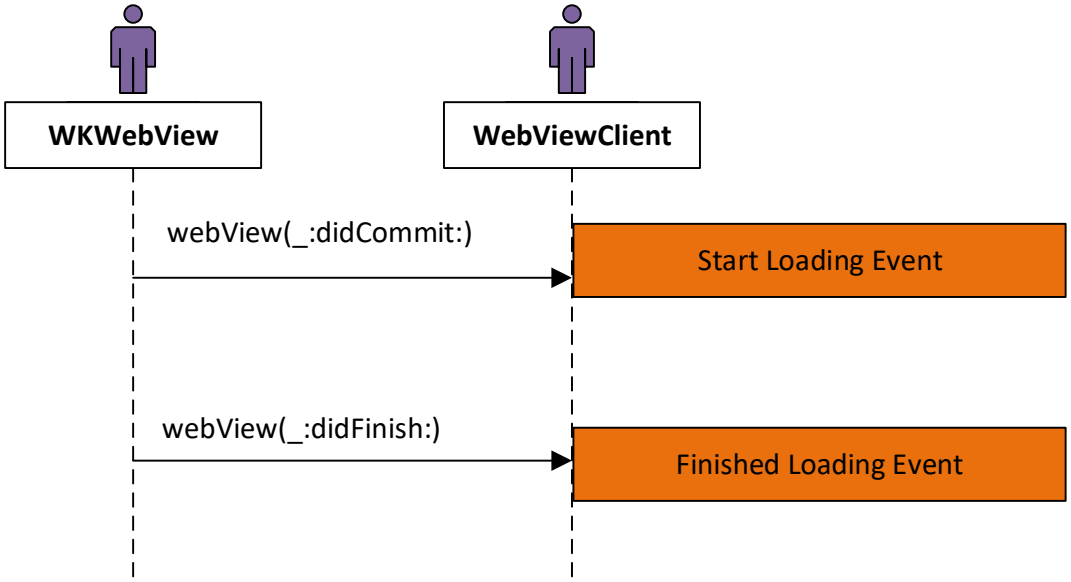



*Abbildung 39: Event IOS*

Als Parameter werden statische Daten generiert, welche bei verschiedenen Events mitgeschickt werden können. Spezifisch für einen Event können weitere Daten, wie die vergangene Zeit benutzt werden. Diese werden analog zu Android mit den VariableEventData angefügt.

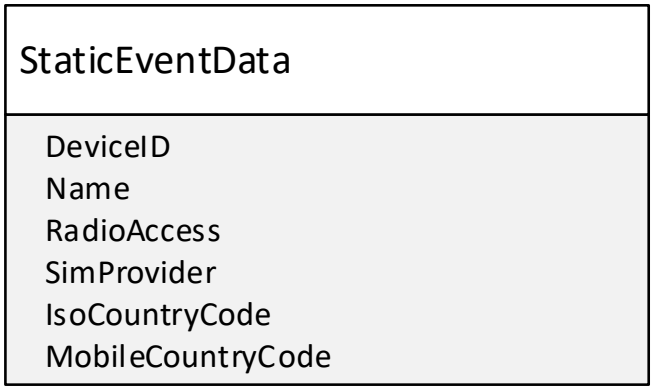



*Abbildung 40: Static Event Data IOS*

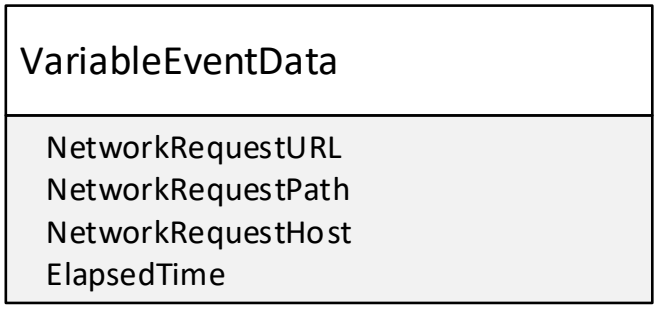



*Abbildung 41: Variable Event Data IOS*

## 6.6 PROBLEMSTELLUNGEN

Bei Android läuft ShouldInterceptRequest auf einem anderen Thread, als die anderen WebView Events, welche auf dem Main Thread laufen. Dadurch können Race Conditions und Data Races auftreten, welche im Stand des Prototyps zu fehlerhaften Messungen führen können. Beispielsweise kann der erste «ReponseEvent» noch eine alte Request ID enthalten. Dieses Zusammenspiel müsste synchronisiert werden, um korrekte Datensammlungen zu erhalten.

Bei IOS bietet die IOS WKWebView leider keine analoge Funktion zur ShouldInterceptRequest Methode und es ist über diesen Weg nicht möglich die einzelnen Request abzufangen.

# 7 DATENAUSWERTUNG MIT BIGQUERY

## 7.1 WAS IST BIGQUERY

BigQuery ist das Data Warehouse von Google, welche es erlaubt Abfragen auf Datensätze über die Google Infrastruktur zu machen. Durch die Nutzung der Google Infrastruktur steht deren bereitgestellten Rechenkapazitäten zur Verfügung.[32] Die Verwendung ist bis zum ersten Terabyte verarbeiteten Daten gratis und anschliessend kostet es 5 Dollar pro Terrabyte.[33] Firebase Analytics kann mit BigQuery verknüpft werden und die gesammelten Event Daten können dann ausgewertet werden. BigQuery bietet mehrere Zugangsvariationen; via Java, C++, Rest API, WEB Interface oder über Drittanbieter.

## 7.2 STANDARD SQL UND LEGACY SQL

BigQuery erlaubt es mit zwei verschieden SQL Dialekten Abfragen zu machen. Standard SQL wurde nach dem SQL 2011-Standard implementiert und ist seit BigQuery 2.0 verfügbar. [34] [35] Legacy SQL ist die alte Abfrage Syntax vom BigQuery. Im Internet finden sich deshalb viele Beispiele in beiden Dialekten, es muss deshalb aufgepasst werden, dass der korrekte Dialekt ausgewählt wird.

## 7.3 Firebase Analytics Events

Die mit dem Webbrowser gesammelten Daten werden alle mit diversen Events an Firebase Analytics weitergeschickt. Nach der Verknüpfung von Firebase Analytics und BigQuery können die Rohdaten anschliessend ausgewertet werden. Es können auf alle Event Parameter und die selbst gesetzten User Properties zugegriffen werden, welche mit dem Event mitgeschickt wurden.

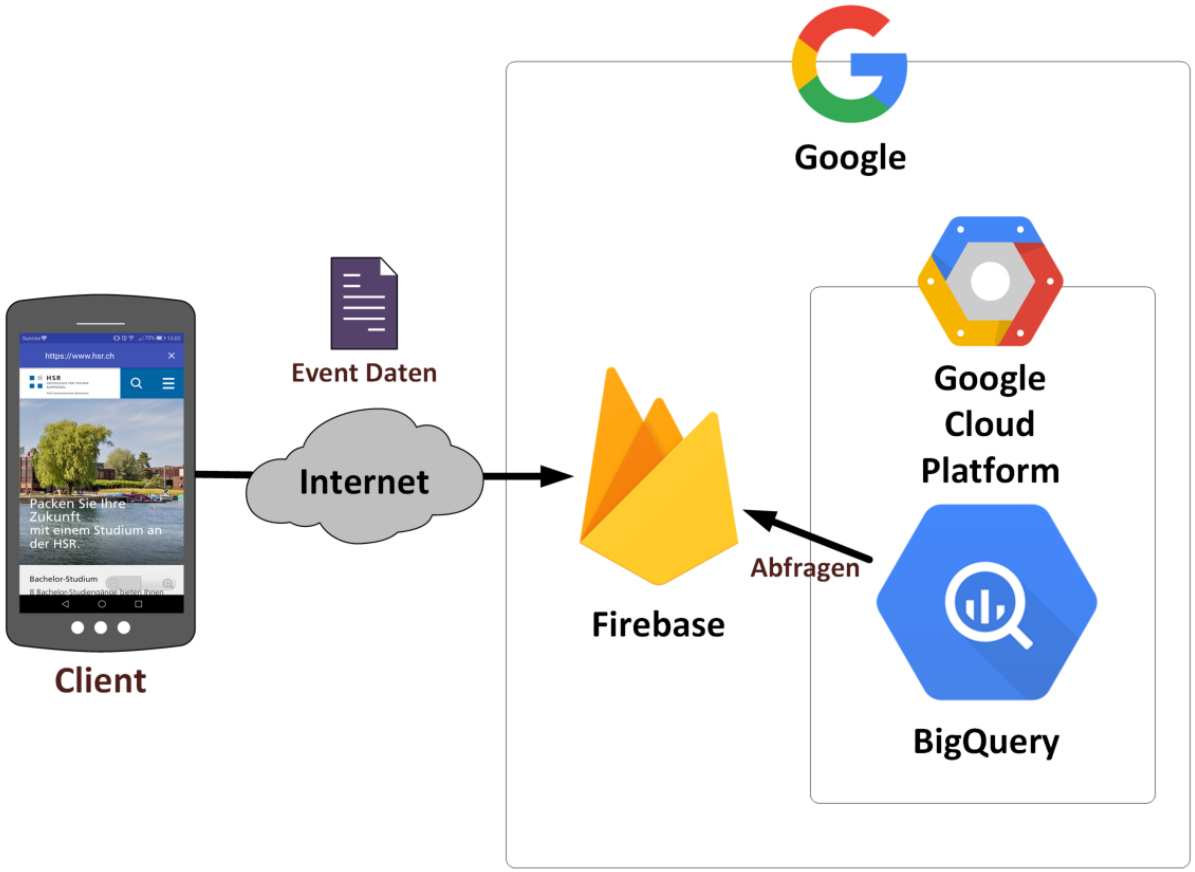



*Abbildung 42: Firebase Performance Browser Systemübersicht*

Alle gesammelten Events können über einen bestimmten Zeitraum in einer ganzen Tabelle angezeigt werden. Jedes Event ist in einer eigenen Zeile abgespeichert, dadurch entsteht ein sehr geschaltetes Format.

Jeder Event wird in die 2 Dimensionen «user_dim» und «event_dim» unterteilt und in folgendem Format gespeichert. Ein «event_dim» kann aber mehrmals vorkommen, da er vom gleichen User kommen kann.

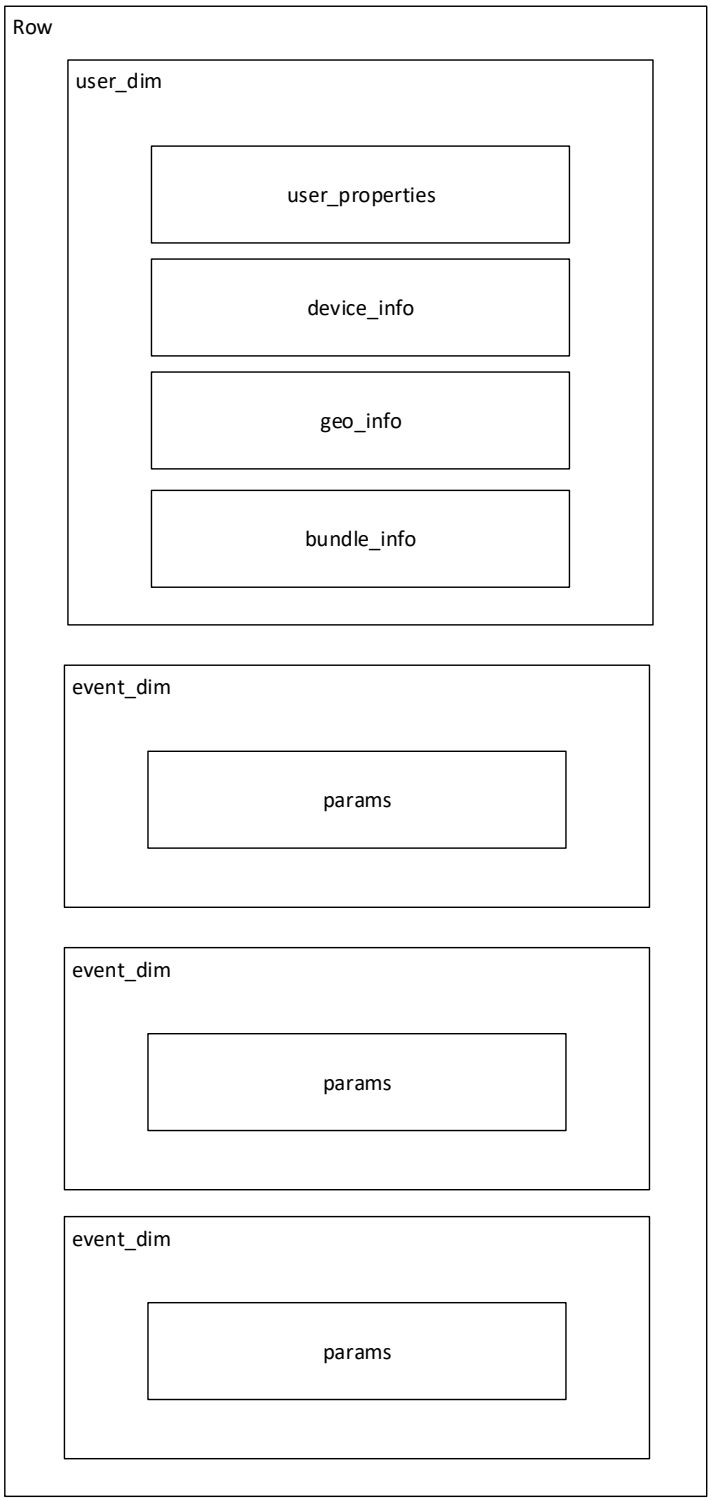



*Abbildung 43: Event Datenreihe*

Der User_Dim besteht weiterhin aus Unterkategorien, in welche die User Properties, Device Infos, Geo Infos und Informationen über das Bundle gespeichert sind. Im Event_Dim befinden sich alle Eventparameter, welche beim Webbrowser gesetzt wurden. Durch die Speicherung dieser Art, entsteht eine sehr flache Hierarchie.

In der Weboberfläche von BigQuery gibt es für die Rohdaten eine Preview. Die Preview ist eine Vorschau der Daten und es ist zu sehen, wie die Daten geschachtelt gespeichert sind. In der folgenden Illustration wird eine Datenreihe gezeigt, welche so angelegt wurde. Für die Vorstellung wurde die ganze Datenreihe gezeigt und hinaus gezoomt. Durch die Ansicht der ganzen Reihe sind die Daten nicht mehr erkennbar, aber im Bild sind sie markiert.


*Abbildung 44: BigQuery Datenreihe*

## 7.4 Auswertung mit BigQuery

### 7.4.1 Datenmodellierung

Wenn keine Änderungen am Datensatz gemacht worden sind, können SQL Abfragen sehr gross werden. BigQuery ermöglicht es Abfrage Ergebnisse als neue Tabelle zu speichern. Es ist so möglich Tabellen zu generieren, welche den eigenen Bedürfnissen entsprechen. Auf diese Tabelle können dann kleine SQL Abfragen gemacht werden. Um an alle Response Events zu gelangen lohnt es sich eine neue Tabelle zu generieren um Auswertungen zu machen.

Die folgende Abfrage generiert eine Tabelle mit Geräten welche auf eine URL einen Request gemacht haben. Für Tabellen mit mehreren Spalten, können zusätzliche Event Parameter hinzugefügt werden.

**Im "New Query" Abschnitt:**

```
SELECT
  (SELECT params.value.string_value FROM x.params WHERE params.key =
'NetWorkRequestURL') AS NetworkRequestURL,
  user_dim.device_info.mobile_marketing_name AS MarketingName,
  user_dim.device_info.mobile_brand_name AS Marke
  FROM `android_hsr_ch_webbrowser_ANDROID.app_events_*`, UNNEST(event_dim) AS
x
WHERE x.name  = 'ResponseEvent' AND _TABLE_SUFFIX BETWEEN '20171209' AND
'20171212'
```

*Code 7: Beispiel Tabelle generieren*

Die Abfrage resultiert in folgendem Ergebnis. Es wird ein Ausschnitt aus dem Resultat gezeigt.

| NetworkRequestURL | MarketingName | Marke |
|---|---|---|
| https://adservice.google.ch/adsid/google/ui | Galaxy S8 | Samsung |
| https://adservice.google.ch/adsid/google/ui | Galaxy S8 | Samsung |
| https://adservice.google.ch/adsid/google/ui | Galaxy S7 Edge | Samsung |
| https://adservice.google.ch/adsid/google/ui | P10 | Huawei |
| https://adservice.google.ch/adsid/google/ui | P10 | Huawei |
| https://adservice.google.ch/adsid/google/ui | P10 | Huawei |
| https://adservice.google.ch/adsid/google/ui | P10 | Huawei |
| https://adservice.google.ch/adsid/google/ui | P10 | Huawei |
| https://adservice.google.de/adsid/google/ui | P10 | Huawei |

*Tabelle 5: Resultat Datenmodellierung*

### 7.4.2 Auswertungen

Mit den gesammelten Daten lassen sich verschiedene Auswertungen generieren. In den folgenden Unterkapiteln werden verschiedene Auswertungen über den Zeitraum vom 20.11.2017 – 14.12.2017 gemacht. Die Auswertungen zeigen einen kleinen Einblick, was alles für Auswertungen möglich sind. Für die Auswertung wurde eine neue Tabelle erstellt und anhand dieser wurden die Auswertungen gemacht. Die Tabelle enthält alle Response Events. Die Spalten sind in folgender Struktur gegliedert:

| Kolonnen Name | Beispiel |
|---|---|
| CellIdentityCode | 20690952 |
| CellLocationAreaCode | 48041 |
| CellCountryISO | de |
| CellSignalStrength | -113 |
| CellPrimaryScramblingCode | -1 |
| CellOperatorName | Vodafone.de |
| DeviceImeinNumber | … |
| SimSerialNumber | … |
| SimOperatorName | Swisscom |
| SimCountryISO | ch |

| | |
|---|---|
| DeviceLocation | La:49.453446575 Lo:11.07688866 |
| DeviceConnectionType | WIFI |
| DeviceConnectionSubType | |
| NetworkRequestID | 97LTXYlqBWHSK7PXsAyfy |
| NetworkRequestDate | Sat Nov 25 18:01:40 GMT+01:00 2017 |
| NetworkRequestHost | www.google.de |
| NetworkRequestURL | https://www.google.de/images/icons/material/system/1x/email_grey600_24dp.png |
| NetworkRequestPathURL | /images/icons/material/system/1x/email_grey600_24dp.png |
| NetworkResponsePayload | 219 |
| NetworkResponseTime | 258 |
| MobileBrand | Huawei |
| MobileModel | VTR-L09 |
| City | Nuremberg |
| OSVersion | 7.0 |
| DeviceCategory | mobile |
| ModelName | VTR-L09 |
| MarketingName | P10 |
| Region | Bavaria |

*Tabelle 6: Auflistung Kolonnen*

### 7.4.3 Auflistung Smartphone

Von welchem Smartphone Modell wird die App am meisten benutzt. Besonders Firmen, mit einer Marketing Abteilung können so herausfinden, welches das beliebteste Smartphone der Benutzer ist. Dadurch können zum Beispiel Gewinnspiele gemacht werden, mit dem beliebtesten Smartphone als Preis.

| Anzahl | Marke | MarketingName |
|---|---|---|
| 6 | Google | Nexus 5 |
| 4 | Motorola | Moto G4 |
| 3 | Samsung | Galaxy S7 Edge |
| 3 | Samsung | Galaxy S7 |
| 2 | Huawei | P10 |
| 2 | Samsung | Galaxy S8 |
| 1 | Samsung | Galaxy A5 2016 Duos |
| 1 | HTC | One A9s |
| 1 | Motorola | Moto G4 Play |
| 1 | Samsung | Galaxy J7 (2016) |
| 1 | Sony | Xperia X |
| 1 | Google | Compute Engine |
| 1 | Samsung | Galaxy S5 |
| 1 | HTC | 10 |
| 1 | Google | Nexus 6P |
| 1 | Huawei | Mate 9 |
| 1 | Huawei | P8 |
| 1 | LG | G4 |
| 1 | Huawei | Honor 8 |

*Tabelle 7: Anzahl Smartphones (inkl. Testgeräte)*

### 7.4.4 Android Version Verteilung

Von Android gibt es immer wieder neue Versionen mit neuen Features. Es kann jedoch sein, dass neue Features nicht Rückwärtskompatibel sind. Für Anbieter ist es Wichtig zu wissen bis auf welche OS Versionen er seine Dienstleistungen anpassen muss. Mit dieser Information hat der Dienstleister die Möglichkeit sich an seine Benutzer anzupassen.

| ANZ | OSVersion |
|---|---|
| 18 | 6.0.1 |
| 11 | 7.0 |
| 4 | 6.0 |
| 1 | 8.0.0 |

*Tabelle 8: OS Version Verteilung(inkl. Testgeräte)*

### 7.4.5 Host Zugriffe

Ein Browseranbieter kann es sehr interessieren, auf welche Hosts seine Benutzer am meisten Zugreifen. Da die meisten Zugriffe gezeigt werden, entsteht für den Anbieter die Möglichkeit Optimierungen für die meistbesuchten Webseiten vorzunehmen. In der folgenden Auswertung sind 10 meisten Zugriffe aufgezählt, welche auf einen Host gingen.

| Anzahl | Host |
|---|---|
| 13818 | www.google.ch |
| 12668 | www.res2ep.scsstatic.ch |
| 7061 | www.google.com |
| 5189 | www.gstatic.com |
| 2789 | encrypted-tbn0.gstatic.com |
| 2696 | f.blick.ch |
| 2615 | www.20min.ch |
| 2463 | www.res1ep.scsstatic.ch |
| 2327 | www.bluewin.ch |
| 2205 | tr1.admeira.ch |

*Tabelle 9: Hostzugriffe (inkl. Testgeräte)*

### 7.4.6 Durchschnittszeiten Verbindungsarten

Benutzer kann es sehr Interessieren, ob sie mit WLAN oder Mobilfunk schneller eine Webseite laden können. In dieser Auswertung wird die durchschnittliche Zugriffszeit auf den Host www.google.ch pro Verbindungsart ausgewertet.

| AVGResponseTime | DeviceConnectionSubType | DeviceConnectionType |
|---|---|---|
| 115 | HSPA+ | MOBILE |
| 88 | LTE | MOBILE |
| 67 | | WIFI |

*Tabelle 10: Durchschnittszeiten pro Verbindungsart*

## 7.5 MÖGLICHKEITEN

Mit BigQuery gibt es Möglichkeiten Events auszuwerten. Alle Events werden an einer Stelle gesammelt und können mit BigQuery über die Google Infrastruktur ausgewertet werden. Durch die Sammlung an Benutzer Daten, wie die User Properties, Device Info, Geo Info und Bundle Info, kann über die Benutzer sehr viel herausgefunden werden.

Mit User Properties lassen sich Benutzer kategorisieren. Daraus kann man ableiten, welche Benutzergruppen am grössten sind. Die User Properties müssen jedoch selber erstellt werden, um Benutzer zu kategorisieren. Google sammelt zwar die Interessen der einzelnen Benutzer, diese sind aber über BigQuery nicht einsehbar. Für Manager bieten diese Informationen eine grosse Möglichkeit sich weiter an seinen Kunden anzupassen, oder gezielt seine Kundengruppen zu vergrössern.

Über die Device Info werden viele Gerätedaten gesammelt, welche für die Weiterentwicklungen der App sehr wertvoll sind. Bei Entwicklung einer App spielt es oft eine Rolle, welches das Minimal OS eines Gerätes sein muss. Mit diesen Informationen kann man mit der Zeit sehr genau abschätzen, welche Minimalanforderungen eine App unterstützen muss.

Mit Geo Info lassen sich Ortsinformationen herausfinden. Mit diesen Informationen kann man einsehen, wo eine App am meisten benutzt wird. Events lassen sich nach Region gruppieren, daraus kann man feststellen, welche Region am Meisten Events generiert.

Im Bundle Info stehen Informationen über die Installation. Es steht wie die App Installiert wurde. Die App wurde entweder über den App Store installiert, oder direkt mit der APK. Bundle Info beherbergt auch die Information, von welcher App Version sie stammt. Daraus kann identifiziert werden, welche Version die Benutzer am meisten nutzen. Mit einer Benachrichtigung können Nutzer gezielt informiert werden, dass eine neue App Version zur Verfügung steht.

In den Events sind die eigens generierten Events mit ihren Parametern. Dies sind die Parameter, welche in einer Applikation zu Firebase Analytics geschickt wurden. Auf jeden dieser Parameter kann zugegriffen werden und Auswertungen können daraus Erfolgen. Im Fall dieser Studienarbeit kann auf die gemessenen Zeiten und Umfeld Parameter zugegriffen werden. Die Umfeld Parameter können Empfangsstärke, Genauer Standort, URL oder andere

Angaben sein. Es ist auch möglich die Messwerte eines Speedtests, wie der von cnlab[36] zu speichern und anschliessend auszuwerten. Es gilt zu beachten, dass die Zeichen auf 100 limitiert sind.

Bei einer Auswertung muss man sich Bewusst sein, dass je nach Anwendungsfall Duplikate vermieden werden müssen. Bei einer Installation wird immer ein Event generiert, obwohl der Event immer vom gleichem Benutzer ausgelöst worden sein kann. Deshalb ist es wichtig, dass Eventdaten immer einem Benutzer eindeutig zugewiesen werden können. Im Fall dieser Studienarbeit war das die IMEINNummer oder die SIMSerialNumber. Für diese Auswertung muss jedoch die Berechtigung vom Benutzer erteilt worden sein. Auf ein Attribut, welches immer Zugegriffen werden kann ist die Advertising ID. Diese kann jedoch von einem Benutzer immer wieder neu generiert werden.

# 8 SCHLUSSFOLGERUNGEN

Firebase bietet eine Möglichkeit um ganze Applikationen über mehrere Plattformen zur Verfügung zu Stellen. Mit Android, IOS und einen Zugang über eine Weboberfläche bietet Firebase einen Zugang über verschiedene Kanäle. Für Backend oder Administrationsfunktionen kann das SDK in Programmiersprachen wie Java, Javascript, Python oder GO genutzt werden.[37]

Während der Studienarbeit hat sich Firebase immer wieder Verändert und es sind neue Funktionen wie A/B Testing[38] oder Firestore[39] hinzugekommen. Durch die vielen Veränderungen wurde klar, dass an Firebase viel weiterentwickelt wird und sich daraus einiges entwickeln könnte.

Während der Studienarbeit wurde Firebase auch in einem privaten Projekt für einen Adventskalender benutzt. Die Realtime Database wurde für die Texte benötigt und Firebase Storage diente als Ablage für Bilder. Das App wurde in wenigen Stunden entwickelt und konnte verschenkt werden.

Es stellt sich jedoch die Frage, ab wann bei Firebase Kosten anfallen. Für den Beginn ist Firebase gratis und dadurch Ideal um einen Service zu entwickeln und zur Verfügung zu stellen. Wenn der Kundenstamm jedoch wächst und wächst, ist man ab einer Zeit gezwungen sein Backend zu skalieren. Wenn alle Kapazitäten von Firebase ausgenutzt werden, dann wird man von Google zu Kasse gebeten. Ab wann und ab wie vielen Benutzern dies der Fall sein wird, ist noch unklar. Weil Firebase so viele Funktionen und Möglichkeiten gibt, entsteht eine hohe Abhängigkeit zu dieser Plattform und es kann schwierig sein, sich wieder davon zu lösen.

Alles in allem bietet Firebase eine Plattform mit vielen Funktionen, welche sich ständig weiterentwickelt.

# 9 LITERATURVERZEICHNIS


1. Melendez, S. *Sometimes You're Just One Hop From Something Huge*. (Fast Company, 2014).
2. Tamplin, J. *Firebase is Joining Google!* (Firebase, Inc).
3. Paret, R. *Fabric is Joining Google*. (2017).
4. Ma, F. *Welcoming Fabric to Google*. (2017).
5. Johnson, J. Divshot: An Awesome Way to Design and Build Bootstrap Pages | Design Shack. Available at: https://designshack.net/articles/css/divshot-an-awesome-way-to-design-and-build-bootstrap-pages/. (Accessed: 17th December 2017)
6. Menge-Sonnentag, R. Google übernimmt Fabric von Twitter | heise Developer. Available at: https://www.heise.de/developer/meldung/Google-uebernimmt-Fabric-von-Twitter-3601871.html. (Accessed: 17th December 2017)
7. Firebase. Available at: https://firebase.google.com/. (Accessed: 6th October 2017)
8. Codelabs, G. Google Codelabs. (2016). Available at: https://codelabs.developers.google.com/. (Accessed: 11th December 2017)
9. Firebase. Add Firebase to Your Android Project | Firebase. Available at: https://firebase.google.com/docs/android/setup. (Accessed: 19th December 2017)
10. CocoaPods.org. Available at: https://cocoapods.org/. (Accessed: 25th September 2017)
11. Firebase Google. Add Firebase to your iOS Project | Firebase. Available at: https://firebase.google.com/docs/ios/setup. (Accessed: 21st December 2017)
12. Codelabs. Firebase iOS Codelab Swift. Available at: https://codelabs.developers.google.com/codelabs/firebase-ios-swift/?authuser=0#0. (Accessed: 25th September 2017)
13. Lemos, B. ios - Xcode doesn't recognize GoogleService-Info.plist file after adding Firebase via pods - Stack Overflow. Available at: https://stackoverflow.com/questions/37578378/xcode-doesnt-recognize-googleservice-info-plist-file-after-adding-firebase-via. (Accessed: 29th September 2017)
14. Propper, H. Firebase Crash Reporting | Firebase. Available at: https://firebase.google.com/docs/crash/. (Accessed: 6th October 2017)
15. Firebase Test Lab for Android. *Google Dev.*
16. Firebase Performance Monitoring | Firebase. Available at: https://firebase.google.com/docs/perf-mon/. (Accessed: 10th November 2017)
17. Firebase Performance Monitoring Automatic Traces | Firebase. Available at: https://firebase.google.com/docs/perf-mon/automatic. (Accessed: 17th October 2017)
18. Firebase Google. Get Started with Firebase Performance Monitoring for Android |

Firebase | Known Issues. Available at: https://firebase.google.com/docs/perf-mon/get-started-android#known_issues. (Accessed: 23rd October 2017)

19. Firebase Google. Get Started with Firebase Performance Monitoring for iOS | Firebase | Issues. Available at: https://firebase.google.com/docs/perf-mon/get-started-ios#known_issues. (Accessed: 19th December 2017)

20. Get Started with Analytics for Android | Firebase. Available at: https://firebase.google.com/docs/analytics/android/start/. (Accessed: 17th October 2017)

21. Log Events | Firebase. Available at: https://firebase.google.com/docs/analytics/android/events. (Accessed: 17th October 2017)

22. Automatically collected events - Firebase Help. Available at: https://support.google.com/firebase/answer/6317485. (Accessed: 17th October 2017)

23. Audiences - Firebase Help. Available at: https://support.google.com/firebase/answer/6317509?hl=en. (Accessed: 17th October 2017)

24. The Firebase Blog: Understanding Attribution in Firebase Analytics. Available at: https://firebase.googleblog.com/2017/04/understanding-attribution-in-firebase.html. (Accessed: 17th October 2017)

25. Google. Automatisch erfasste Nutzereigenschaften - Firebase-Hilfe. Available at: https://support.google.com/firebase/answer/6317486?hl=de. (Accessed: 21st December 2017)

26. Funnels - Firebase Help. Available at: https://support.google.com/firebase/answer/6317523?hl=en. (Accessed: 17th October 2017)

27. StreamView - Firebase Help. Available at: https://support.google.com/firebase/answer/7229836?hl=en. (Accessed: 17th October 2017)

28. DebugView - Firebase Help. Available at: https://support.google.com/firebase/answer/7201382?hl=en&utm_id=ad&authuser=0. (Accessed: 17th October 2017)

29. Google. Set User Properties | Firebase. Available at: https://firebase.google.com/docs/analytics/android/properties. (Accessed: 21st December 2017)

30. » Mobile App Tracking: Firebase Analytics or Google Analytics 360? Available at: http://content.infotrustllc.com/mobile-app-tracking-firebase-analytics-or-google-analytics-360/. (Accessed: 29th September 2017)

31. Apple. WKNavigationDelegate - WebKit | Apple Developer Documentation. Available at: https://developer.apple.com/documentation/webkit/wknavigationdelegate. (Accessed: 21st December 2017)

32. Was ist BigQuery? | BigQuery | Google Cloud Platform. Available at: https://cloud.google.com/bigquery/what-is-bigquery. (Accessed: 12th December 2017)

33. BigQuery – Data Warehouse für Analysen | Google Cloud Platform. Available at: https://cloud.google.com/bigquery/. (Accessed: 12th December 2017)

34. SQL-Referenz | BigQuery | Google Cloud Platform. Available at: https://cloud.google.com/bigquery/docs/reference/standard-sql/?hl=de. (Accessed: 13th December 2017)

35. Zu Standard SQL migrieren | BigQuery | Google Cloud Platform. Available at: https://cloud.google.com/bigquery/docs/reference/standard-sql/migrating-from-legacy-sql?hl=de. (Accessed: 13th December 2017)

36. cnlab. Personal Performance Testing | cnlab. Available at: https://www.cnlab.ch/de/performance/speedtest. (Accessed: 17th December 2017)

37. Add the Firebase Admin SDK to Your Server | Firebase. Available at: https://firebase.google.com/docs/admin/setup. (Accessed: 10th October 2017)

38. Firebase. Create Firebase Remote Config Experiments with A/B Testing | Firebase. Available at: https://firebase.google.com/docs/remote-config/abtest-config. (Accessed: 17th December 2017)

39. Firebase. Cloud Firestore | Firebase. Available at: https://firebase.google.com/docs/firestore/. (Accessed: 17th December 2017)

# 10 ABBILDUNGSVERZEICHNIS

# 11 TABELLENVERZEICHNIS



# 12 CODEVERZEICHNIS

# 13 Abkürzungsverzeichnis (Glossar)

| Abkürzung | Erklärung |
|---|---|
| Android | Betriebssystem für mobile Geräte von Google |
| API | Application Programming Internface, Programmierschnittstelle |
| APK | Android Package Kit, Dateiformat für Distribution und Installation von Android Apps |
| Backend | Software Teil, der für die Datenverarbeitung im Hintergrund zuständig ist. |
| Condition | dt. Bedingung |
| Crash | dt. Absturz, Absturz einer Applikation |
| Exception | dt. Ausnahme, unvorhergesehenes Ereignis |
| FCM | Firebase Cloud Messaging |
| Frame | Technik zur Unterteilung des Anzeigebereichs eines Webbrowsers |
| Getraced | Verb von Trace |
| IOS | Betriebssystem für mobile Geräte von Apple |
| Issue | dt. Problem |
| NoSQL | Datenbanken die einen nicht-relationalen Ansatz verfolgen. |
| OS | Operating System, Betriebssystem |
| View | Container für User Interface Elemente |

# 14 ANHANG

## 14.1 PROJEKTMANAGEMENT

### 14.1.1 Zeitliche Planung

Das Projekt wird im Rahmen der Studienarbeit durchgeführt. Dies hat zur Folge, dass während 14 Wochen am Projekt gearbeitet wird. Endgültiger Termin für die Abgabe ist der 22.Dezember 2017.

Der Aufwand pro Teammitglied beträgt insgesamt 240 Stunden. Somit entstehen ca. 17 Stunden Arbeitsaufwand pro Woche pro Teammitglied.

| Übersicht | |
|---|---|
| Projektdauer | 14 Wochen |
| Anzahl Projektmitglieder | 2 |
| Arbeitsstunden pro Person und Woche | 17.15 Stunden |
| Arbeitsstunden Total | 480 Stunden |
| Projektstart | 18.09.2017 |
| Projektende | 22.12.2017 |

*Tabelle 11: Übersicht Projektmanagement*

### 14.1.2 Zeitliche Auswertung

#### *14.1.2.1 Teammitglieder*

Die Teammitglieder trafen sich jeweils montags, dienstags und freitags an der HSR um an diesem Projekt zu arbeiten. So wurde sichergestellt, dass pro Woche ausreichend Zeit in das Projekt investiert wurde.

| Kalenderwoche | Fabian Schwyter | Stefan Diegas | Total Stunden |
|---|---|---|---|
| KW 38 | 17.00 | 17.00 | 34.00 |
| KW 39 | 16.00 | 14.50 | 40.50 |
| KW 40 | 18.00 | 19.00 | 37.00 |
| KW 41 | 17.00 | 19.00 | 36.00 |
| KW 42 | 17.00 | 18.00 | 35.00 |
| KW 43 | 16.00 | 16.50 | 34.50 |
| KW 44 | 18.00 | 18.50 | 36.50 |
| KW 45 | 15.00 | 15.00 | 30.00 |
| KW 46 | 20.00 | 18.00 | 38.00 |
| KW 47 | 13.00 | 17.00 | 30.00 |
| KW 48 | 13.00 | 12.00 | 25.00 |
| KW 49 | 15.00 | 16.50 | 31.50 |
| KW 50 | 22.00 | 21.00 | 43.00 |
| KW 51 | 30.00 | 28.50 | 58.50 |

*Tabelle 12: Aufwand pro Kalenderwoche*

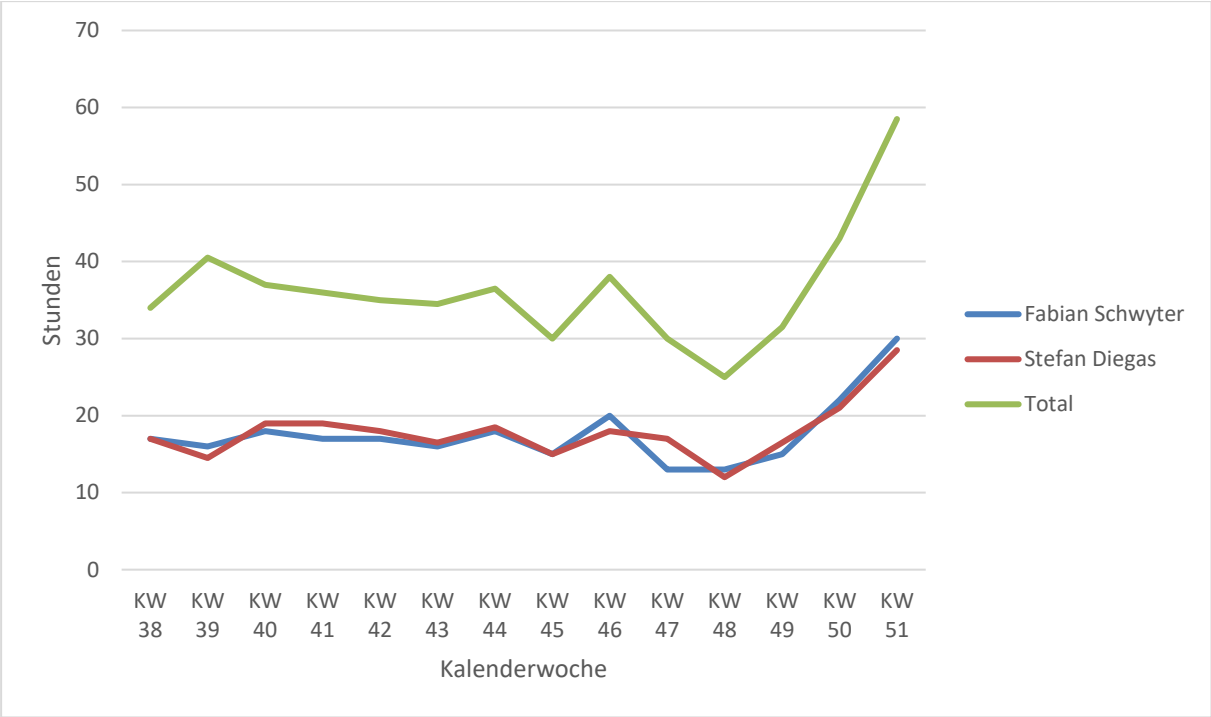



*Abbildung 45: Grafische Auswertung pro Teammitglied in Stunden*

**Auswertung**

Im Liniendiagramm ist ersichtlich, dass die Stunden bis KW 45 konstant waren, in Kalenderwoche 45 waren mehrere Testate in anderen Modulen abzugeben, wodurch sich dieser kleine Einbruch erklärt. Derselbe Grund kann zum Einbruch in der Kalenderwoche 48 verwendet werden. Gegen Ende wurde aufgrund der näher kommenden Abgabe viel mehr Zeit investiert, um unter anderem die Dokumentation wie auch andere Dokumente abzuschliessen. Insgesamt haben wir 509.5 Stunden investiert was leicht über den Soll von 480 Stunden ist.

#### *14.1.2.2 Bereich*

Die Arbeit wurde in verschiedene Bereiche unterteilt:

- FriendlyChat
- WebBrowser
- Dokumentation

| Bereich | Total Stunden |
|---|---|
| FriendlyChat | 111.5 |
| WebBrowser | 225.5 |
| Dokumentation | 172.5 |

*Tabelle 13: Aufwand nach Bereich*


*Abbildung 46: Grafische Auswertung nach Bereich*

### 14.1.3 Phasen

Das Projekt wurde grob in drei Phasen unterteilt.

#### *14.1.3.1 Analysephase*

In der Analysephase wurde Firebase als Ganzes kennengelernt. Dazu wurde Literatur sowie Online-Artikeln zu Firebase gesucht und verarbeitet. Anhand der App «FriendlyChat» wurden die einzelnen Funktionen von Firebase getestet und niedergeschrieben. Das Wissen floss in eine Präsentation ein, welche in eine Verlesung an der HSR gehalten werden kann.

#### *14.1.3.2 Entwicklungsphase*

Nach dem erlernten Wissen in der «Analysephase», ging es in die «Entwicklungsphase». Hier wurde ein Performance messender Web-Browser entwickelt. Die zu verwendenden Firebase Funktionen wurden am Anfang des Projektes vordefiniert. Die Entwicklungsphase endete mit einer ersten stabilen Version des Browsers.

#### *14.1.3.3 Auswertungsphase*

Anschliessend zur «Entwicklungsphase» kam die «Auswertungsphase», die gesammelten Daten des entwickelten Web-Browsers wurden ausgewertet. Die App wurde im Zusammenhang mit der Benutzerfreundlichkeit verbessert.

### 14.1.4 Sprints

Für das gesamte Projekt wurde eine agile Vorgehensweise nach SCRUM festgelegt. Die Entwickler legten zusammen mit dem Betreuer-Team am Anfang jeden Sprints die zu entwickelnden Features fest, priorisieren und planen diese ein.

Anhand der Arbeitspakete werden Features erstellt, wobei ein Features mehrere Tasks enthalten kann. Diese Tasks sind so definiert, dass sie jeweils von einem Gruppenmitglied alleine erledigt werden können. Nach Abschluss des Tasks wird dieser unter dem Vier-Augen-Prinzip angeschaut um die Qualität des Produkts hoch zu halten.

Durch die ständig ändernden Anforderungen, wurde eine Sprintdauer von zwei Wochen gewählt. Die Sprint Reviews passieren jeweils freitags um 15:00 Uhr. Dabei wird der aktuelle Stand der Arbeit besprochen. Noch nicht erledigte Tasks kommen in den Backlog und es wird neu priorisiert.

## 14.2 Meilensteine

### 14.2.1 MS01: Tutorial

**Deadline:** 29.09.2017

**Inhalt:**

- **Literaturrecherche**
- **Kennenlernen von Firebase**

### 14.2.2 MS02: Analysis

**Deadline:** 13.10.2017

**Inhalt:**

- **Vorlesung zu Firebase**
- **Vergleich Google Analytics & Firebase Analytics**

### 14.2.3 MS03: End of Elaboration

**Deadline:** 27.10.2017

**Inhalt:**

- **Machbarkeitsstudie Web-Browser**
    - **Events**
    - **Traces**
    - **Network Requests**

### 14.2.4 MS04: Implementation

**Deadline:** 24.11.2017

**Inhalt:**

- **Implementation Performance Web-Browser**
    - **Events**
    - **Traces**
    - **Network Requests**

### 14.2.5 MS05: Evaluation Data

**Deadline:** 08.12.2017

**Inhalt:**

- **Auswertung Performance Daten**
    - **Google BigQuery**
    - **Dashboards**

### 14.2.6 MS06: Delivery

**Deadline:** 22.12.2017

**Inhalt:**

- **App Benutzerfreundlichkeit verbessern**
- **Abgabe der Dokumentation**

| Meilenstein | Beschreibung | Datum |
|---|---|---|
| MS 1: Tutorial | Abschluss der Tutorials | 29.09.2017 |
| MS 2: Analysis | Source Code grob anschauen | 13.10.2017 |
| MS 3: End of Elab. | Demo Prototyp mit Firebase | 27.10.2017 |
| MS 4: Implementation | Implementation Performance Browser mit Firebase | 24.11.2017 |
| MS 5: Evaluation Data | Auswertung der gesammelten Daten | 08.12.2017 |
| MS 6: Delivery | Abgabe Bericht mit Anhang und Produkt | 22.12.2017 |

*Tabelle 14: Übersicht Meilensteine*

## 14.3 Risikomanagement

Zu Beginn des Projekts wurden die Risiken ermittelt, zur Halbzeit wurden diese aufgrund näheren Informationen angepasst. In den folgenden Abschnitten werden die Risiken genauer erläutert und ausgewertet.

### 14.3.1 Risiken

In der folgenden Tabelle werden die Risiken beschrieben und gewichtet. Die Indikatoren der Gewichtung sowie der Wahrscheinlichkeit liegen zwischen 1 und 5, wobei 1 die niedrigste Gewichtung hat und 5 die grösste Gewichtung hat.

| ID | Beschreibung | Gewicht | Wahrsch. |
|---|---|---|---|
| R01 | Firebase als Ganzes zu komplex | 4 | 1 |
| R02 | Performance Datenerhebung nicht erwartungsgemäss | 3 | 2 |
| R03 | Performance Browser nicht erwartungsgemäss | 4 | 4 |
| R04 | Längerer Ausfall eines Teammitglieds | 3 | 1 |

*Tabelle 15: Übersicht Risiken*

Multipliziert man die Gewichtung mit der Eintrittswahrscheinlichkeit, entsteht ein Risikowert zwischen 1 und 25.

### 14.3.2 Auswertung

Durch die fehlende Erfahrung mit Firebase, waren am Anfang Risiken vorhanden. Schnell konnten diese durch Nachforschungen und Prototypen verringert oder sogar eliminiert werden. Das grösste Risiko beherbergte R03, dieses konnte aber durch mehrere Prototypen eliminiert werden.

### 14.3.3 Umgang mit Risiken

Das grösste Risiko bestand, das der Performance Browser nicht erwartungsgemäss funktionierte. In regelmässigen Abständen wurde die App demonstriert und auch Abgegeben. Es wurde offen mitgeteilt, was Probleme verursachen könnte. In diversen Foren wurde von unserem Unterfangen abgeraten, wodurch wir die Wahrscheinlichkeit hoch ansetzten. Risiken wurden kontinuierlich mittels Prototypen reduziert.

## 14.4 ENTWICKLUNGSUMGEBUNG

Für die Realisierung der Arbeit wurden verschiedene Tools und Konzepte eingesetzt, die nachfolgend näher erläutert werden.

### 14.4.1 Kosten

Für die Umsetzung der Arbeit fallen lediglich die Kosten für das verarbeiten der Daten mit Google BigQuery an. Pro Terabyte Daten welche verarbeitet werden, fallen geringe 5 Dollar an. Bedenkt man was Firebase alles kostenlos bietet, sind diese Kosten sehr gering.

### 14.4.2 Hardware

Für die Entwicklung der IOS und Android Applikationen wurden die persönlichen Rechner (Surface Book und MacBook) verwendet. Als Ergänzung zum MacBook wurde auch ein Schulrechner mit Windows 10 verwendet. Für die IOS Entwicklung wurde ein iPad mini von der Schule ausgeliehen.

### 14.4.3 Software

#### *14.4.3.1 XCode*

Xcode ist die integrierte Entwicklungsumgebung von Apple für die Entwicklung von Apps für iOS, macOS, watchOS und tvOS. Der IDE ist proprietär und deshalb nur für macOS verfügbar, somit werden zwangsweise auch Apple Geräte zur Umsetzung vorausgesetzt. Die Nutzung von XCode ist kostenlos und kann gratis im App Store heruntergeladen werden.

#### *14.4.3.2 SWIFT*

Swift ist eine sehr junge Sprache zur Entwicklung von Apps. Sie löste die Sprache Objective-C als Hauptprogrammiersprache von Apple ab und ist seit Dezember 2015 Open Source. Zur Zeit ist Swift 4.0 die aktuellste Version.

#### *14.4.3.3 Android Studio*

Android Studio ist das Pendant zu Apples XCode. Im Gegensatz zur Apples IDE ist Android Studio frei verfügbar und unterstützt mehrere Plattformen. Android Studio ersetze das Android Developer Tool, welches Ende 2015 eingestellt wurde.

#### *14.4.3.4 Redmine*

Redmine ist eine freie, webbassierte Projektmanagementsoftware. Redmine wird weltweit bei mehreren grossen Projekten verwendet. Redmine ist plattform- und datenbankunabhängig.

#### *14.4.3.5 GitHub*

GitHub wird in erster Linie als zentrales Code Repository verwendet. GitHub erleichtert die Arbeit am gleichem Projekt.

#### *14.4.3.6 Firebase*

Firebase ist eine Mobile und Web Applikation Entwicklungsumgebung welche von Google 2014 aufgekauft wurde. Firebase soll das Programmieren wie auch die Verbreitung von Apps vereinfachen und stellt unter Firebase verschiedene Dienste zur Verfügung.

#### *14.4.3.7 BigQuery*

BigQuery ist das Data Warehouse von Google. Es wird keine Server Infrastruktur benötigt, sondern es genügt die Daten hochzuladen. Auf diese Daten können verschiedene SQL Abfragen gemacht werden.

#### *14.4.3.8 Visio*

Visio ist ein Visualisierungsprogramm von Microsoft. Es eignet sich um Prozesse und Zusammenhänge visualisiert darzustellen.

## 14.5 Richtlinien

Mit der Hilfe von Richtlinien soll eine gewisse Konsistenz über das gesamte Projekt erreicht werden.

### 14.5.1 Design

Die Performance Browser App beruht auf Material Design von Google. Als Farben wurden hauptsächlich die Farben von Firebase (Orange und Blau) verwendet.

### 14.5.2 Testing

Die Performance Browser App wurde mit Test Lab auf folgenden Geräten getestet und sollte Problemlos darauf funktionieren:

- Samsung Galaxy S7, API Level 23
- Samsung Galaxy J7, API Level 23
- Motorola Moto G4 Plus, API Level 23
- Motorola Moto G Play, API Level 23
- Google Nexus 5, API Level 23
- Huawei Honor 8, API Level 24
- Huawei P10, API Level 24

| Test execution | Duration | Locale | Orientation | Issues |
|---|---|---|---|---|
| Galaxy S7, API Level 23 | 5 min 12 sec | German (Switzerland) | Landscape | – |
| Moto G4 Plus, API Level 23 | 5 min 19 sec | German (Switzerland) | Landscape | – |
| Galaxy S7, API Level 23 | 5 min 13 sec | German (Switzerland) | Portrait | – |
| Moto G4 Plus, API Level 23 | 5 min 13 sec | German (Switzerland) | Portrait | – |
| Nexus 5, API Level 23 | 5 min 20 sec | German (Switzerland) | Landscape | – |
| Nexus 5, API Level 23 | 5 min 14 sec | German (Switzerland) | Portrait | – |

*Abbildung 47: Robotest auf verschiedenen Geräten*

## 14.6 QUALITÄTSMASSNAHMEN

### 14.6.1 Entwicklung

Während der Arbeit wurde stets auf eine hohe Code Qualität sowie auf eine stabile Applikation geachtet, dafür wurden folgende Massnahmen getroffen.

### 14.6.2 Zentrale Datenablage OneDrive

Sämtliche Dokumente wurden auf OneDrive abgelegt. OneDrive ermöglicht einen Zugriff von jedem Internetfähigem Client. Ausserdem ist das gleichzeitige Bearbeiten von Office Dokumenten mit OneDrive möglich.

### 14.6.3 Projektmanagement mit Redmine

Definierte Arbeitspakete wurden den selbst definierten Meilensteinen hinzugefügt. Zeit Buchung direkt auf den Arbeitspaketen und jederzeit ist der Status des Arbeitspakets ersichtlich sowie eine prozentuale Anzeige des Fortschritts.

### 14.6.4 Git Branching

Der master Branch definiert zu jederzeit den aktuellen produktiven Status. Parallel wird während der Entwicklung für die neuen Features ein Branch erstellt, sogennante Feature-Git-Branches. Waren die Änderungen klein, wurden mehrere Features in einem Feature-Git-Branch realisiert.

### 14.6.5 Code Reviews

Code Reviews helfen «Code Smells» oder auch Logikfehler aufzudecken und die Qualität des Codes zu verbessern. Code Reviews wurden am Ende eines Sprints durchgeführt. Bei Unsicherheiten betreffend der Logik, wurden kleinere Reviews auch während einem Sprint durchgeführt.

### 14.6.6 Testing

Die Android Version der Performance Browser App wird laufend mit Firebase Test Lab auf verschiedenen physikalischen sowie virtuellen Geräten getestet. Die Android Versionen 6.0 und 7.0 werden für die Tests ausgewählt. Als Testart werden sogenannte «Robo-Tests» verwendet, welche mit Hilfe von künstlicher Intelligenz einen User Simulieren und probieren die App zum Absturz zu bringen.

### 14.6.7 Usablity Testing

Die App wurde regelmässig vom cnlab Team getestet, verschiedene Rückmeldungen konnten in die Entwicklung der App einfliessen. Es wurde ausserdem ein Usability Tests erstellt, welcher weiter unten angehängt ist.

### 14.6.8 Code Style Guidelines

Um einen einheitlichen Code Style zu gewährleisten, basiert der Code auf dem Style Guide von «AOSP Java Code Style for Contributors».

## 14.7 Persönliche Berichte

### 14.7.1 Stefan Diegas

Ich freute mich nach den guten Erfahrungen im Engineering Projekt über längere Zeit an einem Projekt zu arbeiten. Mit der Studienarbeit über Firebase haben wir meiner Meinung nach ein interessantes und aktuelles Thema erhalten. Das war mir persönlich sehr wichtig um mit Herzblut an der Arbeit mitzuarbeiten und so schlussendlich mit dem Geleistetem zufrieden zu sein.

Der Einstieg in die Arbeit war schwierig, durch die am Anfang offene Aufgabenstellung wussten wir im Gegensatz zum Engineerin Projekt nicht, was schlussendlich abgeliefert werden sollte. Auch handelt es sich bei unserer Arbeit nicht um ein klassisches Produkt als Arbeit, sondern um eine eher Wissenschaftliche Arbeit. Dies erschwerte die Dokumentation, ich wusste nicht genau, welche Dokumente am Ende Abgegeben werden mussten.

Im grossem und Ganzen bin ich sehr zufrieden mit der Arbeit, zu mehreren Knacknüssen haben wir schnell gute Lösungsansätze gefunden. Beim Performance Browser mussten wir einen eigenen Browser konstruieren der auch noch die Performance misst. Durch die Anwendung des Interceptor Pattern konnten wir die Requests Messen, jedoch war es schwierig die selbst erstellten Requests auch schön zurückzugeben und darzustellen. Wenn man bedenkt wie komplex ein Mobiler Browser ist, haben wir die Herausforderungen gut gelöst.

### 14.7.2 Fabian Schwyter

Die Studienarbeit war eine interessante Erfahrung für mich und es machte Spass Firebase kennen zu lernen. Eine grosse Freude war, dass es sich bei Firebase um eine relativ neue Plattform handelte und man schnell merkte, dass diese sehr Zukunftsträchtig sein könnte.

Zu Beginn war ich mir nicht ganz sicher, wohin diese Studienarbeit hinführen würde. Vor allem da nach 7 Wochen eine Zwischenpräsentation stattgefunden hat und es noch nicht klar war wie es weitergehen sollte.

An BigQuery hatte ich grosse Freude und es war eindrücklich, welche Daten alle gesammelt werden können. Auch Interessant war, was alles wirklich für Auswertungen gemacht werden können. Ein User Tracking ist theoretisch möglich und ist nicht sehr schwierig nachzubauen. Ich werde nun darauf achten, dass ich Smartphone Berechtigungen nicht einfach Blind weggebe.

Für eine weitere Arbeit achte ich darauf, dass vom Anfang an alle Anforderungen definiert werden und so besser nach Plan gearbeitet werden kann. Dies war jedoch bei dieser Arbeit schwierig, da es sehr offen war.

## 14.8 KONTAKTADRESSEN

### 14.8.1 Studenten

**Fabian Schwyter**

Oberer Hegner 2

8730 Uznach

+41 79 916 35 65

fab.schwyter@gmail.com

**Stefan Diegas**

Rüfi 3

8753 Mollis

+41 76 367 76 99

stefandiegas@gmail.com

### 14.8.2 Betreuer

**Prof. Dr. Peter Heinzmann**

Obere Bahnhofstrasse 32b

8640 Rapperswil-Jona

+41 55 214 33 30

peter.heinzmann@cnlab.com

**Eric Franke**

Obere Bahnhofstrasse 32b

8640 Rapperswil-Jona

+41 55 214 33 37

eric.franke@cnlab.ch

**Patrick Eichler**

Obere Bahnhofstrasse 32b

8640 Rapperswil-Jona

+41 55 214 33 39

patrick.eichler@cnlab.com

## 14.9 Quell Code

Der Quellcode ist auf GitHub verfügbar. Das GitHub Repository ist Private eingestellt. Falls Interesse am Zugang besteht soll Stefan Diegas oder Fabian Schwyter kontaktiert werden.

**Stefan Diegas**

Stefan.diegas@gmail.com

sdiegas@hsr.ch

**Fabian Schwyter**

fab.schwyter@gmail.com

fschwyte@hsr.ch

| Beschreibung | Link |
|---|---|
| WebBrowser Android | https://github.com/sdiegas/WebBrowser_SA_Android |
| WebBrowser IOS | https://github.com/fdschwyter/WebBrowser_SA_IOS |
| Querys für BigQuery | https://github.com/fdschwyter/BigQuery |
| Friendly Chat IOS Tutorial | https://github.com/fdschwyter/FriendlyChat_SA_IOS |
| Friendly Chat Tutorial | https://github.com/sdiegas/FriendlyChat_SA_Android |

*Tabelle 16: Source Code Links*

## 14.10 USABILITY TEST

### 14.10.1 Auswahl Testperson

Bei der Auswahl der Testperson wurde darauf geachtet, dass die Person keinen speziellen Informatik Hintergrund hat.

### 14.10.2 Test Gerät

Als Test Gerät wurde das Smartphone der Testperson verwendet, dem Benutzer wurde dafür die APK per Mail geschickt in einem anderen Test Case wurde die APK per Link verschickt. Die Installation ist Teil des Usability Tests.

### 14.10.3 Test Cases

#### *14.10.3.1 Test Case 1*

| | |
|---|---|
| **Status** | APK per Mail geschickt, WebBrowser App deinstalliert |
| **Vorgabe** | Eine Installation durchspielen |
| **Ziel** | 1. Herausfinden, ob die Installation durch verschicken der APK verständlich ist.<br>2. Herausfinden, ob Probleme mit verschiedenen Geräteherstellern bestehen. |

**Szenario**

Sie wollen die App installieren, welche Sie per Mail erhalten haben. Dafür machen Sie Ihren Mail Client auf und laden den Anhang herunter, anschliessend installieren Sie die App.

#### *14.10.3.2 Test Case 2*

| | |
|---|---|
| **Status** | APK per Link geschickt, WebBrowser App deinstalliert |
| **Vorgabe** | Eine Installation durchspielen |
| **Ziel** | 1. Herausfinden, ob die Installation durch verschicken der APK verständlich ist.<br>2. Herausfinden, ob Probleme mit verschiedenen Geräteherstellern bestehen. |

**Szenario**

Sie wollen die App installieren, welche Sie per Link erhalten haben. Dafür öffnen Sie den Link mit dem Browser Ihrer Wahl, anschliessend installieren Sie die App.

#### *14.10.3.3 Test Case 3*

| | |
|---|---|
| **Status** | App installiert |
| **Vorgabe** | Auf die Webseite der HSR zugreifen |
| **Ziel** | 1. Herausfinden wie intuitiv die App bedient werden kann um auf eine Seite zu gelangen. |

**Szenario**

Die App ist installiert, Sie sollen nun über die App auf die HSR Webseite gelangen.

#### *14.10.3.4 Test Case 4*

| | |
|---|---|
| **Status** | App installiert |
| **Vorgabe** | Auf die Webseite der HSR zugreifen, anschliessend auf Blick gehen und schlussendlich googeln bei welcher Temperatur Wasser die grösste Dichte hat. |
| **Ziel** | 2. Herausfinden wie intuitiv die App bedient werden kann. |

**Szenario**

Die App ist installiert, Sie sollen nun über die App auf die HSR Webseite gelangen, anschliessend gefolgt von der Blick Seite, am Ende sollen Sie herausfinden bei welcher Temperatur Wasser die grösste Dichte hat.

#### *14.10.3.5 Test Case 5*

| | |
|---|---|
| **Status** | App installiert |
| **Vorgabe** | Gehen Sie über das Adressfeld auf blick.ch |
| **Ziel** | 3. Herausfinden wie intuitiv die App bedient werden kann. |

**Szenario**

Die App ist installiert, Sie sollen nun über die Adressleiste oben auf blick.ch zugreifen und nicht über Google.

### 14.10.4 Auswertung

#### *14.10.4.1 Durchführung 1*

| | |
|---|---|
| **Datum** | 16.12.2017 |
| **Moderator** | Stefan Diegas |

**Test Case 1**

*Vorgehen*

1. Testuser1 hat das Smartphone vor Augen, erhält E-Mail. «So das Mail hast du mir geschickt, stimmts?».
2. Geht auf den Chrome Browser, öffnet GMAIL im mobilen Browser.
3. «So jetzt muss ich wahrscheinlich da draufklicken».
4. Klickt auf den Anhang, Anhang wird heruntergeladen.
5. Klickt auf den heruntergeladenen Anhang, Applikations-Installation beginnt.
6. Klickt auf Installieren, Installation beginnt und schliesst erfolgreich ab.
7. «Super, hat geklappt».

*Erkenntnisse*

- Im Nachhinein wurde festgestellt, das die Option «Unbekannte Quellen» nicht sichergestellt wurde. In unserem Fall hatte der Benutzer bereits die «Unbekannten Quellen» erlaubt. Die App muss in einer neuen Version signiert werden.

**Test Case 2**

*Vorgehen*

1. Testuser1 erhält APK per Link in Whatsapp.
2. «Habs bekommen».
3. Klickt auf den Link der mitgeschickt wurde.
4. Klickt auf «Download APK».
5. Installiert die APK ohne Worte.
6. «So wurde installiert».

*Erkenntnisse*

- Im Nachhinein wurde festgestellt, dass die Option «Unbekannte Quellen» nicht sichergestellt wurde. In unserem Fall hatte der Benutzer bereits die «Unbekannten Quellen» erlaubt. Die App muss in einer neuen Version signiert werden.
- Die Installation per Link ist um einiges schneller.

**Test Case 3**

*Vorgehen*

1. Öffnet die App.
2. «Das muss ich erlauben, wieso braucht Ihr Telefonberechtigung?».
3. Erlaubt alle Berechtigungen.
4. «Wie heisst die Seite, HRS?».
5. Tippt HSR in die Google-Suchleiste ein.
6. Klickt auf das Suchresultat zur HSR.

*Erkenntnisse*

- Der Benutzer hat nicht versucht die Seite in der Adressleiste einzugeben, sondern hat mit der Google Startseite danach gesucht.
- Die Berechtigung «Telefonieren» verwirrt den Benutzer, falls die App im Play Store veröffentlicht wird, angeben wieso dieses Recht benötigt wird.

**Test Case 4**

*Vorgehen*

1. Öffnet die App.
2. Tippt «HSR» in Google ein.
3. Geht über das Google Suchresultat auf www.hsr.ch
4. Geht per Back Button zurück zu den Google Resultaten der HSR.
5. Tippt in die Google Suche «blick» ein.
6. Geht über das Google Suchresultat auf www.blick.ch
7. Geht per Back Button zurück zu den Google Resultaten von Blick.
8. Tippt ein: «Temperatur dichte Wasser» wählt Vorschlag von Google aus: Temperatur Dichte Wasser am grössten».
9. Geht auf eine unbekannte Seite mit der Lösung.

*Erkenntnisse*

- Der Benutzer braucht den «Back Button» oft um zu navigieren.
- Google ist sehr beliebt um Webseiten aufzurufen, auch wenn man die URL kennen würde.
- Google Vorschläge werden genutzt, wenn diese eingeblendet werden.

**Test Case 4**

*Vorgehen*

1. Öffnet die App.
2. Tippt «www.blick.ch» in Adressleiste ein, Enter.
3. «Kommt Google»
4. Klickt auf das Suchresultat von Google um auf Blick zu gelangen.

*Erkenntnisse*

- Es ist zu viel verlangt, dass der Benutzer auch das Protokoll mitgibt.
- Das Ziel ist dank der Google Anfrage bei ungültiger URL trotzdem erreicht worden.