

Firestore

Integrate with Firebase Crashlytics,
Google Analytics & Firebase Remote
Config



Firebase

- ☐ Firebase Crashlytics
- ☐ Google Analytics
- ☐ Firebase Remote Config
- ☐ Setup
- ☐ Study Case

Objektif sesi

- Mengintegrasikan sistem pelaporan crash real-time yang membantu mengidentifikasi, mendiagnosa, dan memperbaiki masalah kritis yang menyebabkan crash aplikasi.
- Mengintegrasikan alat analisis untuk memahami bagaimana pengguna berinteraksi dengan aplikasi, yang mencakup pelacakan pengguna, sesi, dan perilaku pengguna.
- Mengimplementasikan sistem konfigurasi yang memungkinkan pengembang untuk mengubah perilaku dan tampilan aplikasi dari jarak jauh tanpa perlu merilis update baru.

Firebase Crashlytics

Firebase Crashlytics adalah layanan yang menyediakan pelaporan crash real-time untuk aplikasi iOS dan Android. Ini membantu pengembang dengan:

- Pelacakan Crash: Mengidentifikasi, secara rinci, alasan dan keadaan crash yang terjadi di aplikasi.
- Analisis Crash: Menyediakan informasi tentang frekuensi, pola, dan lokasi crash dalam kode aplikasi.

Firebase Crashlytics

- **Pemilahan Prioritas dan Isu:** Mengidentifikasi masalah yang paling memengaruhi pengguna, memungkinkan pengembang untuk memprioritaskan perbaikan.
- **Pemecahan Masalah yang Efisien:** Dengan log crash, stack trace, dan informasi perangkat, pengembang bisa lebih cepat mengidentifikasi dan menangani penyebab crash.

Google Analytics

Google Analytics untuk Firebase adalah solusi analisis yang gratis dan kuat untuk aplikasi mobile. Fitur-fiturnya meliputi:

- Pelacakan Pengguna dan Sesi: Memahami bagaimana pengguna berinteraksi dengan aplikasi, termasuk frekuensi dan durasi penggunaan.
- Analisis Perilaku Pengguna: Menyediakan insight tentang perilaku pengguna, seperti fitur atau layar yang paling sering dikunjungi.

Google Analytics

- Pelacakan Event: Mengumpulkan data tentang bagaimana pengguna berinteraksi dengan fitur tertentu di aplikasi Anda.
- Integrasi dengan Produk Google Lainnya: Memungkinkan penggabungan data dari berbagai sumber dan penggunaan alat analitik lanjutan.

Firestore Remote Config

Firestore Remote Config adalah layanan cloud yang memungkinkan pengembang aplikasi mobile untuk mengubah perilaku dan tampilan aplikasi mereka tanpa perlu menerbitkan update aplikasi. Fitur utamanya adalah:

- Konfigurasi Aplikasi yang Fleksibel: Mengubah elemen seperti warna, teks, dan tata letak tanpa perlu update aplikasi.
- Pengujian A/B dan Segmentasi Pengguna: Melakukan eksperimen dengan berbagai konfigurasi untuk segmen pengguna yang berbeda.

Firestore Remote Config

- Penerapan Perubahan Real-time: Mengupdate konfigurasi di server dan menerapkannya pada aplikasi secara langsung.
- Kustomisasi Berdasarkan Kondisi: Menetapkan kondisi tertentu seperti lokasi pengguna, perilaku penggunaan, atau target versi aplikasi untuk konfigurasi yang berbeda.

Setup

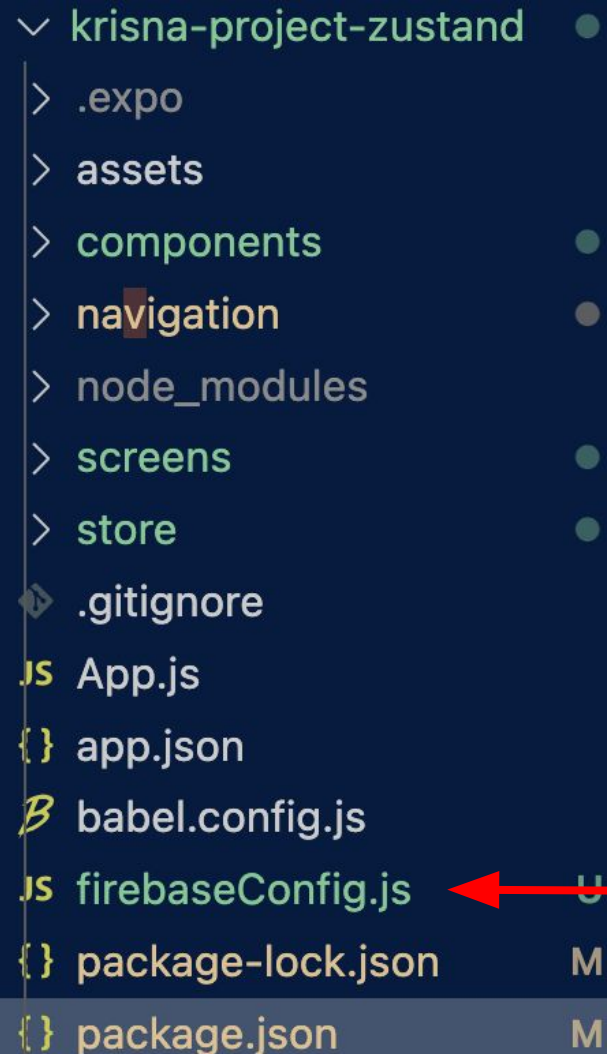
Buat Proyek Firebase:

- Buka <https://console.firebase.google.com/>
- Klik "Add project" dan ikuti langkah-langkah untuk membuat proyek baru.

Setup

Inisialisai SDK di Project

- Buat file firebaseConfig.js
Dirroot Project



```
▼ krisna-project-zustand
  > .expo
  > assets
  > components
  > navigation
  > node_modules
  > screens
  > store
  ◆ .gitignore
  JS App.js
  {} app.json
  B babel.config.js
  JS firebaseConfig.js
  {} package-lock.json
  {} package.json
```

Setup

Install the SDK

- After you have created your Expo project, you can install the Firebase JS SDK using the following command:

```
npm install @react-native-firebase/app  
@react-native-google-analytics/bridge  
@react-native-firebase/crashlytics  
@react-native-firebase/remote-config
```


Setup Crashlytics

- **Inisialisasi Crashlytics di Kode:**

Import Crashlytics di file utama React Native Anda (biasanya App.js).

Inisialisasi Crashlytics: `crashlytics().log('App mounted.');`

- **Testing dan Debugging:**

Build aplikasi Anda untuk iOS/Android dan test untuk memastikan Crashlytics bekerja.

Anda dapat memaksa crash untuk testing dengan `crashlytics().crash();`

Setup Google Analytics:

- **Konfigurasi Google Analytics:**

Buat properti baru di Google Analytics untuk aplikasi mobile Anda. Dapatkan "Tracking ID" dari Google Analytics.

- **Inisialisasi Google Analytics di Aplikasi React Native:**

Import Google Analytics library ke dalam aplikasi Anda.

Inisialisasi Google Analytics dengan Tracking ID yang didapat. Contoh kode:

Setup Google Analytics:



```
1  import { GoogleAnalyticsTracker } from '@react-native-google-analytics/bridge';  
2  
3  const tracker = new GoogleAnalyticsTracker('YOUR_TRACKING_ID');  
4
```

Setup Google Analytics:

- **Mengirim Event dan Data ke Google Analytics:**

Setelah menginisialisasi tracker, Anda dapat mulai mengirim data ke Google Analytics.

Contoh untuk mengirim event:

```
tracker.trackEvent('test_category', 'test_action');
```

Anda juga dapat melacak halaman layar/screen views:

```
tracker.trackScreenView('HomeScreen');
```


Setup Google Analytics:

- **Konfigurasi Firebase di Proyek Anda:**

Tambahkan file konfigurasi Firebase (google-services.json untuk Android dan GoogleService-Info.plist untuk iOS) ke proyek Anda. File-file ini dapat didapatkan dari Firebase Console setelah Anda menambahkan aplikasi Android atau iOS ke proyek Firebase Anda.

Pastikan Anda telah mengikuti petunjuk konfigurasi khusus untuk platform masing-masing.

Setup Google Analytics:

- **Inisialisasi Remote Config di Kode Aplikasi:**

Import Remote Config di aplikasi React Native Anda:

```
import remoteConfig from '@react-native-firebase/remote-config';
```

Inisialisasi Remote Config dan tetapkan nilai default (opsional):

```
await remoteConfig().setDefaults({  
  welcome_message: 'Welcome to our app!'  
});
```

Setup Google Analytics:

- **Pengambilan Konfigurasi dari Firebase:**

Ambil konfigurasi terbaru dari Firebase:

```
await remoteConfig().fetchAndActivate();
```

Setelah konfigurasi diambil, Anda dapat menggunakan nilai tersebut di aplikasi

Anda:

```
const welcomeMessage = remoteConfig().getString('welcome_message');
```

Setup Google Analytics:

- **Pengaturan Cache dan Pengambilan Interval:**

Anda dapat mengatur interval waktu untuk cache konfigurasi agar tidak terlalu sering mengambil data dari server, yang dapat berpengaruh pada kuota dan kinerja:

```
await remoteConfig().setConfigSettings({  
    minimumFetchIntervalMillis: 60000, // 1 menit  
});
```


Kasus Penggunaan: Aplikasi E-commerce "ShopSmart"

- **Struktur Folder**

```
ShopSmartApp/  
|  
├─ android/  
|   └─ (Konfigurasi Android dan file gradle)  
|  
├─ ios/  
|   └─ (Konfigurasi iOS dan file Xcode)  
|  
├─ src/  
|   ├─ components/  
|   |   ├─ ProductCard.js  
|   |   └─ CategoryBanner.js  
|   |   └─ ...  
|   |  
|   └─ screens/  
|   |   ├─ HomeScreen.js  
|   |   ├─ ProductDetailScreen.js  
|   |   └─ CartScreen.js  
|   |   └─ ...  
|   |  
|   └─ analytics/  
|   |   └─ AnalyticsService.js  
|   |  
|   └─ config/  
|   |   └─ RemoteConfig.js  
|   |  
|   └─ utils/  
|   |   └─ CrashlyticsService.js
```

Kasus Penggunaan: Aplikasi E-commerce "ShopSmart"

- **CrashlyticsService.js (utils/CrashlyticsService.js)**

```
import crashlytics from '@react-native-firebase/crashlytics';

export const logError = (error) => {
  crashlytics().recordError(error);
};
```

Kasus Penggunaan: Aplikasi E-commerce "ShopSmart"

- **AnalyticsService.js (analytics/AnalyticsService.js)**

```
import analytics from '@react-native-firebase/analytics';  
  
export const logEvent = (eventName, params) => {  
  analytics().logEvent(eventName, params);  
};
```

Kasus Penggunaan: Aplikasi E-commerce "ShopSmart"

- RemoteConfig.js (config/RemoteConfig.js)

```
import remoteConfig from '@react-native-firebase/remote-config';

export const initializeRemoteConfig = async () => {
  await remoteConfig().setDefaults({
    promo_banner: 'default_banner.jpg',
  });
  await remoteConfig().fetchAndActivate();
};

export const getPromoBanner = () => {
  return remoteConfig().getString('promo_banner');
};
```


Kasus Penggunaan: Aplikasi E-commerce "ShopSmart"

- App.js

```
import React, { useEffect } from 'react';
import { initializeRemoteConfig } from './config/RemoteConfig';
import { logError } from './utils/CrashlyticsService';

const App = () => {
  useEffect(() => {
    initializeRemoteConfig().catch(err => logError(err));
  }, []);

  // Rest of your app code

  return (
    // Your app component
  );
};

export default App;
```

Kasus Penggunaan: Aplikasi E-commerce "ShopSmart"

- HomeScreen.js (screens/HomeScreen.js)

```
import React, { useEffect } from 'react';
import { getPromoBanner } from '../config/RemoteConfig';
import { logEvent } from '../analytics/AnalyticsService';

const HomeScreen = () => {
  useEffect(() => {
    logEvent('screen_view', { screen_name: 'HomeScreen' });
  }, []);

  const promoBanner = getPromoBanner();

  // Render the home screen with promoBanner
  return (
    // Your home screen layout
  );
};

export default HomeScreen;
```



Thank you