Personsökarprojekt

Cloud Functions - komponentbeskrivning

Abstract

Detta dokument beskriver Cloud Functions-komponentens egenskaper och gränssnitt. Komponenten inkluderar funktioner för databashantering och autentisering. Syftet med komponenten är att ha funktioner som körs oberoende av klienter och möjliggör för hårdvaran att utan begränsning hämta data från databasen.

Version History

Date	Version	Author	Description
2020-05-25	1.0	Adam Liliemark	Färdigställt
2020-05-19	0.3	Adam Liliemark	Justeringar efter nya förutsättningar
2020-05-05	0.2	Adam Liliemark	Andra utkast
2020-04-29	0.1	Adam Liliemark	Första utkast

An Essential Unified Process Document

IVAR JACOBSON CONSULTING

Table of Contents

1 Introduktion

Syftet med detta dokument är att beskriva Cloud Functions-komponenten (CF) i personsökarprojektet och dess egenskaper samt gränssnitt. Detta för att:

- underlätta för andra studenter att implementera liknande funktioner,
- underlätta vid implementering av fler funktioner,
- beskriva hur CF-komponenter kan testas

1.1 Avgränsning

Detta dokument behandlar endast CF-komponentens krav i personsökarprojektet, det gränssnitt som CF-komponenten skapas i samt vilka inställningar som behövs för att kunna implementera nya funktioner.

Övriga delar, såsom enhetstester och hur andra komponenter beror på CF-komponenten, berörs inte eller endast ytligt.

1.2 Överblick

Detta dokument innehåller följande sektioner:

- **Funktionella krav** – de krav som ställs på komponenten för att den skall anses fylla sitt syfte,
- **Begränsningar avseende implementation** – de krav som komponenten ställer på utvecklingsmiljö,
- **Gränssnitt och funktionsbeskrivning** – komponentens gränssnitt och de funktioner som ingår i komponenten,
- **Gränssnittsberoenden** – de gränssnitt som komponenten är beroende av,
- **Referenser** – hänvisning till referenser

2 Funktionella krav

Komponenten har följande funktionella krav:

- Antal lästa meddelanden i hårdvarans träd skall begränsas till 20,
- Flytta meddelanden från oläst till läst i hårdvarans träd när en hårdvaruanvändare markerar meddelandet som läst, samt markera meddelandet som läst i mjukvaruanvändarens träd,
- Skapa grundinställningar för nya mjukvaruanvändare,
- Radera alla spår i databasen av en mjukvaruanvändare vid kontoradering

3 Begränsningar avseende implementation

Komponenten körs på serversidan av applikationen och har en rad begränsningar att beakta vid implementering av nya funktioner.

3.1 Implementationsmiljö

Komponentens funktioner körs på Googles tjänst Firebase, där en Node.js-miljö körs.

3.2 Programmeringsspråk

JavaScript (ES6) och TypeScript.

3.3 Kodstruktur

Alla funktioner placeras i *./functions* och kallas från *./functions/index.js*. Externa beroenden specificeras i *./functions/package.json*.

Funktioner exporteras i JS ES6 med följande syntax:

exports.<Funktionsnamn>

Hjälpfunktioner behöver ej exporteras.

Åtkomst till Firebasefunktioner, *firebase-functions* (3) och *firebase-admin* (1), måste skapas genom:

functions = require(firebase-functions)

admin = require(firebase-admin)

Firebaseinstansen måste initieras med:

admin.initializeApp();

Alla funktioner måste ha ett returvärde, som kan vara null.

4 Gränssnitt och funktionsbeskrivning

Komponenten har inget publikt gränssnitt.

Komponenten har 5 lägesförändrande funktioner:

changeReadStatus - När en hårdvaruanvändare markerar ett meddelande som läst, ändras *read*-flaggan i hårdvaruanvändarens träd till *true*. Därefter flyttas meddelandet från

./msg/<HWID>*/unread/*<MSGID>

till

./msg/<HWID>*/read/*<MSGID>.

Slutligen uppdateras *read*-flaggan i

./users/<UID>/messages/<MSGID>

till *true*.

Funktionen startas av ändringar i *./msg/*<MSGID>/.

limitReadMsgs - Begränsar antalet meddelanden i

./msg/<HWID>*/read/*

till 20.

Funktionen startas av skrivning till

./msg/<HWID>*/read/*.

removeUserData - När en mjukvaruanvändare väljer att avsluta sitt konto, raderas allting i

./users/<User ID>/.

Funktionen startas av att *firebase-auth* raderar en användare.

removeUsersMessages - När en mjukvaruanvändare raderas ändras *sender*-fältet i alla meddelanden som denna användaren gjort. Det nya värdet blir *Deleted user*.

Funktionen startas av radering i

./users/<UID>/

createUserDefaultSettings - När en användare skapas genom *firebase-auth* skapas grundstrukten för denna användare. Följande träd skapas automatiskt:

Figur 1: Grundinställningar för användare G20KSs3AQ6WWYAd1g3apB8WJgFl2

Funktionen startas av att *firebase-auth* skapar en ny användare.

5 Gränssnittsberoenden

Komponenten är beroende av Firebasegränssnitten *firebase-admin* (1), *firebase-auth* (2), *firebase-functions* (3), *firebase-database* (4).

Utöver dessa är komponenten beroende av implementationen av databasen. En viss trädstruktur behövs för att komponenten ska fungera. Se figur nedan för strukturen som krävs. Utan denna kommer funktionerna inte att hitta rätt vid traversering av databasen.

Figur 2: Grundstruktur i databasen

6 Testning

För att skapa enhetstester för komponenten kan gränssnittet *firebase-functions-test* (5) användas tillsammans med ett testningsramverk, till exempel *Mocha* (6). Tester skapas då på samma sätt som vanliga funktioner testas med hjälp av *Node Package Manager* (npm). En eller flera testfiler skapas med namnet **.test.js*, och kommer att köras när kodbasen publiceras hos Firebase.

7 Referenser

1. Firebase Admin, https://firebase.google.com/docs/admin/setup (hämtad 2020-05-25)
2. Firebase Authentication, https://firebase.google.com/docs/auth (hämtad 2020-05-25)
3. Firebase Functions, https://firebase.google.com/docs/functions (hämtad 2020-05-25)
4. Firebase Database, https://firebase.google.com/docs/database (hämtad 2020-05-25)
5. Unit testing of Cloud Functions, https://firebase.google.com/docs/functions/unit-testing (hämtad 2020-05-25)
6. Mocha, https://mochajs.org/ (hämtad 2020-05-25)