Bootcamp DevSuperior 2.0

Push Notifications com Firebase

Código fonte

https://github.com/acenelio/poc-push

Links

Guias

Página inicial do FCM
https://firebase.google.com/docs/cloud-messaging

Visão geral da arquitetura
https://firebase.google.com/docs/cloud-messaging/fcm-architecture

Firebase para admin (servidor)
https://firebase.google.com/docs/admin/setup

Firebase para cliente web
https://firebase.google.com/docs/web/setup

Quickstart JavaScript
https://github.com/firebase/quickstart-js

Quickstart Java
https://github.com/firebase/quickstart-java

Repo Admin Java
https://github.com/firebase/firebase-admin-java

API Java
https://firebase.google.com/docs/reference/admin/java/reference/packages

Outros

https://support.google.com/chrome/answer/3220216?co=GENIE.Platform%3DDesktop&hl=pt

https://medium.com/@sazzadsazib/how-to-send-notification-to-react-app-using-firebase-f257d4bdca28

https://stackoverflow.com/questions/62225339/how-to-use-process-env-in-a-react-service-worker#_=_

https://medium.com/@singh.pankajmca/fcm-integration-with-spring-boot-to-send-push-notification-from-server-side-1091cfd2cacf

https://stackoverflow.com/questions/58386934/web-firebase-messaging-onmessage-not-fired-but-background-notification-perfec

https://firebase.google.com/docs/cloud-messaging/js/receive

Criar projeto web

npx create-react-app front-web --template typescript --use-npm

npm install firebase

Github SSH

```
ssh-keygen -t rsa -b 4096 -C "example@example.com"
eval $(ssh-agent -s)
ssh-add ~/.ssh/id_rsa
clip < ~/.ssh/id_rsa.pub
```

Códigos

firebase-messaging-sw.js

```
importScripts('https://www.gstatic.com/firebasejs/8.2.4/firebase-app.js');
importScripts('https://www.gstatic.com/firebasejs/8.2.4/firebase-messaging.js');

// https://github.com/react-boilerplate/react-boilerplate/issues/2952
const firebaseConfig = {
     apiKey: "",
     authDomain: "",
     projectId: "",
     storageBucket: "",
     messagingSenderId: "",
     appId: "",
     measurementId: ""
};

firebase.initializeApp(firebaseConfig);

const messaging = firebase.messaging();

messaging.onBackgroundMessage(function (payload) {
     console.log('sw bg message event: ', payload);
});

// O evento onMessage pertence ao contexto de Windows e não do service worker
//https://stackoverflow.com/questions/42964547/uncaught-firebaseerror-messaging-this-method-is-available-i
n-a-window-context
```

App.tsx

```
import { useEffect } from 'react';
import logo from './logo.svg';
import './App.css';
import firebase from 'firebase/app';
import 'firebase/messaging';

function App() {

  useEffect(() => {

    if ("serviceWorker" in navigator) {
      navigator.serviceWorker
        .register("./firebase-messaging-sw.js")
        .then(function (registration) {
          console.log("Registration successful, scope is:", registration.scope);
        })
        .catch(function (err) {
          console.log("Service worker registration failed, error:", err);
        });
    }

    // https://github.com/react-boilerplate/react-boilerplate/issues/2952
    const firebaseConfig = {
      apiKey: "",
      authDomain: "",
      projectId: "",
      storageBucket: "",
      messagingSenderId: "",
      appId: "",
      measurementId: ""
    };

    firebase.initializeApp(firebaseConfig);

    firebase.messaging().onMessage(function(payload) {
      console.log("onMessage event ", payload);
    });

    navigator.serviceWorker.addEventListener("message", (message) => {
      console.log("message event", message);
    });

  }, []);

  const getMessaging = () => firebase.messaging();

  // Send the registration token your application server, so that it can:
  // - send messages back to this app
  // - subscribe/unsubscribe the token from topics
  const sendTokenToServer = (currentToken: any) => {
    if (!isTokenSentToServer()) {
      console.log('Sending token to server...');
      // TODO(developer): Send the current token to your server.
      setTokenSentToServer(true);
    } else {
      console.log('Token already sent to server so won\'t send it again unless it changes');
    }
  }

  const isTokenSentToServer = () => {
    return window.localStorage.getItem('sentToServer') === '1';
  }

  const setTokenSentToServer = (sent: boolean) => {
    window.localStorage.setItem('sentToServer', sent ? '1' : '0');
  }

  const requestPermission = () => {
    console.log('Requesting permission...');
    Notification.requestPermission().then((permission) => {
      if (permission === 'granted') {
        console.log('Notification permission granted.');
        // TODO(developer): Retrieve a registration token for use with FCM.
      } else {
        console.log('Unable to get permission to notify.');
      }
    });
  }

  const getToken = () => {
    // 1st time: network call, next: cache
    getMessaging().getToken({ vapidKey: '' }).then((currentToken) => {
      if (currentToken) {
        console.log("TOKEN: ", currentToken);
        sendTokenToServer(currentToken);
      }
      else {
        console.log('No registration token available. Request permission to generate one.');
        setTokenSentToServer(false);
      }
    }).catch((err) => {
      console.log('An error occurred while retrieving token. ', err);
      setTokenSentToServer(false);
    });
  }

  const deleteToken = () => {
    getMessaging().deleteToken().then(() => {
      console.log('Token deleted.');
      setTokenSentToServer(false);
    }).catch((err: any) => {
      console.log('Unable to delete token. ', err);
    });
  }
```

FCMService.java

```
import java.io.IOException;
import java.util.concurrent.ExecutionException;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;

import com.devsuperior.pocfcm.dto.PushNotificationRequestDTO;
import com.google.auth.oauth2.GoogleCredentials;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseOptions;
import com.google.firebase.messaging.FirebaseMessaging;
import com.google.firebase.messaging.Message;
import com.google.firebase.messaging.Notification;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

@Service
public class FCMService {

        private Logger logger = LoggerFactory.getLogger(FCMService.class);

        @PostConstruct
        public void initialize() throws IOException {
                FirebaseOptions options =
FirebaseOptions.builder().setCredentials(GoogleCredentials.getApplicationDefault()).build();
                FirebaseApp.initializeApp(options);
        }

        public void sendMessageToToken(PushNotificationRequestDTO request) throws InterruptedException,
ExecutionException {
                Message message = getPreconfiguredMessageToToken(request);
                Gson gson = new GsonBuilder().setPrettyPrinting().create();
                String jsonOutput = gson.toJson(message);
                String response = sendAndGetResponse(message);
                logger.info("Sent message to token. Device token: " + request.getToken() + ", " + response
+ " msg " + jsonOutput);
        }

        private String sendAndGetResponse(Message message) throws InterruptedException, ExecutionException
{
                return FirebaseMessaging.getInstance().sendAsync(message).get();
        }

        private Message getPreconfiguredMessageToToken(PushNotificationRequestDTO request) {
                return getPreconfiguredMessageBuilder(request).setToken(request.getToken()).build();
        }

        private Message.Builder getPreconfiguredMessageBuilder(PushNotificationRequestDTO request) {
                Notification notification =
Notification.builder().setTitle(request.getTitle()).setBody(request.getMessage()).build();
                return Message.builder().setNotification(notification);
        }
}
```