SGN-35016 INTERNET OF THINGS AND MEDIA SERVICES PROJECT WORK - SNEAKSPEAK

Iiro Nykänen, Ville Vaarala

< student numbers >
< email addresses >
Tampere University of Technology

ABSTRACT

SneakSpeak is a self hosted group chat application. It conquers the market leaders Slack and Flowdock by giving organizations and teams ability to not to give 3rd parties access to the communication. The proof-of-concept consists of a Node.js GCM HTTP Connection Server [1] and an Android client. The server can be hosted in a local network e.g. on a Raspberry Pi computer.

1. INTRODUCTION

This document describes a simple messaging application proof-of-concept created as a project work for the course *Internet of things and media services*. The application consists of a Node.js server and an Android client.

The project originated from the realisation that nowadays different mobile instant messaging applications are very popular. Most of the people that own a modern smart phone have WhatsApp installed, the more critical users may prefer Telegram, and software developers are starting to migrate towards Slack. The thing these services have in common is that they provide the software as a service. All the messaging and information goes through their own servers, which may or may not be secure enough for your use. This project provides an alternative to these premade services. Since third party servers are deemed untrustworthy, the solution is for users to host their own private servers.

From this basic concept the project started to take shape. The first iterations of the core idea revolved around the messaging protocol, like if it were possible to use XMPP and websockets or if a server that only handled handshakes between clients would be a good idea. Eventually the project steered towards a push notification system, which is a quite popular way to implement instant messaging nowadays.

2.3. Right now our project is in a kind of ”research via implementation“ phase, in which we experiment how the push notification system and the cloud integration works. So far we have managed to implement initial messaging between client and server, but some things are not clear to us yet.

2. METHODS

This section gives a quick glance at the technologies used in the application.

2.1. Android client

2.2. Node.js server

2.3. Google Cloud Integration

3. RESULTS

Too early to say.

4. CONCLUSIONS

Something something success story.

REFERENCES

[1] Google, *Implementing an HTTP Connection Server*, 2016, accessed 1.3.2016. [Online]. Available at: https://developers.google.com/cloud-messaging/http.