Dokumentation Proof of Concept #7

Das Ziel des Proof of Concept ist es, eine Nachricht über einen Android Client in ein bestimmtes Topic zu Publizieren und diese Nachricht dann auf einem Server durch abonnieren dieses bestimmten Topic zu empfangen.

Zur Durchführung dieses Proof of Concept muss zusätzlich zudem schon beschriebenen Android Studio und der Android SDK auch noch die PubNub Android SDK 3.7.5 installiert werden. Diese gibt es unter https://www.pubnub.com/docs/android-java/pubnub-java-sdk zum Download.

Um den PubNub Dienst unter gewissen Einschränkungen Kostenlos nutzen zu können ist es Notwendig sich dort kostenlos zu Registrieren. Nach erfolgreicher Registrierung und erhalt der Account gebundenen Publish und Subscribe Key's kann mit der Implementierung begonnen werden.

Nach erstellen eines Projektes in Android Studio mit einer leeren Actitvity und Einbinden der PubNub Android SDK wird mit der Einrichtung des Layout der App begonnen.

```
<Button android:id="@+id/PubBut"
    android:text="I will publish a message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />
```

Wichtig hierbei ist das Vergeben der Button ID, diese wird später noch benötigt. Im nächsten Schritt wird die MainActivity.java implementiert. Als erstes müssen hier die benötigten Bibliotheksdateien eingebunden werden.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import com.pubnub.api.*;
```

Zum Initialisieren eines PubNub Objektes müssen der Account eigene Publish und Subscribe Key angegeben werden. Der Letzte Parameter gibt an ob die Daten Verschlüsselt über SSL oder unverschlüsselt übertragen werden sollen.

```
Pubnub pubnub = new Pubnub("Publish_Key", "Subscribe_Key", false);
```

Dem eben erstellten Button wird ein OnClickListener hinzugefügt, der bei einem Klick auf den Button eine Nachricht in den Channel "MyChannel" publiziert. Diese Nachricht wird nun an alle Subscriber dieses Channel gesendet. Das Publizieren der Nachricht passiert über den Methodenaufruf pubnub.publish().

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Callback callback = new Callback() {
            public void successCallback(String channel, Object response) {
                System.out.println(response.toString());
            }
            public void errorCallback(String channel, PubnubError error) {
                System.out.println(error.toString());
            }
        };
        pubnub.publish("MyChannel", "Diese Nachricht kommt vom AndroidClient" , callback);


    }
});
```

Wie im Proof of Concept beschrieben ist der Subscriber der Server. Dafür wird ein Node Server mit den folgenden Modulen gestartet. Diese müssen vorher über npm install installiert werden.

```
global.express = require('express');

var http = require('http');
var pubnub = require('pubnub');
var app = express();
var server = http.createServer(app);
```

Auch auf dem Node Server muss ein PubNub Object initialisiert werden. Dies geschiet ebenfalls über Angabe der eigenen Publish und Subscribe Key's.

```
//Initialisiren
var PUBNUB_demo = pubnub.init({
    publish_key: 'Publish_Key',
    subscribe_key: 'Subscribe_Key'
});
```

Der Server muss jetzt den Channel "MyChannel" abonnieren, um die Nachricht die von der Android App aus publiziert wird zu erhalten. Alle erhaltenen Nachrichten werden auf der Konsole ausgegeben, um den erhalt zu Protokollieren.

```
PUBNUB_demo.subscribe({
        channel: 'MyChannel',
        message: function(m){console.log(m)},
});
```

Die App wird nun entweder über den Emulator von Android Studios oder über ein externes reales Android Gerät gestartet. Um die Nachrichten auch empfangen zu können wird ebenfalls der Server gestartet.
Durch Klick auf den Button in der Android App wird eine Nachricht publiziert, der Server erhält diese Nachricht unmittelbar danach und gibt diese auf der Konsole aus. Zusätzlich zu dem im Proof of Concept beschrieben Funktionsumfang ist der Android Client auch Abonnent von "MyChannel". Die Nachrichten die der Android Client erhält werden ebenfalls über die Konsole ausgegeben.

Die Ausgabe vom Node Server sieht folgendermaßen aus:

```
Manuels-MBP:PubNubTut manuelmichels$ node server.js
Server listens on Port 3000.
Diese Nachricht kommt vom AndroidClient
```

Das Interface der Android App:

Die Ausgabe des Android Client:

```
11-06 17:05:38.530 2381-2401/? I/System.out: SUBSCRIBE : MyChannel : class java.lang.String : Diese Nachricht kommt vom AndroidClient
```