Implementation eines Notification-Service auf einem Android-Client

Es soll ein Notification-Service auf dem Android-Client implementiert werden, wodurch neue Nachrichten eines abonnierten Topics von dem RabbitMQ-Server als Benachrichtigung in der Android Statusleiste angezeigt werden. Der Service muss auch dann weiter laufen, wenn die Applikation bereits geschlossen ist.

Da die Funktion eines Publish-Subscribe-Systems mit RabbitMQ bereits durch ein POC erfolgreich getestet wurde, kann diese Technologie in diesem "Proof-of-Concepts" verwendet werden (siehe Artefakt "POC Publish-Subscribe mit RabbitMQ").

Die Notifications werden daher von neuen Nachrichten des RabbitMQ-Servers auf ein Topic angestoßen und der Inhalt der Notification soll der Text aus der neuen Nachricht des Topics sein.

Bezug zum Projekt

Das Publish-Subscribe-System mit RabbitMQ wurde erfolgreich getestet und Nachrichten können von unseren Clients untereinander versendet und empfangen werden. Es ist nun wichtig, dass die Benutzer benachrichtigt werden, sobald eine Nachricht empfangen wurde. Dies ist besonders für das Pflegepersonal wichtig, da die Erinnerung einer Verabreichung eines Medikaments durch solche Benachrichtigungen umgesetzt werden sollen. Auch der Patient soll Benachrichtigungen zu seiner Medikation erhalten. Ohne eine signifikante Benachrichtigung des Clients über neue Nachrichten eines Topics sind unsere Alleinstellungsmerkmale gefährdet.

Exit-Kriterium

Das "Proof-of-Concepts" gilt als erfolgreich abgeschlossen, wenn eine Benachrichtigung auf dem Android-Client auf der Statusleiste angezeigt wird, die durch eine Nachricht auf ein abonniertes Topic über RabbitMQ angestoßen wird. Benachrichtigungen sollen auch wenn die Applikation geschlossen ist, weiterhin angezeigt werden.

Fail-Kriterium

Sollten die Benachrichtigungen, während die Android-Applikation läuft oder geschlossen ist, nicht auf der Statusleiste des Android-Clients angezeigt werden,

wenn eine neue Nachricht auf ein abonniertes Topic empfangen wird, so gilt das "Proof-of-Concepts" als gescheitert.

Fallback

Sollte die Implementation des oben beschriebenen "Proof-of-Concepts" scheitern, so muss sichergestellt werden, dass in der Applikation intern ohne den Androidsystemservice zu nutzen, ein Benachrichtigungssystem implementiert wird, wodurch der Nutzer über Neuigkeiten informiert werden. Eine weitere Möglichkeit, die in Betracht gezogen werden kann, ist die Benachrichtigung per SMS oder E-Mail.

Prozessdarlegung

Da die benötigten Programme und Libraries bereits installiert wurden, kann sofort mit der Entwicklung begonnen werden. Für die Implementation des oben beschriebenen "Proof-of-Concepts" sind drei Komponenten notwendig:

1. Publish-Subscribe-System mit RabbitMQ
2. Notification
3. Android-Service

Das Publish-Subscribe-System wurde bereits erfolgreich getestet und funktioniert. Daher kann die aus dem POC entstandene Entwicklung an dieser Stelle als Startpunkt weiter verwendet werden.

Zunächst soll eine simple Notification erstellt werden, die aus der Activity heraus erzeugt und getriggert wird. Die Android-SDK Dokumentation bietet uns für die Programmierung hierfür eine geeignete Hilfestellung mit Beispielen[1]. Die Notification konnte erstellt und in der Statusleiste angezeigt werden.

Damit die Notification nun durch neue Nachrichten auf den Topic "/verordung" getriggert werden, muss die Notification im Code in dem Listener des RabbitMQ-Consumers aufgebaut werden. Der Listener wird dann aufgerufen, wenn Nachrichten auf den Topic "/verordnung" empfangen werden. Der Textinhalt der Nachricht aus dem Topic kann in der Notification mit einer Methode übergeben werden. Der neue Code konnte ausgeführt werden, und die Notification wurden, wie gewünscht angezeigt.

[1] http://developer.android.com/guide/topics/ui/notifiers/notifications.html Sichtung: 03.11.2015, 18:25

Sobald die Applikation geschloßen wird, werden die Notifications nicht mehr angezeigt, obwohl neue Nachrichten auf dem Topic "/verordnung" empfangen werden. Der Grund dafür liegt im Lebenszyklus einer Activity[2]. Die onDestroy-Methode wird aufgerufen, wenn die Applikation geschlossen wird. Dadurch werden die Prozesse innerhalb der Activity ebenfalls beendet. Ein Service jedoch hat einen anderen Lebenszyklus[3]. Bei der Betrachtung des Lebenszyklus eines Services, wird ersichtlich, dass ein Service weiterhin aktiv ist, auch wenn die Activity "zerstört" wurde. Ein Service kann also benutzt werden, um die Verbindung zu RabbitMQ aufrecht zu erhalten und den "Nachrichten-Listener" weiter laufen zu lassen, wenn die Applikation geschlossen wird.
Der Service konnte nach dem Lesen der Android-SDK-Dokumentation zu Services[4] implementiert und gestartet werden. Der bereits vorhandene Code zu der Notification, der Verbindung des RabbitMQ-"Consumers" zum RabbitMQ-Servers, die Erstellung und Anbindung an den Exchange "amq.topic" der Queue und dem Listener, der auf Nachrichten auf dem Topic "/verordnung" wartet, konnte nun in die onCreate-Methode des Services verschoben werden. Der Service kann dann in der onCreate-Methode der Activity gestartet werden. Der Service läuft solange, bis seine onDestroy-Methode aufgerufen wird.

Da der Node-Server bereits funktiontüchtig ist und eine Verbindung zu RabbitMQ aufbauen und Nachrichten auf den Topic "/verordnung" senden kann, ist ein Test des "Proof-of-Concepts" nach den Exit-Kriterien nun der nächste Schritt.

Status

Der Android-Client wurde erfolgreich nach den Exit-Kriterien getestet. Der "Proof-of-Concept" ist demnach erfolgreich abgeschlossen und die getestete Technologie kann für die weitere Entwicklung genutzt werden.

Ausblick

Die Notifications funktionieren wie gewünscht. Allerdings sind einige Anpassungen nötig, die aus den Anforderungen abgeleitet werden. Anpassungen der Benachrichtigungen könnten im Inhalt und Layout der Benachrichtigung durchgeführt werden. Ebenso sollte die Laufzeit des Service weiter betrachtet werden. Es ist noch unklar, welche Faktoren den Lebenszyklus des Service beeinflußen. Die Faktoren sollten im weiteren Verlauf der Entwicklung

[2] https://developer-blog.net/programmieren/android-tutorial-teil-4/ Sichtung: 03.11.2015, 20:35
[3] http://developer.android.com/images/service_lifecycle.png Sichtung:03.11.2015, 20:40
[4] http://developer.android.com/guide/components/services.html Sichtung: 03.11.2015 19:30

recherchiert und erkannt werden. Der Code sollten dann dahingehend geändert werden, dass eine ungewollte Beendigung des Service verhindert wird. Sollte dies nicht beachtet werden, kann der Service, ohne dass die Benutzer es merken, beendet werden und zu einem weiteren Risiko führen.

Anhang:

Installationen:

1. Erlang
2. RabbitMQ-Server
3. Node-JS
    a. express
    b. amqp
    c. bodyparser
4. Eclipse
    a. ADT-Plugin
5. Android SDK herunterladen
6. RabbitMQ Java Client herunterladen