**Implementierung der App**

Wir haben die native Android-App von Openhab benutzt, welche im Google Play Store runtergeladen werden kann.



Das Layout der App wird durch die Sitemap bestimmt, welche sich in den Konfigurationsdateien von Openhab befindet:

sitemap Tuer label="Hauptmenue"

{

Frame{

Image url="http://localhost:8080/cameras/image" refresh=5000

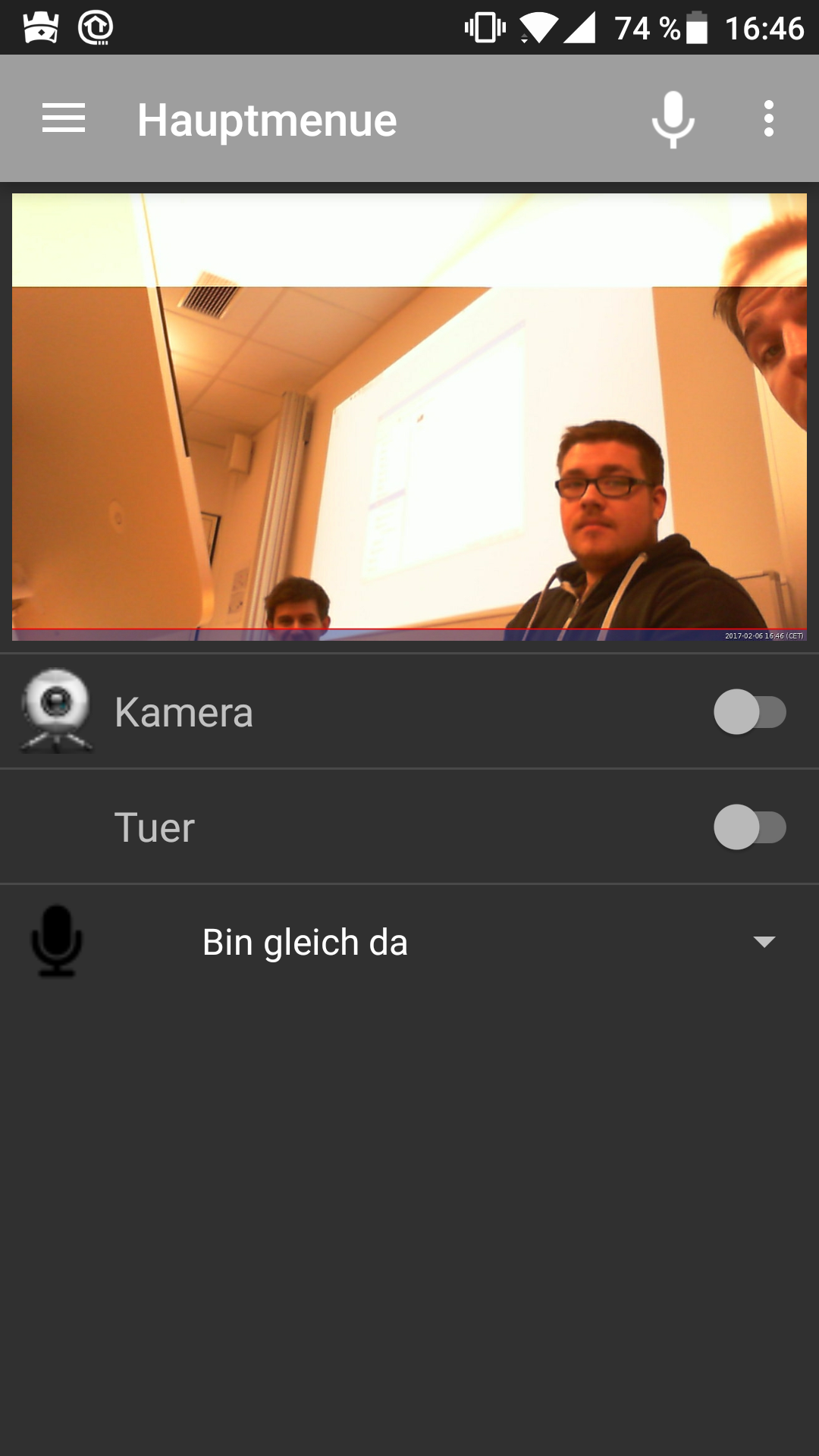
Switch item=camSwitch label="Kamera" icon="camera"

Switch item=tuerSwitch label="Tuer" icon="door"

Selection item=sagenAuswahl mappings=[ 0="Bin gleich da", 1="Bin nicht zu Hause", 2="Bin in einer Stunde da", 3="Stellen Sie es in der Garage ab" ]

}

}



Die in der Sitemap verwendeten Items werden in einer extra Datei definiert, damit diese auch in den später erläuterten Rules angesprochen werden können:

Switch camSwitch "Kamera" <camera>  
 Switch tuerSwitch "Tuer" <door>  
 Switch raspiLED { gpio="pin:17" } //LED simuliert Tuer  
 Contact taster "Taster"{ gpio="pin:22"}

Switch LED2 {gpio="pin:27"} //LED simuliert Klingelton

Number sagenAuswahl "" <micro>

// Syntax: itemtype itemname [„labeltext“] [<iconname>] [{bindingconfig}]

In der Rules-Datei wird die Logik der App definiert. Eine Regel wird ausgeführt, sobald die entsprechende Trigger Condition erfüllt ist:

import org.openhab.script.actions.\* //Import fuer die Push-Notifications zum Smartphone  
  
 rule klingeln

when

Item taster changed to OPEN

then

sendBroadcastNotification("Ding Dong!") // Push-Notification an das Smartphone

sendCommand(camSwitch,ON) // rule „KameraAN“ wird aktiviert

sendCommand(LED2,ON) // Klingelton wird simuliert

Thread::sleep(2000)

sendCommand(LED2,OFF)

end

rule KameraAN

when

Item camSwitch received command ON

then

// Datei „kameraAn“ wird zur Kommunikation mit der Kamera-Steuerung erzeugt

executeCommandLine("touch /opt/openhab/webapps/cameras/kameraAn")

end

rule KameraAUS

when

Item camSwitch received command OFF

then

// Datei „kameraAn“ wird wieder gelöscht

executeCommandLine("rm /opt/openhab/webapps/cameras/kameraAn")

end

rule tuerOeffnen

when

Item tuerSwitch received command ON

then

sendCommand(raspiLED, ON) // Tueroeffnung wird simuliert

end

rule tuerSchliessen

when

Item tuerSwitch received command OFF

then

sendCommand(raspiLED, OFF) // Tuerschliessung wird simuliert

end

rule auswahlBox

when

Item sagenAuswahl received command

then

logInfo("auswahlBox", "Wir sind vor switch case ")

switch(receivedCommand)

{

case 0: executeCommandLine( " /opt/openhab/configurations/scripts/speech.sh Bin gleich da")

case 1: executeCommandLine( " /opt/openhab/configurations/scripts/speech.sh Bin nicht zu Hause")

case 2: executeCommandLine( " /opt/openhab/configurations/scripts/speech.sh Bin in einer Stunde da")

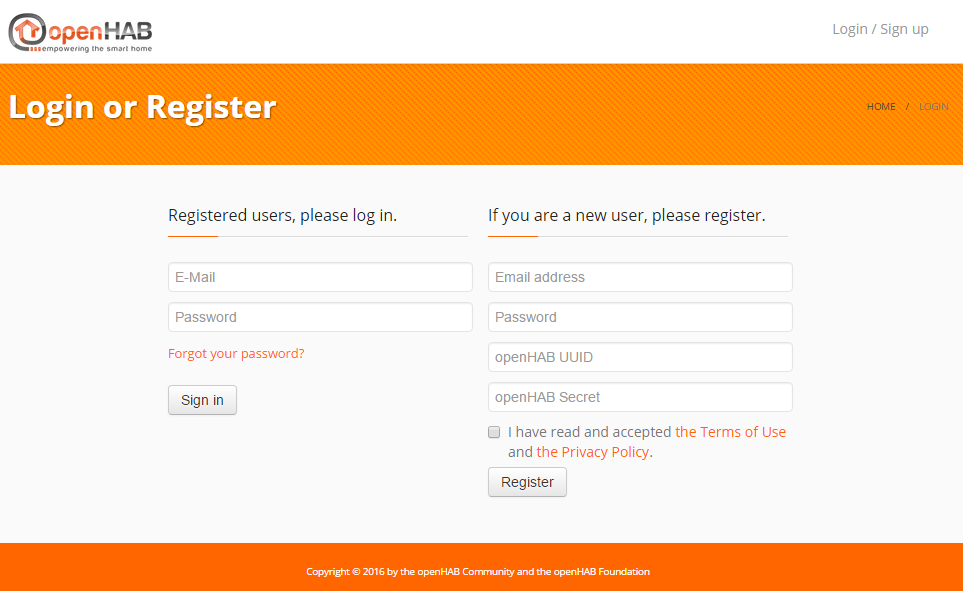
case 3: executeCommandLine( " /opt/openhab/configurations/scripts/speech.sh Stellen sie es in der Garage ab")

}

end

Zur Kommunikation zwischen der App und dem Openhab-Server haben wir den Service myopenhab.org benutzt. Dafür mussten wir das entsprechende Binding im Addons-Ordner von Openhab installieren.

Die Benutzer der App müssen sich anschließend bei myopenhab.org registrieren, um sich dann anschließend bei der App authentifizieren zu können:



Die UUID wird vom Openhab-Server und das Secret vom myopenhab-Binding jeweils lokal erzeugt, sodass eine eindeutige Authentifizierung zum Openhab-Server möglich ist.

Über dieses erstellte Benutzerkonto können dann noch weitere Benutzer hinzugefügt werden, sodass nicht jedes mal neu die UUID und das Secret in der Openhab-Konfiguration ausfindig gemacht werden müssen.

Über dieses Konto können sich dann die Benutzer in der App authentifizieren. Dazu sind dann folgende Einstellungen nötig:

