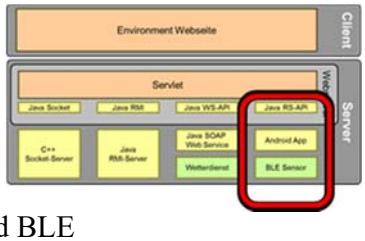
Verteilte Informationssysteme

Person:

Name:

Matrikelnummer:



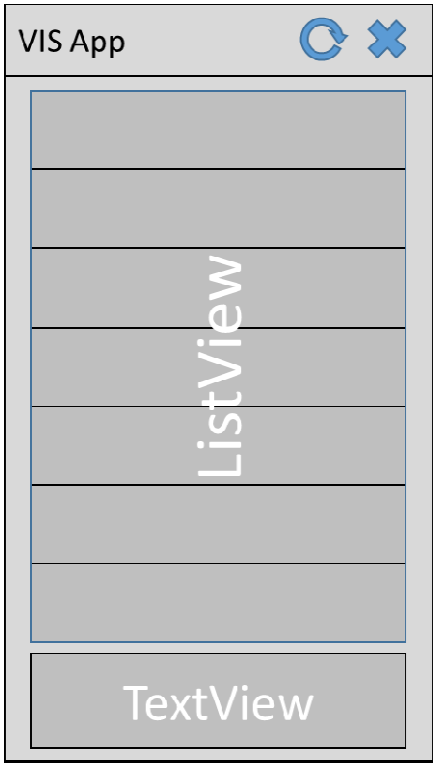
Übungszettel 4

Web Services – Verwendung eines REST Web Services
(Datenextraktion und -bereitstellung)
Android Foreground Services, Messenger-based Communication und BLE

Aufgabe 1a

„M“ **(15%)**

Erstellen Sie eine **`Activity`**, die über folgende einfache Struktur und über zwei Optionen verfügt:



Sie enthält ein **`ListView`** Widget in dem später die verfügbaren BLE Geräte angezeigt werden und ein **`TextView`** Widget in dem später die verfügbaren Sensordaten visualisiert werden.

Als Optionen stehen „Service starten“ und „Service stoppen“ zur Verfügung

Erstellen Sie zunächst einen entsprechenden Adapter, der die Liste mit Dummy-Daten füllt und fangen Sie die Auswahl eines Eintrages in ihrem Code ab.

Rufen Sie bei der Auswahl der beiden Optionen jeweils die dazugehörige **`startService()`** und **`stopService()`** Methode auf und versehen sie alle notwendigen Methoden mit entsprechenden Log-Ausgaben.

Überprüfen Sie in der **`startService()`**-Methode das Vorhandensein der entsprechenden Permissions (**`BLUETOOTH_ADMIN`**, **`ACCESS_FINE_LOCATION`** und **`INTERNET`**) und initiieren Sie bei Bedarf die Aktivierung von Bluetooth und des Location-Dienstes über die entsprechenden Intents.

Aufgabe 1b „M“ (15%)

Erstellen Sie einen Service (**`BLEService`**), der als Foreground-Service zu starten ist und versehen Sie alle relevanten Methoden mit **`Log.i()`** Aufrufen. Die hierfür notwendige Notification des Foreground-Services „zeigt“ dabei auf die Activity des Projektes. Starten und Stoppen Sie den Service über die entsprechenden Activity Optionen. Testen Sie die Lauffähigkeit des Services auch ohne aktive Activity und starten Sie die Activity über die Notification. Achten Sie dabei darauf, dass sich immer nur eine Instanz der Activity auf dem BackStack befindet.

Stellen Sie sowohl in der Aktivität als auch in dem Service jeweils zwei Messenger Referenzen und eine Handler Instanz zur Verfügung (**`mActivityMessenger`**, **`mServiceManager`** und **`mHandler`**).

Erstellen Sie in der **`onCreate()`**-Methode der Aktivität die passenden Objekte für **`mHandler`** und **`mActivityMessenger`** und in der **`onCreate()`**-Methode des Services die passenden Objekte für **`mHandler`** und **`mServiceMessenger`**. Übermitteln Sie die Referenz auf den **`mActivityMessenger`** beim Start des Services über ein entsprechendes Extra-Feld des Intents an den Service. Dieser packt die Referenz aus und verknüpft sie seiner **`mActivityMessenger`** Referenz. Sobald der Service die Activity-Messenger-Referenz erhalten hat, schickt er der Activity über den erhaltenen Messenger eine **`UPDATE_ACTIVITY_MESSENGER`** Message und übermittelt dabei mittels der **`replyTo`**-Variable der Message seinerseits eine Referenz auf seinen **`mServiceMessenger`**. Der Erhalt des Service-Messengers wird wiederum von der Activity, durch eine **`UPDATE_SERVICE_MESSENGER Message`** an den **`Service`**, quittiert. Versehen sie die **`handleMessage()`** Methoden der beteiligten Handler mit entsprechenden Log-Ausgaben.

Berücksichtigen Sie den Fall, dass der Service bereits im Hintergrund läuft, die Activity zunächst aber beendet wurde und in weiterer Folge über die Notification erneut gestartet wird. Um eine Kommunikation über die Messenger zu gewährleisten, muss die Activity erkennen, dass der Service bereits aktiv ist. In diesem Fall initiiert die Actitity den erneuten „Start“ (der seitens des Service bis auf übermittelten Metadaten „ignoriert“ wird) des Services und damit den Austausch der aktuellen Messenger-Referenzen. Lösen Sie das Problem durch den Einsatz von SharedPreferences und einem darin enthaltenen booleschen Flag **`SERVICE_ALIVE`**.

Aufgabe 1c „M“ (15%)

Erweitern Sie den Service um die notwendigen BLE-Aufrufe und Support-Funktionen:

1. Extrahieren Sie in der **`onStartCommand()`**-Methode zunächst den BluetoothManager, über den Manager extrahieren Sie sich eine Instanz des BluetoothAdapters. Über den Adapter bekommen Sie Zugriff auf den BluetoothLeScanner, dessen Referenz sie als globale Variable im Service verwalten.
2. Erstellen Sie eine anonyme Implementierung des ScanCallback-Klasse (Bluetooth), wobei Sie in der zur Verfügung zu stellenden Methode **`onScanResult()`** eine (global referenzierte) Liste von ScanResult Objekten mit den Ergebnissen aus **`onScanResult()`** füllen. Achten Sie dabei darauf, dass einige Geräte während des Scannens mehrfach gefunden werden können und nur einmal in der Liste enthalten sein sollten.
3. Erstellen Sie im Service die beiden Methoden **`startScan()`** und **`stopScan()`** und rufen Sie die **`startScan()`** am Ende der **`onStartCommand()`**-Methode auf.

4. Benutzen Sie in der **`startScan()`**-Methode die **`startScan()`**-Methode der global bekannten Variablen des BluetoothLeScanners, eine Verwendung von Filtern ist im Rahmen der Übung nicht notwendig. Bevor Sie den Scan-Vorgang starten, sollten Sie mittels des globalen Handlers **`mHandler`** und dessen **`postDelayed()`**-Methode den Aufruf der **`stopScan()`**-Methode, um damit den automatischen Abbruch des energiehungrigen Scannens nach Geräten zu initiieren.
5. Rufen Sie in der **`stopScan()`**-Methode die, im Service zu implementierende, Methode **`sendBLEDeviceList()`** auf. Diese organisiert die Übergabe der Liste der gefundenen BLE-Geräte an die Activity, durch den Einsatz des Messengers **`mActivityMessenger`**. Da die Klasse ScanResult nicht serialisierbar ist, müssen – Sie eine eigene Daten-Container Klasse zur Verfügung stellen, die die benötigten Informationen enthält. Implementieren Sie hierfür die serialisierbare POJO-Klasse **`BLEDeviceDescription`**, die neben dem Namen des gefunden Gerätes, dessen MAC-Adresse und die RSSI enthält und übertragen Sie die Informationen aus der ScanResult-Liste in die **`BLEDeviceDescription`**-Liste.

Aufgabe 1d „M“ (15%)

Packen Sie auf der Activity-Seite die BLEDeviceDescription-Liste aus und füllen Sie den List-Adapter der Activity mit den erhaltenen Informationen. Wenn die Anwender ein Gerät auswählen, übermitteln Sie mittels des Messengers die MAC-Adresse des ausgewählten Gerätes zurück an den Service, der daraufhin die Kontaktaufnahme mit dem Gerät startet.

Aufgabe 1e „M“ (15%)

Ermitteln Sie auf der Service Seite, mittels der erhaltenen MAC-Adresse und der ScanResult-Liste, das zur Adresse passende Gerät (BluetoothDevice) und rufen Sie dessen **`connectGatt()`**-Methode auf.

Hierfür wird eine BluetoothGattCallback-Implementierung benötigt. Stellen Sie hierfür ein globales Objekt mit einer entsprechenden anonymen Implementierung zur Verfügung. Es werden im Rahmen der Übung nur die Methoden **`onConnectionStateChange()`**, **`onServicesDiscovered()`** und **`onCharacteristicChanged()`** benötigt.

In **`onConnectionStateChange()`** wird der Status einer global bekannt zu machenden Referenz zu einem BluetoothGatt-Objekt organisiert (wird der Service zwischenzeitlich geschlossen, so muss die entsprechende **`close()`**-Methode des BluetoothGatt-Gerätes aufgerufen werden) und im Zustand **`STATE_CONNECTED`** die Methode **`discoverServices()`** aufgerufen und somit die Suche nach BLE-Services gestartet. Suchen Sie zunächst nach folgendem Service mit der UUID: **`0000180d-0000-1000-8000-00805f9b34fb`**

In **`onServicesDiscovered()`** wird im Rahmen der Übung nur der Verbindungsaufbau zum „Heart Rate Service“ mittels der BluetoothGatt-Referenz initiiert. Wurde der Service gefunden, so muss als nächstes die passende Characteristic aus dem Service extrahiert werden. Verwenden Sie hierfür die folgende UUID: **`00002a37-0000-1000-8000-00805f9b34fb`**. Im Anschluss daran, muss zunächst mittels der BlueetoothGatt-Referenz und dessen Methode **`setCharacteristicNotification()`** und in weiterer Folge mittels des „Client Characteristic Configuration Discriptors“ mit der UUID: **`00002902-0000-1000-8000-00805f9b34fb`** der “Notification”-Modus initiiert werden.

Hat die Initiierung des „Notification“-Modus geklappt, wird in weiterer Folge die **`onCharacteristicChanged()`** Methode zyklisch aufgerufen und es kann die aktuelle Herzfrequenz aus der Characteristic extrahiert werden. Sie wird als 8 Bit unsigned int geliefert.

Aufgabe 1f „M“ (15%)

Stehen die Herzfrequenz-Daten zur Verfügung, gilt es als nächstes die Daten sowohl an den Web Service als auch an die Activity zu übermitteln.

Stellen Sie für die Visualisierung in der Activity im Service die Methode **`visualizeData()`** zur Verfügung, die Mittels des **`mActivityMessenger`**-Objektes die Daten an die Activity übermittelt. Die Activity ihrerseits empfängt die Daten in der **`handleMessage()`**-Methode des Handlers und visualisiert die Werte in dem entsprechenden Widget im Layout.

Stellen Sie für die Übermittlung an den Web Service die Methode **`communicateData()`** zur Verfügung. Lagern Sie dabei den Kommunikationsanteil in einen Thread aus. Verwenden Sie hierfür das Callable-Interface in Kombination mit einem „SingleThread“-ExecutorService. Quittieren Sie seitens des Web Services den Empfang der Daten über eine entsprechende Ausgabe.

Aufgabe 2 „M“ (10%)

Binden Sie analog zu den C++, RMI und JAX-WS „Environment Data“-Servern, den REST-basierten JAX-RS „Environment Data“ Server in das Übersichts-Servlet ein. Der REST Web Service stellt dabei die vom Android Client ermittelten BLE-Sensor-Daten zur Verfügung.

Ergänzen Sie hierfür die bisher verwendete REST-API um folgende „Methode“:

`<Servlet-Context>/Values`
→ **`POST-basierte URL für die Übermittlung der Sensordaten von der AndroidApp zum REST-Service`**