Proof of Concept #3 – Implementierung einer Android App und Prüfung der Client-Server Kommunikation über HTTP

Dieser PoC adressiert das Risiko, dass eine Implementierung mobiler Clients für die Android Plattform mangels Erfahrung des Projektteams scheitern könnte. Der enge Zeitrahmen der Veranstaltung erlaubt keine ausschweifenden Einarbeitungszeiten. Ziel des PoCs ist der Nachweis der Fähigkeit zur ausreichend schnellen Einarbeitung in das Themenfeld Programmierung verteilter Anwendungen mit Java. Zudem ist die Kommunikation zweier Systemkomponenten (Android App und Node.js Server) über HTTP zu prüfen.

Dieser Proof of Concept wurde zeitlich nach PoC #11 – Integration eines externen Mapping Dienstes durchgeführt. Zunächst wurde ein Node.js Server aufgesetzt, der auf Port 3000 unter der localhost-Adresse lauscht. Sein einziger Endpunkt definiert ein POST auf der Default-Route (http://localhost:3000), loggt Anfragen auf die Konsole und Antwortet mit dem erhaltenen JSON-body und einem HTTP-Statuscode 201.

```
1  global.express = require('express');
2  var http = require('http');
3  var app = express();
4  var server = http.createServer(app);
5  var bodyParser = require('body-parser');
6
7  app.post('/', function(req, res) {
8      console.log("Standort erhalten !");
9      res.status(201).json(req.body).end();
10 });
11
12 app.listen(3000, function() {
13     console.log("Server listens on Port 3000.");
14 });
```

Die benötigten Module Express sowie der Body Parser wurden in einer package.json festgehalten.

Um den Umgang mit HTTP Requests in der Android App komfortabler zu gestalten wird die Android-Volley Libary verwendet. Diese verkapselt die Funktionalitäten der HttpUrlConnection-Klasse und erlaubt zudem den recht übersichtlichen Umgang mit Parsing von JSON Payload.

Um diese Libary verwenden zu können wurde das Git-Repository

https://android.googlesource.com/platform/frameworks/volley/

zunächst geklont und in der Android Studio DIE mittels File>New>Import Module eingebunden.

Der im zuvor durchgeführten PoC ermittelte Standort wird nun in eine JSON Darstellung überführt und per HTTP-Post an den Server übermittelt. Ist der Post erfolgreich wird die Antwort in einem Textview-Element auf dem Android Gerät angezeigt.

Die Postmethode wurde wie folgt implementiert :

```
public void postLocation() {

    // Requestqueue anlegen
    RequestQueue queue = Volley.newRequestQueue(this);

    //Request Body, später in JSON umgewandelt
    HashMap<String, String> params = new HashMap<String, String>();

    //Fülle JSON Parameter für den Request
    if (letzterStandort != null) {
       params.put("laengengrad", String.valueOf(letzterStandort.getLongitude()));
       params.put("breitengrad", String.valueOf(letzterStandort.getLatitude()));
    }

    //localhost:3000 funktioniert nur bei Debugging mit Chromes Port-Forwarding ,
    //Im Emulator muss die Loopback-Adresse 10.0.2.2:3000 verwendet werden
    JsonObjectRequest myRequest = new JsonObjectRequest(
          Request.Method.POST,
          "http://localhost:3000",
          new JSONObject(params),

          new Response.Listener<JSONObject>() {
             @Override
             public void onResponse(JSONObject response) {

                try {
                   tvPost.setText(response.toString());
                 } catch (JSONException e) {
                   e.printStackTrace();
                }
             }
          },
          new Response.ErrorListener() {
             @Override
             public void onErrorResponse(VolleyError error) {
                VolleyLog.e("Error: ", error.getMessage());
             }
          })
       {

       @Override
       public Map<String, String> getHeaders() throws AuthFailureError {
          HashMap<String, String> headers = new HashMap<String, String>();
          headers.put("Content-Type", "application/json; charset=utf-8");
          headers.put("User-agent", "My useragent");
          return headers;
       }
    };
    queue.add(myRequest);
  }
}
```

In der implementierten App wurde dieser Vorgang über ein Button-Onclick event angestoßen. Es ist zu erwähnen, dass Debugging bei der Durchführung dieses PoC auf einem Android Device (nicht dem Emulator) durchgeführt wurde. Damit die verwendeten Localhost-Adressen die Entwicklungsmaschine adressierten, nicht das Android Gerät, wurde auf das Reverse-Port-Forwarding Feature von Googles Chrome Browser zurückgegriffen[1]. Somit konnte auch von dem Android Device aus die Localhost-Adresse der Entwicklungsmaschine verwendet werden.

Die Implementierte Anwendung erfüllt das Exit-Kriterium dieses PoC, der ermittelte Gerätestandort kann an den Server übertragen, die Antwort auf dem Endgerät angezeigt werden.

Quellen:

[1] Anleitung für Debugging mit Reverse Port Forwarding
https://developer.chrome.com/devtools/docs/remote-debugging#reverse-port-forwarding