Technology
Arts Sciences
TH Köln

TH Köln Campus Gummersbach
Fakultät für Informatik und
Ingenieurswissenschaften

Entwicklungsprojekt Interaktive Systeme

Implementationsdokumentation

Thuy Trang Nguyen
Duc Giang Le

betreut von:
Prof. Dr. Kristian Fischer
Prof. Dr. Gerhard Hartmann
Robert Gabriel
Sheree Saßmannshausen

16. Januar 2017

1 Installationsdokumentation

Vorweg ist zu sagen, dass die folgende Beschreibung zur Installation des Systems auf Grundlage des Windows-Betriebssystems erfolgt, da für das Team nur lediglich dieses Betriebssystem zur Verfügung steht.
Der allererste Schritt um das System zu installieren, ist das Klonen bzw. Synchronisieren unseres Github-Verzeichnisses. Dadurch werden die Dateien geholt bzw. auf den neusten Stand gebracht.

1.1 Server

Systemvoraussetzung

- NodeJS
- MongoDB
- Internetverbindung

Installation des Servers

1. Es muss eine Verbindung zur MongoDB-Datenbank gestartet werden. Dies erfolgt mit dem Befehl mongod.exe. Davor muss mithilfe der Kommandozeile in das MongoDB-Verzeichnis navigiert werden.
   Beispiel: C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe
2. In einer anderen Kommandozeile muss in das root-Verzeichnis des Servers navigiert werden. Dort können mit dem Befehl npm install alle wichtigen Abhängigkeiten installiert werden.
   Beispiel: ...EISWS1617NguyenLe\MS3\Implementation\PreFer-Server
3. Mit dem Befehl node app.js wird der Server auf dem Port 3000 gestartet.
4. Auf der Konsole erscheint der Text "Server listens on Port 3000", wenn der Server erfolgreich verbunden ist und der Text "DB connected!" wird angezeigt, wenn der Server erfolgreich mit der MongoDB-Datenbank verbunden ist.
5. Mit dem Befehl gulp plantdb lässt sich die Datenbank mit fiktiven Pflanzen-Daten füllen. Jedoch muss vorher mit dem Befehl npm install –global gulp gulp global installiert werden.
6. Durch REST-Clients wie PostMan können manuelle POST-Requests an den Server gesendet werden. In der Datei soilData.json sind einige Bodendaten definiert.

1.2 Android Client

Systemvoraussetzung

- Aktuellste Android Studio Version (Android Studio 2.2.3)
- Folgende Pakete müssen aus dem Android SDK Manager installiert werden:
  - Android 6.0 (API 23)
  - Android SDK Tools 25.2.5
  - Google Play Services
  - Intel x86 Emulator Accelerator (HAXM installer)
  - Android Support Repository
  - Google Repository
- Internetverbindung (optimal: im gleichen Netz wie das Gerät, auf dem der Server läuft)

Android Client über den Emulator starten

1. Android Studio starten und aus dem Implementations-Ordner den EISWS1617PreFer öffnen.
2. Es muss ein Android Virtual Device eingerichtet werden. Während der Implementationsphase wurde das Nexus 5X API 23 genutzt.
3. Der Server muss nun gestartet werden. Es muss unbedingt darauf geachtet werden, dass der Server problemlos läuft.
4. Den Emulator starten und im Emulator die App starten.
5. Alternativ kann auf dem Emulator die kompilierte APK installiert und anschließend gestartet werden.

2 Abweichungen von Modellen aus MS2 und Anmerkungen

Im Folgenden sollen Abweichungen von Modellen aus MS2, die während der Implementationsphase entstanden sind, erläutert werden.
Als Erstes wurde erkannt, dass es in der modellierten Version der REST-Spezifikation keine Möglichkeit gab, dass der Server die simulierten Bodendaten empfangen kann. Deshalb wurde eine zusätzliche Ressource soil hinzugefügt, damit auf dieser Ressource die Bodendaten empfangen und verarbeitet werden können.
Außerdem wurde in der Projektdokumentation erwähnt, die Nährstoffkarte mithilfe einer Interpolation der berechneten Düngeempfehlung zu erstellen. Zur Abgabe der Implementation konnte dieser Teil der Anwendungslogik nicht realisiert werden. Während der versuchten Realisierung dieses Aspektes wurden verschiedene Lösungsansätze getestet, die jedoch nicht zu einem zufriedenstellenden Ergebnis geführt haben. Eines dieser Lösungsansätze war mithilfe der HeatMap aus der GoogleMaps API die Düngeempfehlung zu visualisieren. Der folgende Code repräsentiert den Lösungsansatz:

```
...
public class HeatMapActivity extends AppCompatActivity
    implements OnMapReadyCallback {

    private GoogleMap mMap;

    private HeatmapTileProvider mProvider;
    private TileOverlay mOverlay;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_heatmap);

        SupportMapFragment mapFragment =
                (SupportMapFragment)
                   getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap map) {
        mMap = map;
        mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
        addHeatMap();
    }

    private void addHeatMap() {
```

```
        ArrayList<WeightedLatLng> list = new
            ArrayList<WeightedLatLng>();
        LatLng point1 = new LatLng(51.054900, 7.552804);
        LatLng point2 = new LatLng(51.055278, 7.553869);
        LatLng point3 = new LatLng(51.055602, 7.554901);
        LatLng point4 = new LatLng(51.055333, 7.555560);
        LatLng point5 = new LatLng(51.054805, 7.556392);

        list.add(new WeightedLatLng(point1, 1.2));
        list.add(new WeightedLatLng(point2, 1.5));
        list.add(new WeightedLatLng(point3, 2));
        list.add(new WeightedLatLng(point4, 1.7));
        list.add(new WeightedLatLng(point5, 0.1));

        int[] colors = {
                Color.rgb(255, 0, 0),    // red
                Color.rgb(102, 225, 0) // green
        };

        float[] startPoints = {
                0.2f, 1f
        };

        Gradient gradient = new Gradient(colors,
            startPoints);

        mProvider = new HeatmapTileProvider.Builder()
                .weightedData(list)
                .gradient(gradient)
                .build();
        mProvider.setRadius(100);
        mOverlay = mMap.addTileOverlay(new
            TileOverlayOptions().tileProvider(mProvider));
    }
}
```

Nach der Realisierung der HeatMap wurde ersichtlich, dass eine HeatMap nicht der richtige Lösungsweg für das Erstellen der Nährstoffkarte ist, da die HeatMap nur die Dichte der verteilten Punkte visualisiert. Die Verteilung der Nährstoffwerte werden bei einer HeatMap nicht berücksichtigt.