Google Maps und Nokia Maps in APEX

Dr. Jan Golka
Data Design & Management GmbH
Remagen

Schlüsselworte
PL/SQL, APEX, Spatial, JavaScript.

Einleitung
Es gibt inzwischen eine große Anzahl von Beispielen für die Verwendung von Google Maps in APEX. Allerdings wird dort in der Regel die Google Maps JavaScript API Version 2 (v2) verwendet, die nicht mehr funktioniert (bzw. ab dem 8.9.2013 nicht mehr funktionieren wird).

Die nachfolgende JavaScript (JS) API v3 unterscheidet sich von der v2 nicht nur dadurch, dass jetzt kein API Key notwendig sein wird. Viele Anwendungen werden z. B. auch davon betroffen sein, dass in API v3 eine Request-Antwort nur als JSON oder XML geliefet wird. Dennoch ist die API v3 wesentlich klarer strukturiert und dadurch einfacher im Einsatz, wie die nachfolgenden Codebeispiele belegen.

Nach dem Erwerb von NAVTEQ bietet jetzt auch Nokia eine freie Nutzung der eigenen Maps inklusive JavaScript API an. Ihre Anwendung in APEX hat gegenüber Google Maps den zusätzlichen Vorteil, dass ihre Basisdaten auch für die Oracle Datenbank - zum Teil sogar frei - verfügbar sind.

Im Demoteil des Vortrags wird der Gebrauch der Google und Nokia Maps (JS API und REST) in APEX in einer live Demo veranschaulicht. In zahlreichen Beispielen werden außerdem Ähnlichkeiten sowie Unterschiede der beiden Maps APIs analysiert und demonstriert.

Frei verfügbare Kartendienste von Google und Nokia
Sowohl Google als auch Nokia verfügen über immense Mengen von Geodaten und bieten umfangreiche Schnittstellen und Tools zu deren Nutzung und Darstellung, teils sogar als frei verfügbare Dienste.

Die Nutzungsbedingungen erlauben einen freien Zugriff auf die Geodaten nur unter der Voraussetzung, dass die Daten auch auf der Karte angezeigt werden. Außerdem wird die nichtkommerzielle Kartendienstnutzung über eine maximale Anzahl von Zugriffen pro Tag eingeschränkt (z. B. Google's Geocoding API erlaubt max. 2.500 Geocoding-Requests pro Tag).

Google verzichtet inzwischen auf einen API-Key, für die Nutzung der Nokia Kartendienste ist jedoch die Beantragung der entsprechenden Credentials („appId", „authenticationToken") auf per-Anwendung Basis notwendig.

Grundlagen
Google und Nokia bieten die freie Nutzung der Geodaten sowie die Programmierschnittstellen für ihre Verwendung in Browserprogrammen, allerdings ohne direkte Zugriffsmöglichkeiten auf die Rohdaten.

Angeboten werden jeweils zwei Arten von Programmierschnittstellen:

1. Eine JavaScript (JS) API, bestehend aus einem Satz von Funktionsaufrufen in der Programmiersprache JavaScript, die auch für einen dynamischen Zugriff z. B. aus einem BO Element geeignet sind.

2. Webservices, also eine Sammlung der HTTP-Schnittstellen zum Bereitstellen der geografischen Daten für Kartenanwendungen. Google bietet nur RESTful Webservices, Nokia - sowohl REST als SOAP (allerdings nur für kommerzielle Nutzer).

Gemeinsam ist in allen Fällen die Verwendung von XML und JSON ("JavaScript Object Notation") als Ausgabeformate.

Alle Schnittstellen haben ungefähr den gleichen Funktionsumfang, von Karten, Geocoding und Reverse-Geocoding sowie Elevation hin zu Services wie Routing bzw. Directions, Positioning und Traffic. Die Places-Bibliothek erlaubt mehr oder weniger ausgefallene Suchoptionen für "Points of Interests" (PoI) sowie deren umfangreiche Details. Nokia bietet sogar eine "UI Bibliothek" mit Oberflächenwidgets für die Suche und Darstellung der PoI Daten.

Google Maps JavaScript API v3: Änderungen gegenüber der API v2

1) Kein API Key mehr notwendig.

2) Die Ausgabe als CSV ist nicht mehr möglich. JSON ist das empfohlene Ausgabeformat.

3) v2 Codebeispiel:

```
function initialize() {
   if (GBrowserIsCompatible()) {
       var map = new GMap2(document.getElementById(„karte"));
       map.setCenter(new GLatLng(37.4419,-122.1419), 13);
       map.setUIToDefault();
   }
}
```

4) Gleiches Codebeispiel in API v3:

```
function initialize() {
   var myOptions = {zoom: 13,
                    center: new google.maps.LatLng(37.4419, -122.1419),
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                   };
   var map = new google.maps.Map(
           document.getElementById(„karte"),myOptions);
}
```

5) Google Empfehlungen für die v2 --> v3 Migration:
http://www.google.com/events/io/2010/sessions/v2-javascript-maps-api-v3.html

Google Maps API Web Services (REST)

URL für die Webanfrage:
`http://maps.googleapis.com/maps/api/<dienst>/<output>?<parameters>`

`dienst:` `geocoding, elevation, directions, ...`
`output:` `json` (empfohlen), Ausgabe in JavaScript Object Notation (JSON).
`xml`, Ausgabe in XML-Format.
`parameters:` Kontextabhängig. Abschluss mit „`sensor=(true/false)`".

URL-Beispiel:
`http://maps.googleapis.com/maps/api/geocode/json?address=Hauptstr.+1+Remagen&sensor=false`

Der Parameter `sensor=false` besagt, dass die Anwendung keinen Sensor zur Standortbestimmung verwendet.

Codebeispiel 1: Kartenanzeige mit Hilfe der Google Maps JS API

Bibliothek: `"http://maps.google.com/maps/api/js?sensor=false"`

```
function initMap() {
    var options = {
       zoom: 10,
       center: new google.maps.LatLng(&P23_LOCATION.),
       mapTypeId: google.maps.MapTypeId.ROADMAP,
       mapTypeControl: true,
       mapTypeControlOptions:  {style:
          google.maps.MapTypeControlStyle.DROPDOWN_MENU},
       scaleControl: true,
       navigationControl: true,
       navigationControlOptions: {style:
          google.maps.NavigationControlStyle.DEFAULT},
       keyboardShortcuts: false
    };
    var map = new
       google.maps.Map(document.getElementById("karte"), options);
    var marker = new google.maps.Marker();
    marker.setPosition(map.getCenter());
    marker.setMap(map);

    var title = "Str.: &P23_STREET. <br>Ort: &P23_CITY.<br>Koord.:
&P23_LOCATION.";
    var infoWindow = new google.maps.InfoWindow();
    infoWindow.setContent(title);
    infoWindow.open(map, marker);
}
```

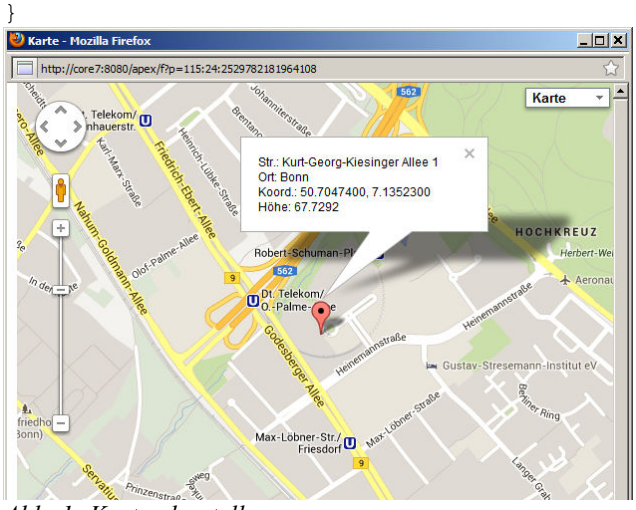

Abb. 1: Kartendarstellung.

Codebeispiel 2: Kartenanzeige mit Hilfe der Nokia Maps JS API

Bibliothek: `"http://api.maps.nokia.com/2.2.4/jsl.js?with=all"`

```
nokia.Settings.set("appId", "...");
nokia.Settings.set("authenticationToken", "...");

function initMap() {
   infoBubbles = new nokia.maps.map.component.InfoBubbles();
   var options =
     { center: [&P23_LOCATION.]
     , zoomLevel: 12
     , components: [infoBubbles
     , new nokia.maps.map.component.Behavior()
     , new nokia.maps.map.component.ZoomBar()
     , new nokia.maps.map.component.ScaleBar()
     , new nokia.maps.map.component.TypeSelector()
     , new nokia.maps.map.component.Overview()
     , new nokia.maps.map.component.PublicTransport()
     , new nokia.maps.map.component.Traffic() ]
     };
   var map = new
      nokia.maps.map.Display(document.getElementById(„karte"), options);
   var marker =
      new nokia.maps.map.StandardMarker( [&P23_LOCATION.],
        { text: "A", draggable: false });
   map.objects.add(marker);

   var title = "<div>"+"Str.: &P23_STREET."+"<br/>Ort.:
&P23_CITY."+"</div>";
   var bubble = infoBubbles.addBubble(title, [&P23_LOCATION.]);
}
```

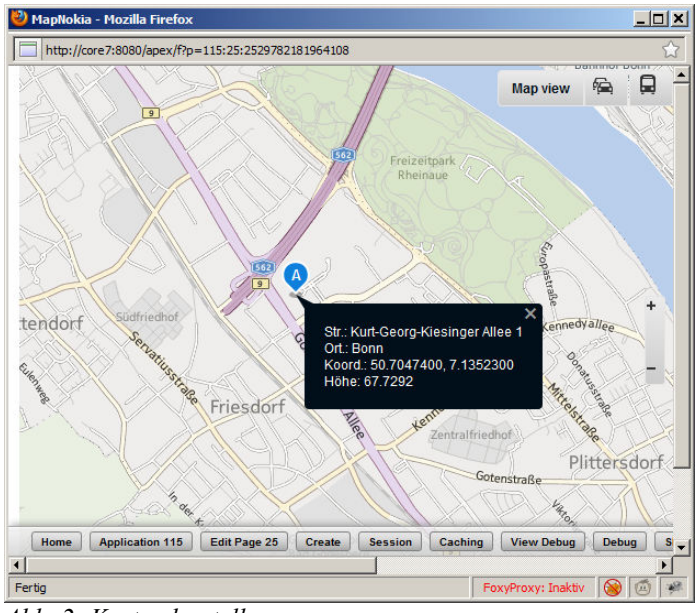

Abb. 2: Kartendarstellung.

Komplexeres Beispiel: Google Routenplanung

JavaScript Code:

```
var directionDisplay;
var directionsService = new google.maps.DirectionsService();
var map;

function initialize() {
  directionsDisplay = new google.maps.DirectionsRenderer();
  var point = new google.maps.LatLng(50.5, 7.2);
  var mapOptions = {
    zoom:7,
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    center: point,
    mapTypeControlOptions: {style:
       google.maps.MapTypeControlStyle.DROPDOWN_MENU}
  }
  map = new google.maps.Map(document.getElementById("karte"), mapOptions);
  directionsDisplay.setMap(map);
  directionsDisplay.setPanel(document.getElementById("directions"));
}

function calcRoute() {
var selectedMode = $v("P36_MODE");
var request = {
   origin: $v("P36_START_ADR"),
   destination: $v("P36_ENDE_ADR"),
   travelMode: google.maps.DirectionsTravelMode[selectedMode]
   };
   directionsService.route(request, function(response, status) {
     if (status == google.maps.DirectionsStatus.OK) {
         directionsDisplay.setDirections(response);
     }
   });
}
```

Ergebnis für `origin` = "Kurt-Georg-Kiesinger Allee 1, Bonn", `destination` = " Hamborner Straße 51, Düsseldorf ", und `travelMode` = "Transit":

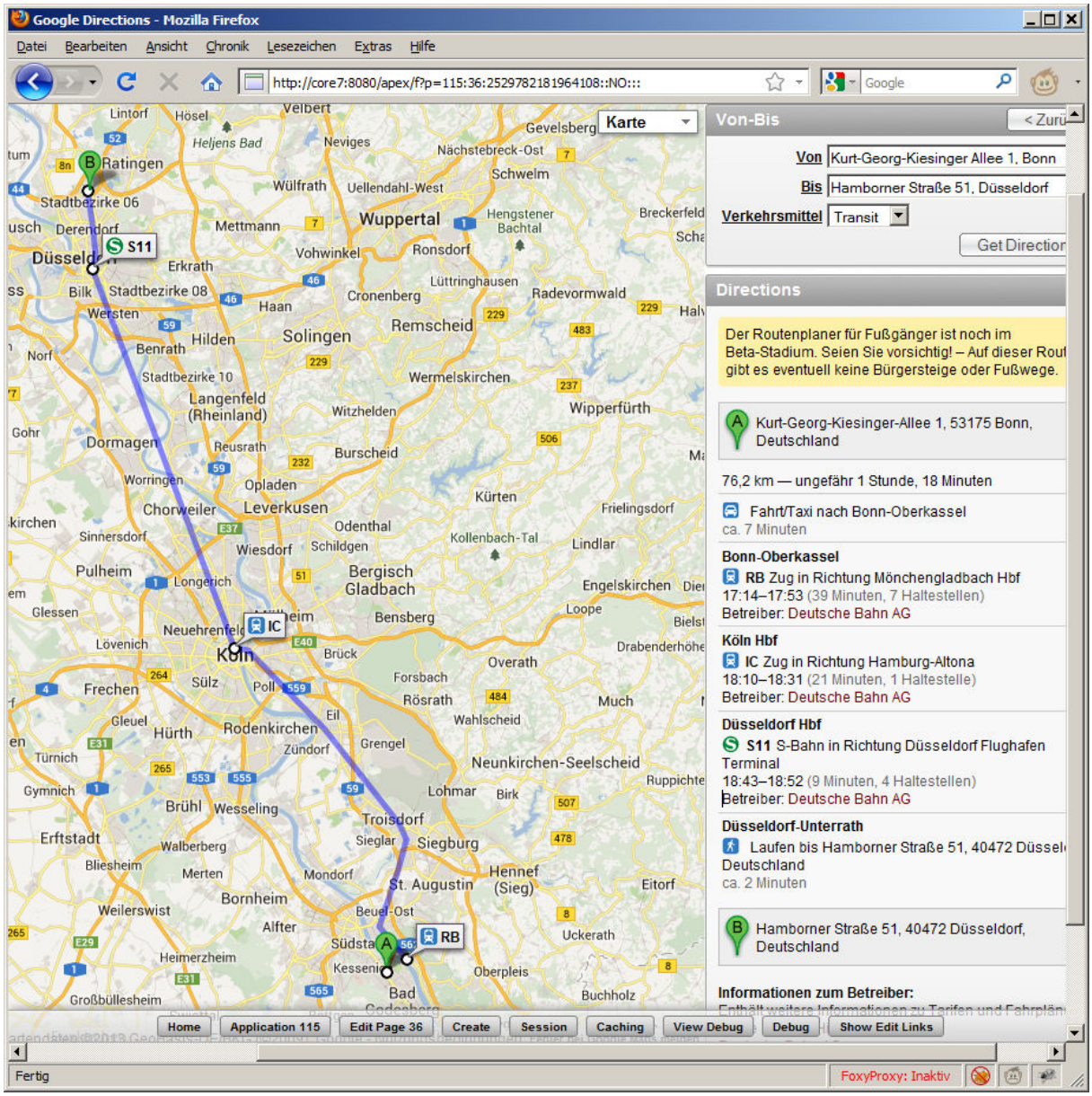

Abb. 3: Google Directions.

Beispiel für Nokia Maps "Places" API

JavaScript Code:

```
nokia.Settings.set("appId", "...");
nokia.Settings.set("authenticationToken", "...");

var mapContainer;
var map;
var searchPlacesCenter = new nokia.maps.geo.Coordinate(48.78, 9.18);
```

```
function initMap() {
   mapContainer = document.getElementById("mapContainer");
   map = new nokia.maps.map.Display(mapContainer, {
      center: [48.78, 9.18],
      zoomLevel: 10,
      components: [new nokia.maps.map.component.Behavior()
                  ,new nokia.maps.map.component.ZoomBar()
                  ,new nokia.maps.map.component.ScaleBar()
                  ,new nokia.maps.map.component.TypeSelector()
                  ,new nokia.maps.map.component.Overview()
                  ,new nokia.maps.map.component.PublicTransport()
                  ,new nokia.maps.map.component.Traffic() ]
   });
} // initMap

function getSearchCenter(){
   var startWo = $v("P79_WO");
   var searchStart = new nokia.maps.geo.Coordinate(48.78, 9.18);

   var processResults = function (data, requestStatus) {
      var locations;

      if (requestStatus == "OK") {
         locations = data.results ? data.results.items : [data.location];
         if (locations.length > 0) {
            $s("P79_LOCATION",locations[0].position.latitude + ", "
               + locations[0].position.longitude);
            $s("P79_LAT",locations[0].position.latitude );
            $s("P79_LNG",locations[0].position.longitude);
         } else {
            alert("Kein Ergebnis");
            searchStartCenter = { latitude: 48.78344110,
                                  longitude: 9.1785870 };
         }
      } else {
         alert("Status: "+requestStatus);
      }
   };

   function searchCenter(address) {
      nokia.places.search.manager.findPlaces({
         searchTerm : address,
         onComplete:  processResults,
         searchCenter: searchStart
      });
   }
   searchCenter(startWo);
} // getSearchCenter

function searchPlaces() {
   var serchManager = nokia.places.search.manager;
   var resultSet;
   searchPlacesCenter = {
         latitude: &P79_LAT.,
         longitude: &P79_LNG.
```

```
   };
   var term = $v("P79_WAS");

   var processCentersResults = function (data, requestStatus, requestId) {
      var i, len, locations, marker;
      if (requestStatus == "OK") {
         locations = data.results ? data.results.items : [data.location];
            var search_res = data;
            var cnt = search_res.results.items.length;
            $s("P79_ANZ_ITEMS",cnt);

            if (locations.length > 0) {
               if (resultSet) map.objects.remove(resultSet);
               resultSet = new nokia.maps.map.Container();
               for (i = 0, len = locations.length; i < len; i++) {
                  marker = new
                    nokia.maps.map.StandardMarker(locations[i].position,
                      { text: i+1 });
               resultSet.objects.add(marker);
            }
            map.objects.add(resultSet);
            map.zoomTo(resultSet.getBoundingBox(), false);
        } else {
                  alert("Your search produced no results!");
            }
      } else {
            alert("The search request failed");
      }
   };

   nokia.places.search.manager.findPlaces({
      searchTerm: term,
      onComplete: processCentersResults,
      searchCenter: searchPlacesCenter,
      limit:50
   });
} // searchPlaces

onload="getSearchCenter(), initMap(), searchPlaces()".
```

Ergebnis für die Suche nach "Hotel" in "München":

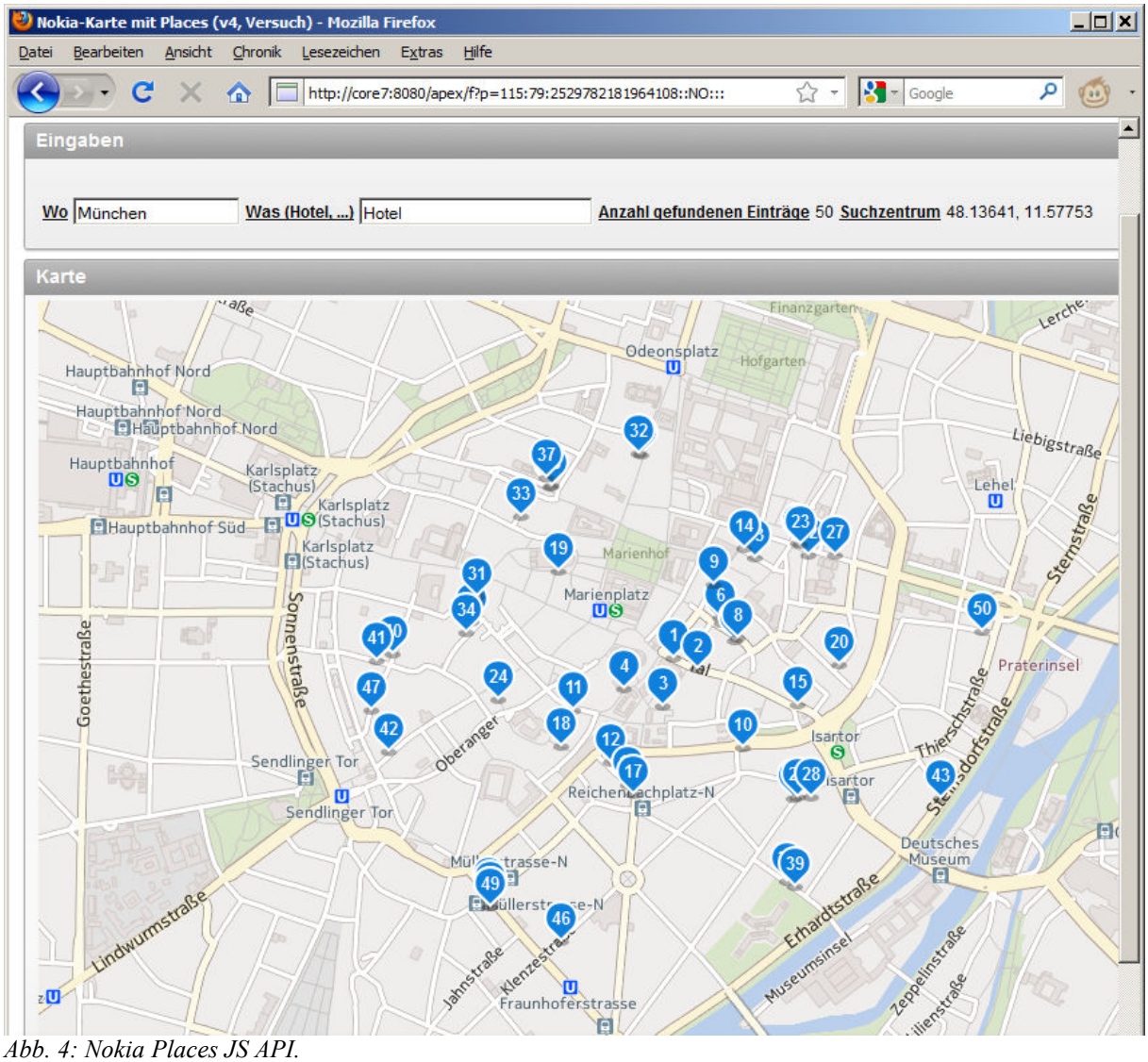

Abb. 4: Nokia Places JS API.

Nokia Places Widget

JavaScript Codefragment:

```
function showPlace(p_id) {
   var template = $v("P78_TEMPLATE");
   var basicPlace = new nokia.places.widgets.Place({
      placeId: p_id,
      targetNode: "placeContainer",
      template: template
   });
}
```

```
"placeId" ist ein Nokia Objekt-ID
(z. B. 276u281v-04063715bf074e8f9ba1b7ffcf3dd776).
```

Nokia Places Widget "Nokia.blue.extended" für das Objekt 15:

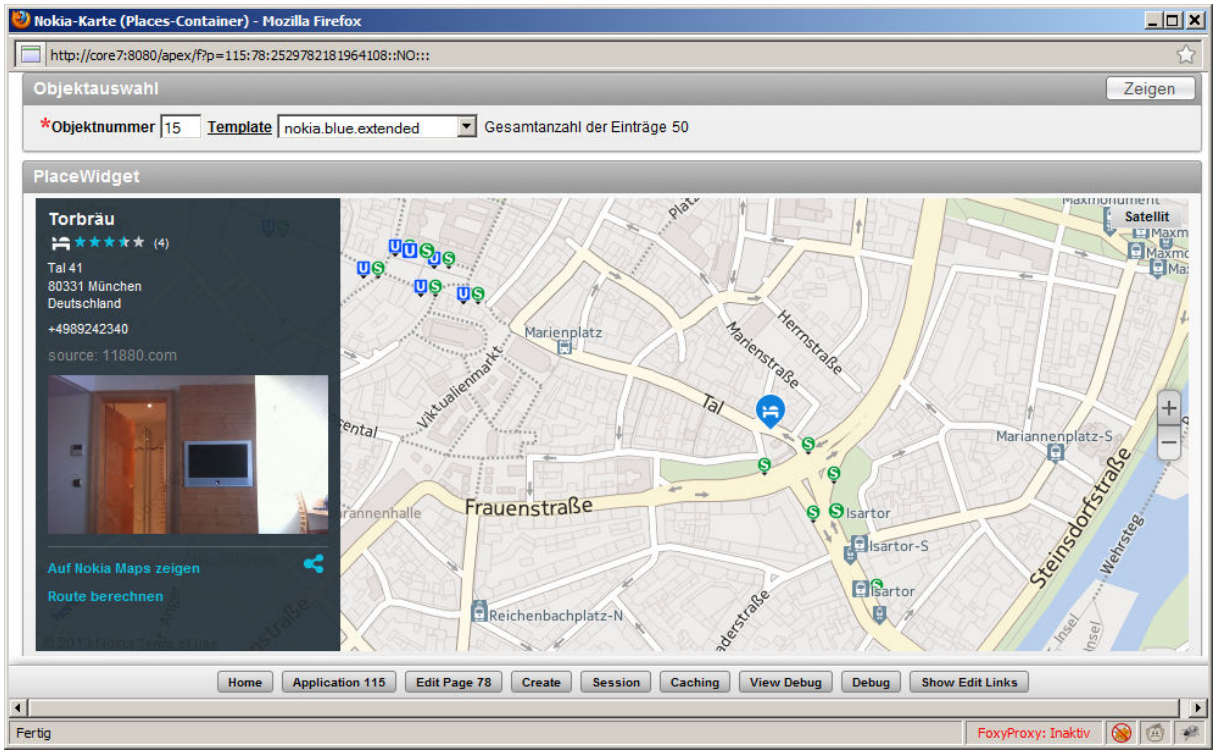

Abb. 5: Nokia Places Widget.

Kontaktadresse:
Dr. Jan Golka
Data Design & Management GmbH
Westerwaldweg 16
D-53424 Remagen

Telefon: +49 (0) 172-710 3230
E-Mail j.golka@d-d-m.de
Internet: www.d-d-m.de