Steakhunt

Schnitzeljagd mit Google

Fabian Tschirschnitz
Fabian Bornhofen

2010-07-12

Seminar Mobile Application Development with Android

Agenda

- Motivation
- Anforderungen
- Demo
- Mobile Karten
- Architektur
- Datenmodell & Designentscheidungen
- Ausblick
- Fazit

Motivation

- Inspiration: GeoCaching; Schnitzeljagden basierend auf Geodaten
- GeoCaching ist kommerzieller Service
- Idee: Stadtführungen für Gruppen (Schulklassen etc.) in Form einer Schnitzeljagd
- Austausch der Positionen der Teilnehmer schafft Wettbewerbsgedanken

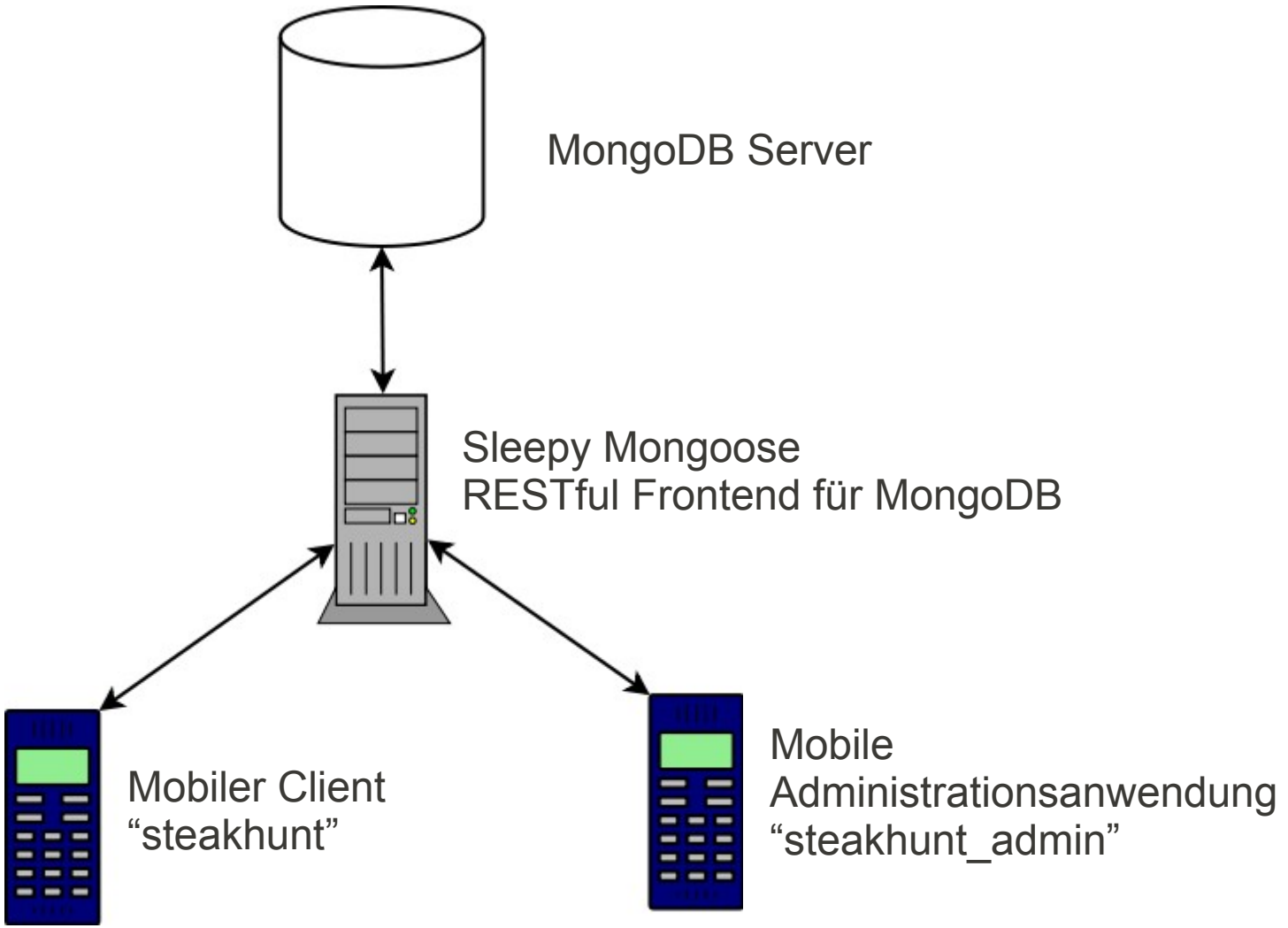
Anforderungen

- Bedienung mit mobilen Applikationen

  → möglichst wenig Administration
- Schnitzeljagden (“Quests”) persistieren und mehrfach durchführen
- Abschirmung verschiedener Teilnehmergruppen gegeneinander
- Schulklassen: Wo befinden sich die Schüler gerade?

Demo

Architektur



Architektur

- Ziel: Serverseitigen Implementierungsaufwand gering halten (Erfolg mäßig) → fertiges RESTful Frontend für Datenbank
- Sleepy Mongoose für MongoDB führte zu Problemen mit Apache Commons HttpClient (100-continue)
- Lösung: schlanker HTTP-Client für Sleepy Mongoose auf BSD-Sockets in < 100 Zeilen; Kapselung der Anfragen kostenlos dazu

Mobile Karten

- OpenStreetMap (i.V.m. Sony Ericsson Maps) zu unausgereifte API (Overlays, etc.)
- andere OSM-Lösungen ebenfalls zu unausgereift
- FU-Lösung interessant aber zu spät
- Google-Maps bot alle interessanten Funktionen und mit Abstand beste Dokumentation

Datenmodell

- Erster Kontakt mit Key-Value-Store als Entwickler
- Fragestellungen: Was gehört in welche Datenbank, wann fassen wir Objekte in Collections zusammen?

Datenmodell: Designentscheidungen

- 3 Datenbanken
  - “quests” - Fragen und Positionen der Quests
  - “instances” - Instanzen der Quests
  - “locations” - ID und Positionen der Teilnehmer
- Identifikation von Instanz und Nutzern über einzelnes Token → Verzicht auf Login-Dialog
  - Schema: 6 Zeichen Instanzname, beliebig viele für Teilnehmer-ID (UserName)
  - Nachverfolgbarkeit durch vorheriges Festlegen (TODO!)

Datenmodell: JSON

quests/demo

```
{
  "QuestName":"Demo",
  "Quests":[
    {
      "GeoPoint":{"Latitude":
52400000,"Longitude":13130000},
      "Question":"Was verlief an
dieser Stelle?",
      "Answers":["Chinesische Mauer",
"Berliner Mauer","Potsdamer Mauer",
"Maginot-Linie"],
      "rightAnswerIndex":1
    },
    {
      "GeoPoint":{"Latitude":
52430000,"Longitude":13180000},
      "Question":"Welches sind die
coolsten Boote im Bootsverleih
Wannsee?",
      "Answers": ["Bundeswehr-
Fregatten","Tretboote","Solarboote",
"Fischkutter"],
```

instances/demosj

```
{
  “QuestName”:”Demo”
}
```

locations/demosj

```
{
  “UserName”:””,
  “Longitude”:13129980,
  “Latitude”:52400021
},
{
  “UserName”:””,
  “Longitude”:13129980,
  “Latitude”:52400021
}
```

Ausblick: TO DO

- Feinarbeit beim Admin-Tool: Nachträgliche Bearbeitung von Fragen etc.
- Webservice absichern im Falle d. Produktiveinsatzes
- An einigen Stellen Best Practices mehr befolgen (→ Refactoring)
- Lösung für Instanzerstellung und -löschung finden
- Fehlerbehandlung / Robustheit verbessern

Fazit

- Mobile Entwicklung erfordert Einschränkungen in Layout und Workflows
- Programmiermodell klingt einfach, ist aber nicht trivial (wissenintensiv)
- Kennenlernen von “nichtdeterministischen” Entwicklungswerkzeugen (Testen sehr umständlich)
- Blick über den SQL-Tellerrand