PirateBayTours

Projekt im Modul Verteilte

Informationsysteme WS2016/17

Jesko Appelfeller

Robin Naundorf

Frederik Broer

Jonas Droste

betreut durch

Prof. Dr.-Ing. Thomas Christian Weik

4. April 2017

Zusammenfassung

Abstrakt schreiben

Abstract

Abstrakt übersetzen

Inhaltsverzeichnis

Kapitel 1

Motivation und Anforderungen

Im Rahmen des Moduls 'Verteilte Informationssysteme' an der Fachhochschule Münster wird ein Semester begleitendes Datenbank Projekt durchgeführt. Ziel des Projektes ist es, ein Buchungssystem für Bootstouren zu implementieren. Die Firma Pirate-Bay-Tours beschäftigt mehrere Vertriebsmitarbeiter, die an Touristen Hotspots Tickets für Bootstouren vertreiben. Für diese Mitarbeiter soll ein neues Buchungssystem implementiert werden.
Wichtigstes Funktionsmerkmal des Systems ist die Offline-Fähigkeit. Das heißt, auch wenn der Mitarbeiter keine Verbindung zum zentralen Buchungsserver hat, muss die Buchung von Touren möglich sein. Dazu soll eine Replikationslogik entwickelt werden, die offline getätigte Buchungen bei verfügbarer Online-Verbindung mit dem Server synchronisiert. Weiterhin soll ein Quotensystem entwickelt werden, mit dem es möglich ist, verfügbare Tourtickets auf unterschiedliche Agenten zu verteilen. Zusätzlich muss eine Strategie entwickelt werden, wie mit Überbuchungen umgegangen werden soll.

Motivation des Projektes
Was waren seine Anforderungen?

Kapitel 2

Lösungskonzept

2.1 Architektur

Aus welchen Komponenten besteht die Software?
Was sind die Schnittstellen?

2.2 Backend

2.2.1 Django und Admin Interface

Was bieted Django alles von Hausaus?

2.2.2 Database

Wie ist die Datenbank aufgebaut?
Welche Replikation setzen wir ein?
Wie ist diese Implementiert?

2.3 Client

2.3.1 Java Client

Welche Konzepte nutzt der Client? MVC
Wie ist die Struktur im Client?
Welche Klassen wurden implementiert?
Wie erfolgt die Datenhaltung?

2.3.2 Local Cache DB

Um offline einsatzfähig zu sein, wurde eine lokale SQLite Datenbank implementiert. Während der Synchronisierung vom Server zum Client wird ein Abbild der Serverdatenbank heruntergeladen und in der lokalen Datenbank gecached. Wir haben uns entschieden eine SQLite Datenbank zu nutzen, da keine zusätzlich Datenbankserver installiert werden müssen und alle notwendigen Ressourcen gut in die Applikation eingebunden werden können. Weiterhin wird SQLite von einer Vielzahl an Programmiersprachen gut unterstützt und bietet eine ressourcensparende Möglichkeit Daten abzulegen.
Wie erwähnt, enthält die Datenbank nach einmalige Synchronisation ein komplett Abbild der Server-Datenbank. Das heißt, lokal sind folgende Tabellen verfügbar:

- agents
- customers
- quotas
- ships
- tours

Zusätzlich dazu werden zwei lokale Tabellen angelegt.

- offline bookings
- offline customers

Die 'offline bookings' Tabelle enthält alle Buchungen, die der Vertriebsmitarbeiter im Offline-Fall tätigt. Werden im Offline-Modus neue Kundenstammdaten angelegt werden diese in der Tabelle 'offline customers' gespeichert. Dies ist notwendig, um ID Konflikte bei der Replizierung zu vermeiden. Weiterhin lässt sich so mit wenig Aufwand nachvollziehen welche Buchungen noch nicht repliziert wurden. Da das Serverabbild bei einer Offline-Buchung nicht verändert wird, besteht kein Risiko, das es zu inkonsistenten Ausgangsdaten kommt und ein Offline-Arbeiten nicht mehr möglich ist.

2.4 Replizierung

Wie wird repliziert??

Kapitel 3

Geschäftslogik

Wie werden Fälle wie Überbuchen gehandelt?
Wie erfolgt die Replizierung von Server zu Client und zurück?
Wir replizieren an zwei Punkten pro Tag

3.1 Replikationszeitpunkt

Auf Grund der begrenzten Verfügbarkeit von mobilen Internet für unsere mobile Anwendung, wird im Normalfalle nur zu bestimmten Zeitpunkten eine Replizierung von der zentralen Datenbank zur lokalen Datenbank und zurück durchgeführt. Beider Vorgänge werden manuell vom Benutzer ausgelöst. Bei erhöhter Verfügbarkeit des Internets steht es diesem frei häufiger zu replizieren, minimal sind jedoch folgende Replizierungen notwendig:

Muss eine Reihenfolge eingehalten werden?

Lesend/Morgens Zu Beginn des Arbeitstages muss jeder Agent einen Lesevorgang auslösen. Dies geschieht über das GUI mittels der Schaltfläche

Füge Namen des Buttons ein

. Hierbei werden die Tabellen Schiffe, Routen, Buchungen und Quoten

TTabellenNamen checken

vom Server abgeholt und in die lokale DB eingelesen. Somit verfügt der Agent nach Abschluss des Vorgangs über alle Touren und Schiffe und kann mittels der bereits vorhandenen Buchungen ausrechnen welche Pältze noch frei sind. Er kennt außerdem seine persönlichen Quoten und weiß daher wie viele Pätze er maximal verkaufen kann.

Schreibend/Abends

Kapitel 4

Fazit

Was sind die Lessons Learned
Was könnte man das nächste mal besser machen?
Waren unsere ausgewählten Komponenten für das Problem geeignet?