Technology
Arts Sciences
TH Köln

Medieninformatik

Projektdokumentation
Entwicklungsprojekt Interaktive Systeme

Entwicklung eines Raumbuchungssystems

Betrachtung der Laufwegoptimierung und Zeitersparnis bei der Raumsuche

Implementierungsdokumentation

Webserver und Datenbankserver

Abgabetermin: Gummersbach, den 28.01.2018

Prüfungsbewerber:

Bastian Fuchshofer
Dieringhauser Str. 107
51645 Gummersbach

Niklas Fonseca Luis
Dieringhauser Str. 107
51645 Gummersbach

Inhaltsverzeichnis

1 Funktionen des Systems **2**
1.1 Allgemeine Funktionsweise . . . 2
1.1.1 Webserver . . . 2
1.1.2 Datenbankserver . . . 2
1.2 Routen und Abläufe . . . 2
1.2.1 Webserver . . . 2
1.2.2 Datenbankserver . . . 4

2 Systemvoraussetzungen **4**

3 Installation **5**

4 Einrichtung **5**
4.1 Webserver . . . 5
4.2 Datenbankserver . . . 5

5 Nutzung **6**

6 Modellierte Knotenpunkte **6**
6.1 Gebäude . . . 6
6.2 Treppenhäuser . . . 6
6.3 Aufzüge . . . 6
6.4 Räume . . . 7

1 Funktionen des Systems

1.1 Allgemeine Funktionsweise

In dem für dieses Projekt implementierten Prototypen wurden lediglich die für die Ausführung einer Buchung, bzw. die Stornierung einer Buchung benötigten Funktionen implementiert.

1.1.1 Webserver

Der in diesem Projekt erstellte Webserver dient als Mittelsmann zwischen der Anwendung auf dem Endgerät des Benutzers, dem Datenbankserver mit angehängten Datenbanken und den Mini-PCs. Er übernimmt die Weitergabe von Informationen und Daten zwischen diesen Komponenten und ebenfalls die Verarbeitung bzw. die Anreicherung einiger dieser Daten.
Auf dem Webserver wird die verkettete Liste der Knotenpunkte erstellt und verwaltet. Außerdem werden über die Routen des Webservers die einzelnen Funktionalitäten des Systems abgefragt und ausgeführt.

1.1.2 Datenbankserver

Der Datenbankserver dieses Projektes ist für die Verwaltung der angehängten Datenbanken zuständig, also das Speichern, Abrufen und Löschen von für das System wichtigen Informationen und Daten. Die an den Datenbankserver angehängten Datenbanken beinhalten die Informationen über wöchentlich wiederkehrende Veranstaltungen und Informationen über Räume, Gegenstände, für die Erstellung der verketteten Liste wichtige Informationen über Knoten, aber auch Informationen über Treppen, Ein- und Ausgänge und Aufzüge. Während der Laufzeit werden weiterhin die Informationen über eine aktuelle Reservierung oder Buchung in der Datenbank gesichert. Des Weiteren wird bei Start des Datenbankservers eine Liste mit freien Räumen erstellt, indem die Räume mit den momentan stattfindenden Veranstaltungen abgeglichen werden.

1.2 Routen und Abläufe

1.2.1 Webserver

Da in diesem Projekt keine Informationen über einen Browser beim Webserver abgerufen werden sollen, sondern nur Informationen an den Webserver übertragen werden, besteht der Router des Webservers nur aus POST Anfragen mit den dazugehörigen Pfadangaben. Im Folgenden werden die Routen aufgelistet und sowohl die inneren Abläufe, als auch die im Request body benötigten Daten:

Tabelle 1: Webserver Routen

Route	**Body**	**Abläufe**
/room	user, beacon, room_id, person, roomType, blackboard, whiteboard, beamer, chairTable, token	Über die Route /room werden alle durch das Endgerät des Benutzers angeforderten Aufgaben erledigt. Damit die verschiedenen Schritte durchgeführt werden können wird für das Feld *token* entweder *GET* für die Suche und das Reservieren eines Raumes, oder *UPDATE* für die erneute Suche und Reservierung eines Raumes im Falle eines Standortwechsels, oder *BOOK* bzw. *EXTEND* für die Buchung eines bereits reservierten Raumes und die Verlängerung einer Buchung, oder *CANCEL* für den Abbruch einer Reservierung oder einer Buchung. Nur für die Tokens *GET* und *UPDATE* werden zusätzlich die Felder **person**, **roomType**, **blackboard**, **whiteboard**, **beamer** und **chairTable** benötigt. Die die Tokens *UPDATE*, *BOOK*, *EXTEND* und *CANCEL* benötigen zusätzlich das Feld **room_id**.
/miniPcPing	room_id, host, port	Die Route /miniPcPing dient dazu die IP-Adresse für einen bestimmten MiniPC im System zu vermerken.
/roomList	room_id, data	Die Route /roomList wird von einem MiniPC aufgerufen, wenn sich ein Rauminhalt geändert hat.
/databasePing	ip, addr, port	Über diese Route meldet sich der Datenbankserver beim Start beim Webserver, sodass die IP-Adresse vermerkt werden kann.

Bei der Route /room werden abhängig des gesetzten Tokens unterschiedliche Aktionen durchgeführt. Für das Token *GET* wird erst überprüft, ob für diesen Benutzer bereits eine Reservierung oder Buchung vorliegen. Daraufhin wird durch die Verwendung des Djikstra-Algorithmus und die gewichtete verkettete Liste der nächstgelegene freie Raum ausgegeben. Dieser freie Raum wird dann für den Benutzer reserviert oder gebucht und die Buchung im System vermerkt. Bei den anderen Tokens wird jeweils nur überprüft der angegebene Raum auch wirklich für den Benutzer reserviert oder gebucht wurde und führen dann die durch die Tokens angedeutete Funktion aus.

1.2.2 Datenbankserver

Der Datenbankserver verfügt ebenfalls nur über POST Anfragen. Die Anfragen sind direkt an die Befehle der Datenbank angelehnt und dienen nur dem Speichern oder Abrufen von Daten.

Tabelle 2: Datenbankserver Routen

Route	**Body**	**Abläufe**
/keys	key	Über diese Route wird aus der Datenbank eine Liste mit Keys ausgegeben die mit dem mitgesendeten **key** beginnen.
/set	key, data	Diese Route setzt die in **data** mitgelieferten Daten unter dem Schlüssen **key**.
/get	key	Ruft die unter **key** befindlichen Daten ab.
/del	key	Löscht den **key** mitsamt der Daten.
/setHash	key, data	Setzt ein Hash mit den Daten in **data** unter dem Schlüssel **key**.
/setHashField	key, field, data	Setzt ein einzelnes Feld eines Hashes.
/getHash	key	Ruft ein bestimmten Hash ab.
/getHashField	key, field	Ruft ein einzelnes Feld eines Hashes ab.
/addToList	key, data	Erstellt oder fügt ein Element einer Liste hinzu.
/removeFromList	key, data	Löscht das Element in **data** aus der Liste unter **key**.
/getList	key	Gibt die gesamte Liste unter **key** aus.
/getUsedRooms		Gibt eine Liste mit zum Zeitpunkt der Anfrage genutzten Räumen zurück.

2 Systemvoraussetzungen

Für den Webserver, den Datenbankserver und die Datenbanken werden folgende Dinge vorausgesetzt.

- Betriebssystem mit NodeJS
- Redis
- Internetzugang
- diverse Module

3 Installation

- Laden Sie den Ordner **Webserver**/ auf das System
- Öffnen Sie die Unterordner **server**/ und **database**/
- Führen Sie in beiden Unterordnern den Befehl *npm install* aus.
- Installieren Sie Redis auf ihrem System.

4 Einrichtung

4.1 Webserver

- Sie befinden sich im Ordner *Webserver/server/*
- Öffnen Sie den Unterordner **cfg**/
- Setzen Sie in der Datei *webserver.json* unter dem Feld *port* den Port den Sie für den Webserver verwenden wollen.
- Geben Sie die IP-Adresse des Systems auf dem der Webserver läuft unter dem Feld *ip*, und *address* ein.
- Geben Sie die IP-Adresse und den Port des Datenbankservers in den Feldern *database.addr*, *database.ip* und *database.port* ein.
- In der Datei *Webserver/server/utiltest_data.json* befinden sich Testdaten die Sie bei Bedarf verändern können.

4.2 Datenbankserver

- Sie befinden sich im Ordner *Webserver/database/*
- Öffnen Sie den Unterordner **cfg**/
- Setzen Sie in der Datei *databaseserver.json* in dem Feld *port* den Port und in dem Feld *ip* die IP-Adresse des Datenbankservers ein
- Geben Sie die IP-Adresse und den Port des Webservers in den Feldern *webserver.addr*, *webserver.ip* und *webserver.port* ein
- Danach geben Sie die für Ports die sie für die beiden Datenbanken verwenden wollen in den Feldern *db_room.port* und *db_veranstaltung.port* ein
- In der Datei *Webserver/database/util/test_data.json* befinden sich Testdaten die Sie bei Bedarf verändern können.

5 Nutzung

- Starten Sie Webserver indem Sie im Ordner *Webserver/server/* den Befehl *node server.js* ausführen.
- Starten Sie dann die Datenbanken mit den zuvor angegebenen Ports. Der Linux Befehl Lautet: *redis-server –port PORT*. Für Windows navigieren Sie in den Ordner in dem sich die Datei redis-server.exe befindet und führen Sie den Befehl: *. |redis-server –port PORT* aus.
- Wenn der Webserver und die Datenbanken ohne Fehler gestartet sind starten Sie nun den Datenbankserver indem Sie im Ordner *Webserver/database* den Befehl *node server.js* ausführen.

6 Modellierte Knotenpunkte

Die Folgenden Räume und markanten Knotenpunkte sind im System modelliert, und können als Standort verwendet werden.

6.1 Gebäude

- g12
- g13
- g22
- g23

6.2 Treppenhäuser

- t200a
- t200b
- t202
- t204
- t300b
- t300a
- t302
- t304

6.3 Aufzüge

- a32
- a33

6.4 Räume

- 2105
- 2106
- 2114
- 2115
- 2200
- 2201
- 2202
- 2203
- 2204
- 2205
- 2206
- 3103
- 3104
- 3105
- 3114
- 3200
- 3201
- 3202
- 3203
- 3204
- 3205
- 3206
- 3207
- 3208