Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Entwicklung einer Hotelbuchungssoftware

Development of hotel booking software

Alexander Falkenberg, Hasan Alhelal

Teamprojekt

Betreuer: Titel Vorname Nachname

Ort, DD.MM.YYYY

Vorwort

Ein Vorwort ist nicht unbedingt nötig. Falls Sie ein Vorwort schreiben, so ist dies der Platz, um z.B. die Firma vorzustellen, in der diese Arbeit entstanden ist, oder um den Personen zu danken, die in irgendeiner Form positiv zur Entstehung dieser Arbeit beigetragen haben.

Auf keinen Fall sollten Sie im Vorwort die Aufgabenstellung näher erläutern oder vertieft auf technische Sachverhalte eingehen.

Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

Abstract

The same in English.

Inhaltsverzeichnis

Abbildungsverzeichnis

1

Einleitung

1.1 Motivation

Diese Arbeit handelt von der Umsetzung eines Buchungssystems für ein Hotel. Ein solches System ist besonders wichtig um den allgemeinen Aufwand beim Verwalten von verfügbaren Zimmern deutlich zu verringern und der Fehleranfälligkeit beim Vergeben von eben diesen vorzubeugen. Außerdem kann diese Software in leicht abgeänderter Form auch problemlos für andere Anwendungen benutzt werden.

1.2 Ziele der Arbeit

Ziel dieser Arbeit ist es ein simples Buchungssystem zu schaffen dass sich ohne großen Arbeitsaufwand verwenden lässt. Es soll den Nutzern des Systems möglich sein ohne die Erstellung eines Nutzerprofils Buchungen tätigen und diese verwalten zu können. Die Informationen über jede Buchung werden den Nutzern anschließend per Email zugesandt.

2

Verwandte Arbeiten

Die Gliederung hängt natürlich vom Thema und von der Lösungsstrategie ab. Als nützliche Anhaltspunkte können die Entwicklungsstufen oder -schritte z.B. der Software-Entwicklung betrachtet werden. Nützliche Gesichtspunkte erhält und erkennt man, wenn man sich

- in die Rolle des Lesers oder
- in die Rolle des Entwicklers, der die Arbeit z.B. fortsetzen, ergänzen oder pflegen soll,

versetzt. In der Regel wird vorausgesetzt, dass die Leser einen fachlichen Hintergrund haben - z.B. Informatik studiert haben. D.h. nur in besonderen, abgesprochenen Fällen schreibt man in populärer Sprache, so dass auch Nicht-Fachleute die Ausarbeitung prinzipiell lesen und verstehen können.

Die äußere Gestaltung der Ausarbeitung hinsichtlich Abschnittformate, Abbildungen, mathematische Formeln usw. wird in Kapitel **??** kurz dargestellt.

3

Grundlagen

3.1 HTML, CSS, JS

3.1.1 HTML

HTML-Dokumente bilden das Grundgerüst einer Webanwendung und sind von daher essenziell. Ein solches Dokument besteht aus einzelnen, oft schon vorgefertigten Elementen die ineinander geschachtelt werden können. Diese können anschließend durch referenzierte „Stylesheets“ (im Folgenden anhand von Less) in ihrem Aussehen verändert oder durch JavaScript mit weiteren Funktionalitäten ausgestattet werden.

3.1.2 Less

In dieser Anwendung wird die „Stylesheet-Sprache“ Less verwendet, um die verschiedenen HTML Elemente zu gestalten. Innerhalb des Less-Codes kann normaler CSS-Code verwendet werden, bietet jedoch einige Vorteile im Vergleich zu dieser herkömmlichen Stylesheet-Sprache wie zum Beispiel Variablen, Funktionen und Verschachtelung. Für die endgültige Nutzung dieser Sprache wird eine Kompilierung des Codes zu CSS-Code benötigt. Less bietet außerdem die Möglichkeit den Code auf mehrere Dateien aufzuteilen und diese durch die Kompilierung zu einer CSS-Datei zu bündeln.

3.1.3 JavaScript

Die Programmiersprache JavaScript mit ihren etlichen verfügbaren Modulen kann dafür verwendet werden Webanwendungen dynamisch zu gestalten. Zentraler Bestandteil dafür ist die „DOM-Manipulation“, durch die dynamisch Elemente in einem Webdokument verändert, hinzugefügt und entfernt werden können. Außerdem kann mithilfe von JavaScript eine HTTP-Anfrage erstellt, versendet und auf diese reagiert werden um mit einer serverseitigen Anwendung kommunizieren zu können. Insbesondere in dieser Anwendung wird von dieser Möglichkeit Gebrauch gemacht indem „XMLHttpRequest“-Objekte mit allen dazugehörigen Informationen initialisiert werden. Innerhalb dieser Objekte kann angegeben werden wie auf eine Antwort reagiert werden soll. Eine solche Initialisierung kann wie folgt aussehen:

```
1       const request = new XMLHttpRequest();
2       request.addEventListener('load', () => {
3           console.log(request.response);
4       });

6       request.open('POST', '/order');
7       request.setRequestHeader('Accept', 'application/json');
8       request.setRequestHeader('Content-Type', 'application/json');
9       request.responseType = 'json';
10      request.send(data);
```

Three.js

Werden dreidimensionale Darstellungen von Objekten oder Räumen benötigt, kann das Three.js Modul verwendet werden. Zu jedem Three.js-Projekt gehören folgende drei Bestandteile:

1. scene (engl. für Szene)
2. camera (engl. für Kamera)
3. renderer (engl. für Renderer)

Alle Objekte die gerendert werden sollen müssen sich in einer Scene befinden. Das Kamera-Objekt ermöglicht, die mithilfe von Three.js erstellten Szenen, zu visualisieren. Es existieren verschiedene Arten von Kamera-Objekten wie z.B. die perspektivische Kamera, die sich wie das menschliche Auge verhält, also näher stehende Objekte größer erscheinen lässt, und die orthografische Kamera, die die Größe der Objekte nicht an die Entfernung anpasst. Der Renderer als Hauptkomponente von Three.js bekommt eine erstellte Szene und Kamera übergeben und erzeugt so die finale dreidimensionale Darstellung des gewünschten Objekts.

Um eine solche Darstellung möglichst realistisch wirken zu lassen müssen geeignete Bilder im „.hdr“-Format ausgewählt werden. Damit Kandidaten dieses Formats verwendet werden können wird ein zusätzliches Modul mit dem Namen „RGBE-Loader“benönigt.

Mit den sogenannten „Orbit Controls“ aus dem Three.js-Modul können Nutzer die Ausrichtung der Kamera in der gerenderten Szene per Mausbewegung anpassen und so beispielsweise dreidimensionale Räume erkunden.

3.2 Node.js

Die serverseitige Grundlage des Systems bildet die Laufzeitumgebung Node.js. Mit dieser ist es möglich JavaScript Programme abseits eines Browsers ausführen zu können. Mithilfe von „npm“, dem Paketmanager von Node.js, werden außerdem externe Pakete bzw. Module für bestimmte Funktionalitäten verwendet um die Implementierung zu vereinfachen.

3.2.1 Express

Das Express-Framework ermöglicht eine unkomplizierte Erstellung einer Webanwendung. So wird beispielsweise die Verarbeitung von HTTP-Anfragen mithilfe von spezifischen Methoden deutlich vereinfacht und macht das Erstellen einer API besonders effizient. Innerhalb der Buchungssoftware sorgt Express für die Bereitstellung eines Servers, der alle clientseitigen Dateien zur Verfügung stellt und auf HTTP-Anfragen zur Erzeugung oder zum Abrufen von Buchungen, reagiert. Besonders wichtig ist das Konzept der Routen. Das Definieren von Routen bestimmt wie auf eine Client-Anfrage an eine bestimmte URI des Servers und eine spezifische HTTP-Methode, geantwortet wird. Die Zuweisung einer Route kann wie folgt aussehen:

```
1       app.METHOD(PATH, HANDLER);
```

In dem Beispiel steht app exemplarisch für ein express-Objekt, METHOD für eine HTTP-Methode, PATH für den Serverpfad und HANDLER für die Funktion die ausgeführt wird wenn eine Client-Anfrage existiert, die mit der Route übereinstimmt. Die Funktion die als HANDLER angegeben wird hat außerdem Zugriff auf ein HTTP-Request und ein HTTP-Response Objekt. Über das Request-Objekt kann auf Informationen der HTTP-Anfrage zugegriffen und mit dem Response-Objekt auf die Anfrage geantwortet werden.

3.2.2 Nodemailer

Nodemailer ist ein „npm“-Paket, das dafür verwendet wird E-Mails mit Node.js zu versenden. Durch Angabe der Adresse von der die E-Mail verschickt werden soll und verschiedenen Optionen wie z.B. dem Betreff wird das Senden vereinfacht. Um eine E-Mail zu versenden muss im ersten Schritt ein „Transporter“zusammen mit dem verwendeten Mail Dienst und den Anmeldeinformationen der zu sendenen E-Mail Adresse, erzeugt werden. Anschließend werden die Mail-Optionen initialisiert und über die Methode „sendMail“des Transportes, die E-Mail versandt.

3.2.3 Formidable

Das Node.js-Modul Formidable dient zur serverseitigen Verarbeitung von Formulardaten, insbesondere von Formularen die hochgeladene Dateien beinhalten. Da das Hochladen von Dateien durch einen Server behandelt werden muss bieten sich Module wie Formildable besonders an. So können die hochgeladenen Dateien in das Dateisystem des Servers eingefügt und verändert werden. Es werden viele Einstellungsmöglichkeiten geboten. So kann z.B. die maximale Größe einer hochgeladenen Datei festgelegt werden. Außerdem wird es dadurch möglich diese Dateien beispielsweise an eine E-Mail anzuhängen.

3.3 MongoDB

Unter den dokumentenorientierten NoSQL-Datenbanken ist MongoDB die meist verwendete. Die wichtigsten Bestandteile einer MongoDB Datenbank sind Dokumente und Collections. Dokumente sind Datenstrukturen die sich aus einem oder mehreren Paaren, bestehend aus Feldern und dazugehörigen Werten, zusammenfügen. In ihrem Aufbau sind diese Dokumente identisch mit JSON Objekten. Collections hingegen sind Sammlungen von verschiedenen Dokumenten und sind vergleichbar mit Tabellen aus relationalen Datenbanken.

Damit eine Node.js-Anwendung mit einer MongoDB kommunizieren kann wird beispielsweise das „npm“-Paket „mongodb“benötigt. Hier können dann auf den einzelnen Collections, Methoden angwendet werden. Die wichtigsten Methoden für das zu erstellende System sind „find“, „insertOne“, „deleteOne“und „deleteMany“. Mithilfe von find werden alle Dokumente einer Collection geliefert, auf die ein in der Methode spezifiziertes Kriterium zutrifft. InsertOne ermöglicht es ein neues Dokument in eine Collection hinzuzufügen und deleteOne bzw. deleteMany ermöglicht das Löschen von Dokumenten aus einer Collection.

4

Konzept

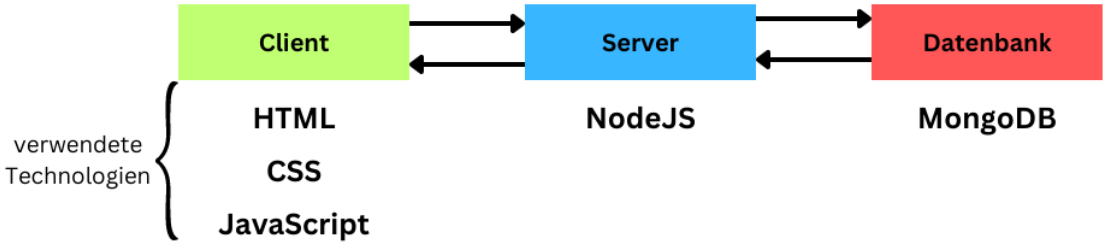

Abbildung 4.1: Grobe Architektur des Systems

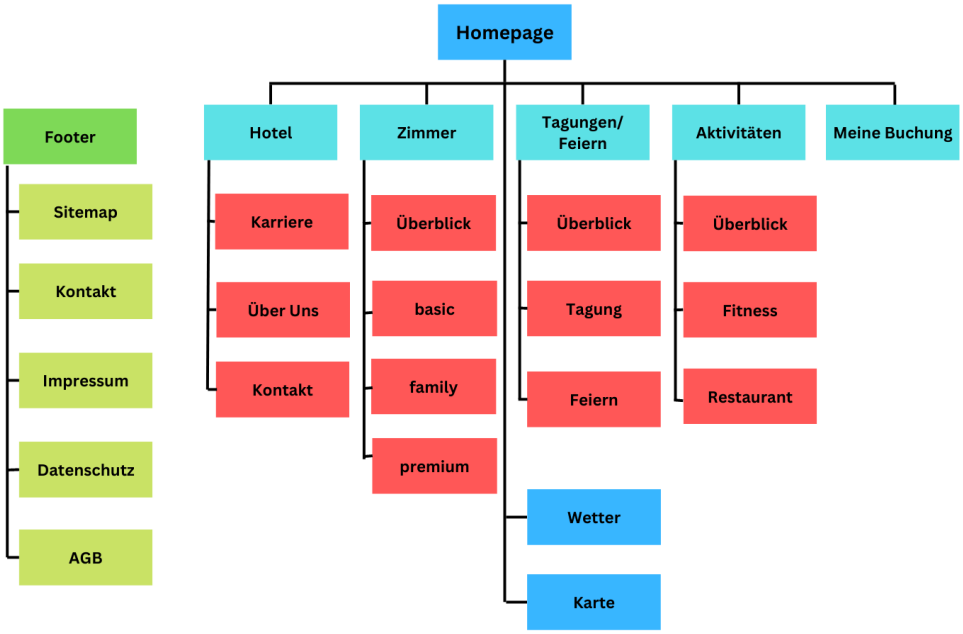

Abbildung 4.2: Sitemap

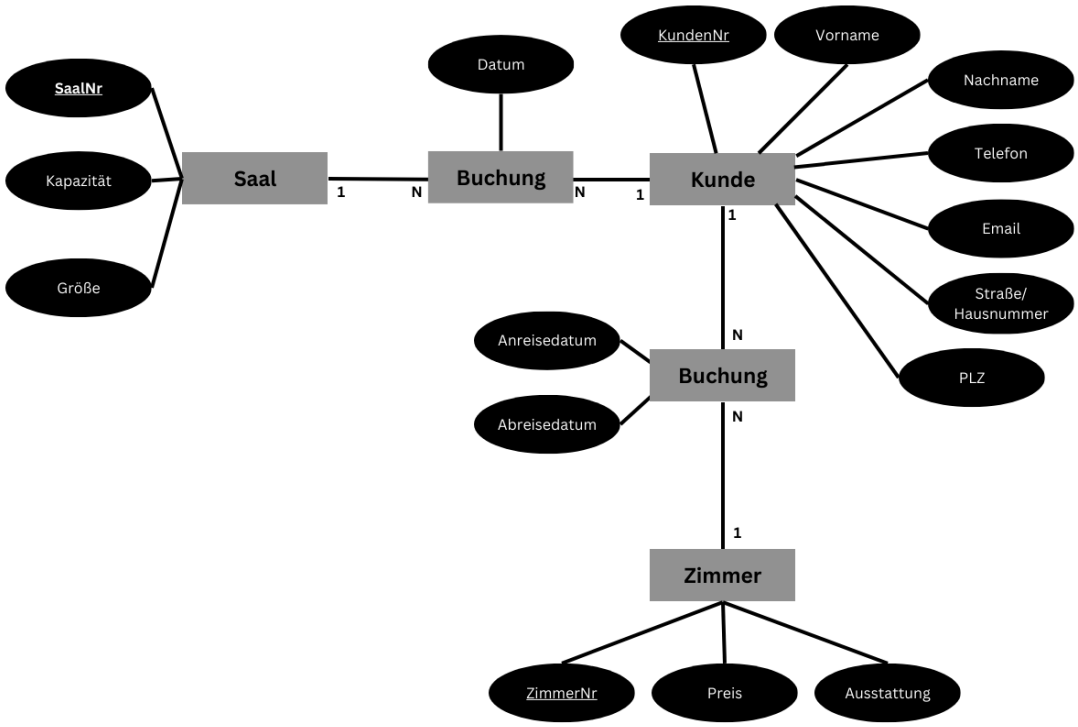

Abbildung 4.3: Datenbankentwurf

Algorithmus 1: Algorithmus um alle verfügbaren Räume zu erhalten

Data: rooms = alle Räume, reservations = alle Buchungen, arrival = Anreisedatum, departure = Abreisedatum
Result: available = verfügbare Räume
notAvailable ← [];
for *room of rooms* **do**
  for *reservation of reservations* **do**
    if $(room.id == reservation.room)$ && $!(departure < reservation.arrival \,||\, arrival > reservation.departure)$ **then**
      notAvailable.push(room);
      break;
return $available = rooms.filter(item \Rightarrow !notAvailable.includes(item))$

5

Realisierung

5.1 Buchung

6

Implementierung

6.1 Buchung

Serverseitig wird eine API erstellt, die die clientseitige Kommunikation mit dem Server ermöglicht. Dafür wird ein express-Router mit verschiedenen Routen benötigt, die unter anderem das Erstellen, Löschen, Verändern und Abfragen von Buchungen abwickeln. Ein solcher Router wird mit express-Methode „express.Router()“ erstellt. Für jede Route wird es außerdem von Nöten sein, Zugriff auf die MongoDB-Datenbank zu besitzen. Mithilfe des MongoDB-Moduls wird dafür ein „MongoClient“ erzeugt, der die Adresse der MongoDB-Datenbank übergeben bekommt.

6.1.1 Erstellen einer Buchung

Für die Erstellung einer Buchung wird dem Router eine Route hinzugefügt, die auf einen HTTP-POST mit dem Pfad „/order “ reagiert. Innerhalb der Funktion die die Anfrage behandelt werden zu aller erst alle für die Buchung benötigten Informationen aus dem Request-Objekt in Konstanten gespeichert. Zu den benötigten Informationen gehören: Vorname, Nachname, E-Mail-Adresse, Telefonnummer, Wohnort (mit Ort, Straße, Hausnummer und PLZ), Ankunftsdatum, Abreisedatum und die Art von Zimmern die gebucht werden sollen. Anschließend wird geprüft ob die Konstanten gültig sind. Sollte eine nicht zulässig sein wird eine HTTP-Antwort mit dem Code 400 an den Client gesendet. Bestehen jedoch alle Konstanten die Prüfung, wird mit der Verbindung des MongoClients fortgeführt. Der Aufruf der „connect“- und daraufhin der „db“-Methode (mit dem Namen der gesuchten Datenbank) des bereits erstellten MongoClients, liefert ein Datenbank-Objekt mit dem nun einzelne Collections abgerufen werden können.

Um einen neuen Nutzer anzulegen muss ein Collection-Objekt zur Anpassung der Collection die alle Nutzer enthält, mithilfe der „collection“-Methode des Datenbank-Objekts, erzeugt werden. Das Collection-Objekt erlaubt es mit der Methode „insertOne(elemToInsert)“ den neuen Nutzer mit allen dazugehörigen Informationen aus den gespeicherten Konstanten (Vorname, Nachname, Adresse, E-Mail, Telefonnummer) in die Collection einzufügen.

Im weiteren Verlauf werden die Collection mit allen Buchungen und die Collection mit allen Zimmern benötigt. Werden auf beiden Collections jeweils die Methoden „find().toArray()“ ohne Parameter aufgerufen, können alle Buchungen und Zimmer als Feld in jeweiligen Variablen abgelegt werden. Mithilfe des Algorithmus in 1 werden alle zu der angegebenen Ankunfts- und Abreisezeit verfügbaren Zimmer geliefert. Sind genug Zimmer mit den vom Client angegebenen Zimmertypen nicht vergeben werden für jeden zu buchenden Raum eine neue Buchung in die Buchungs-Collection eingefügt. Dabei besteht jede Buchung aus der ID des neuen Nutzers, der ID des verfügbaren Raums, des Ankunfts- und Abreisedatums.

Zuletzt wird per nodemailer eine Mail an die E-Mail-Adresse des Clients gesendet. Diese enthält die ID des angelegten Nutzers, welche später zu Änderung oder Stornierung der Buchung verwendet werden kann. Als Antwort erhält der Client eine HTTP-Antwort mit dem Status-Code 200 insofern keine Komplikationen auftreten.

6.1.2 Ändern einer Buchung

Um einem Nutzer zu ermöglichen die Daten einer bereits getätigten Buchung zu ändern, wird eine Route hinzugefügt, die auf einen HTTP-POST mit dem Pfad „/order/dates“ reagiert. Dafür müssen im ersten Schritt die Variablen mit den Werten aus dem HTTP-Body initialisiert und anschließend auf Gültigkeit überprüft werden. Die Werte bestehen aus der ID des Nutzers, das neue Ankunfts- und Abreisedatum. Sollte einer der Werte ungültig sein, wird eine HTTP-Antwort mit dem Status-Code 400 an den Client versendet. Sind sie jedoch gültig verbindet sich der MongoClient mit der Datenbank und fordert alle Buchungen an. Anschließend werden mithilfe der find-Methode einerseits alle Buchungen die mit der ID des Nutzers übereinstimmen und andererseits alle anderen Buchungen, in zwei verschiedenen Konstanten gespeichert. Aus den Raum-IDs der Buchungen des Nutzers werden alle Räume aus der Raum-Collection die diese IDs besitzen, in einem Feld gesammelt. Dadurch ist es möglich die in der Buchung gebuchten Zimmertypen in Variablen zu speichern, damit bei der Änderung der Daten auch wieder dieselben Zimmertypen gewählt werden. Wie schon in Kapitel 6.1.1 erklärt müssen auch hier alle Zimmer die während des neuen Ankunftsdatum bis zum neuen Abreisedatum verfügbar sind, ermittelt werden.

Sollte es nicht genügend Zimmer mit denselben Zimmertypen wie in der ursprünglichen Buchung geben, die während der neuen Daten verfügbar sind wird die bestehende Buchung nicht geändert und eine HTTP-Antwort mit dem Status-Code 400 an den Client gesendet. Sollte es jedoch genügend Zimmer geben wird die bisherigen Buchungen aus der „reservation“-Collection mithilfe der „deleteMany“-Methode gelöscht und anschließend neue Buchungen der gleichen Zimmertypen zu anderen Zeiten in diese Collection eingefügt. Ist der Ablauf problemlos wird der Status-Code 200 an den Client gesendet.

6.1.3 Anfragen einer Buchung

6.1.4 Löschen einer Buchung

6.2 Dreidimensionale Zimmer

7

Beispiele

8

Anwendungsszenarien

9

Zusammenfassung und Ausblick

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die z.B. Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

Literaturverzeichnis

CDK02. Coulouris, George, Jean Dollimore und Tim Kindberg: *Verteilte Systeme: Konzepte und Design*. Addison Wesley, München, 3. Auflage, 2002.

Che85. Cheriton, David R.: *Preliminary Thoughts on Problem-oriented Shared Memory: A Decentralized Approach to Distributed Systems*. ACM SIGOPS Operating Systems Review, 19(4):26–33, 1985.

CLR90. Cormen, Thomas H., Charles E. Leiserson und Ronald L. Rivest: *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1990.

GO03. Gottwald, Tim und Rainer Oechsle: *DisASTer (Distributed Algorithms Simulation Terrain): A Platform for the Implementation of Distributed Algorithms*. Diplomarbeit, Fachhochschule Trier, Fachbereich Informatik, 2003.

Mal97. Malte, Peter: *Replikation in Mobile Computing*. Seminar 31/1997, Universität Karlsruhe, Institut für Telematik, 1997.

Mos93. Mosberger, David: *Memory Consistency Models*. Technical Report 93/11, University of Arizona, Department of Computer Science, November 1993.

Nan03. Nannen, Volker: *The Paradox of Overfitting*. Master-Abschlussarbeit, Rijksuniversiteit Groningen, 2003.

Wik. *Wikipedia: Programmausdruck*. `https://de.wikipedia.org/wiki/Programmausdruck`. Abgerufen am 13.10.2021.

A

Selbstständigkeitserklärung

☐ Diese Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

☐ Diese Arbeit wurde als Gruppenarbeit angefertigt. Meinen Anteil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser:

Meine eigene Leistung ist:

Datum

Unterschrift der Kandidatin/des Kandidaten