



Ein Produkt von:

Kajendran Pulendran, Laurin Widerkehr, Rahel Frei

Modul 151

Datenbanken in Webauftritt einbinden

bei
Herr Walter Eiden, TBZ

Inhalt

1	Projektbeschreibung	4
2	Produktbeschreibung	4
3	MoSCoW (Must, Should, Could, Won't)	5
4	Vorgehen	6
5	Kompetenzen	6
5.1	DB Modelle (Big Data, historische DB).....	6
5.1.1	Relationale Daten Banken	6
5.1.2	Unterschiede.....	6
5.1.3	No SQL.....	6
5.1.4	SQL	6
5.1.5	Transaktionssichere Abfragen	6
5.1.6	Beispiele	7
5.1.7	Leftjoin.....	7
5.1.8	Rightjoin.....	7
5.1.9	Innerjoin.....	8
5.2	Architaturen	8
5.2.1	3 Tier Architektur	8
5.2.2	4 Tier Architektur	8
5.2.3	5 Tier Architektur	8
5.2.4	Vorteile	9
5.2.5	Nachteile.....	9
5.3	PHP 7.0	9
5.3.1	Geschichte	9
5.3.2	Syntax.....	9
5.3.3	PHP 7.0 vs PHP 5.6.....	10
5.4	Webservices.....	10
5.4.1	Was ist ein Webservice?	10
5.4.2	Merkmale	10
5.4.3	Abgrenzung des Webservices zur Webanwendung	11
5.4.4	Architektur	11
5.4.5	Vorteile	11
5.4.6	Nachteile.....	11
5.4.7	Beispiele	11
5.4.8	Beispiel	12
5.5	Datenbankhandling.....	12

5.5.1	Stored Procedures	12
5.5.2	Prepared Statements	12
5.5.3	Database Session	13
5.5.4	Transaction	13
5.6	TLS-/SSL-Verschlüsselung	13
6	Teamarbeit (Warum die Arbeit gut war, Konfliktbehandlung, Stärken und Schwächen) 14	
6.1.1	Konflikte.....	14
6.1.2	Warum die Arbeit gut war.....	14
6.1.3	Stärken und Schwächen	14
7	Reflexion	14



1 Projektbeschreibung

Projekttitlel:	SCHOOLTOOL
Modul:	Modul 151
Lehrperson:	Walter Eiden
Entwickler:	Kajendran Pulendran, Laurin Wiederkehr, Rahel Frei

2 Produktbeschreibung

SCHOOLTOOL ist eine WEB-Applikation, welche dazu dient, Prüfungen zu erstellen und Noten einzutragen. Dabei haben die Lehrpersonen sowie auch Schüler eine Userfreundliche Umgebung.

Die Applikation beinhaltet eine Person und Klassenübersicht. Berechtigte Personen, können Klassen, Schüler und Lehrer hinzufügen.

Lehrpersonen können Prüfungen online erstellen und auswerten. Beim Verlassen des Tabs werden die Schüler für eine Zeit gesperrt und können nicht mehr an der Prüfung weiterarbeiten.

3 MoSCoW (Must, Should, Could, Won't)

Must	Should	Could
Create a Person	Scoring-System for Exam	Free Text Validator
Edit a Person	Person Overview marks	Persents overview
Delete a Person		Diary
Create a School-Class		
Edit a School-Class		
Delete a School-Class		
Asigne people to a School-Class		
Create an Exam		
Edit an Exam		
Delete an Exam		
Asigne an Exam to a School Class		
Create a Question		
Edit a Question		
Delete a Question		
Create an Option		
Edit an Option		
Delete an Option		
Access control based on roles		

4 Vorgehen

Jeden Dienstagmorgen begann der Tag so, dass wir uns zusammensetzten und in einem kurzen Scrum-Meeting zusammenfassten was wir in der letzten Woche gemacht haben und was wir heute machen wollen. Danach setzten wir uns alle an unsere Tasks und arbeiteten weiter, wenn jemand ein Problem hatte versuchten wir ihn gemeinsam zu unterstützen. Laurin war dabei eine wichtige Person, da er der erfahrenste von uns war und uns so aktiv unterstützte.

5 Kompetenzen

5.1 DB Modelle (Big Data, historische DB)

5.1.1 Relationale Daten Banken

Relationale Datenbanken ist das am weitesten verbreitete Datenbankmodell. Es setzt auf das relationale Datenbankmodell, das auf der Speicherung von Informationen in verschiedenen Tabellen basiert, die untereinander über Beziehungen (Relationen) verknüpft sind.

5.1.2 Unterschiede

5.1.3 No SQL

- ...Technologien können mit den eingangs erwähnten Herausforderungen besser umgehen.
- ...ermöglicht eine Skalierbarkeit durch Scale-out - also durch das Hinzufügen von Servern. So lassen sich auch große Datenmengen relativ günstig verwalten.
- ...kann man die Datenstrukturen effizient bearbeiten.
- In dokumentenorientierten Datenbanken lassen sich auch wenig strukturierte Daten speichern. Vorteil bei komplexen Datenstrukturen.
- In Graphendatenbanken können vor allem Graphen gespeichert werden, die sich mit dem relationalen Modell nur schwer bearbeiten lassen.

5.1.4 SQL

- ...war mit dem Anspruch angetreten, dem Endbenutzer auf natürliche Weise die Suche in Datenbanken zu ermöglichen.
- ...bot eine mengenorientierte Schnittstelle geboten, die die Beschränkung früherer Abfragesprachen, die nur einen Satz pro Datenbank-Zugriff verarbeiten konnten, aufhob.
- ...ist eine deskriptive Sprache, der Programmierer gibt also nur noch an, welche Informationen er will, aber nicht, wie auf die Daten zugegriffen werden soll.
- Zudem ist die völlige physische Datenunabhängigkeit gegeben, so dass physische Datenbank-Reorganisationen (zum Beispiel ein neuer Index) vorgenommen werden können, ohne dass Anwenderprogramme angepasst werden müssen.

5.1.5 Transaktionssichere Abfragen

Eine Transaktion bezeichnet eine Menge von Datenbankänderungen, die zusammen ausgeführt werden (müssen). So ist beispielsweise die Buchung (als Transaktion) eines Geldbetrags durch zwei atomare Datenbankoperationen „Abbuchung des Geldbetrages von Konto A“ und „Buchung des Geldbetrages auf Konto B“ gekennzeichnet. Kann die vollständige Abarbeitung der elementaren Datenbankoperationen der Transaktion nicht durchgeführt werden (z. B. aufgrund eines Fehlers), müssen alle durchgeführten Änderungen an dem Datenbestand auf

den Ausgangszustand zurückgesetzt werden. Der Vorgang, der alle Änderungen einer Transaktion zurücksetzt, wird als Rollback bezeichnet. Der Begriff Commit bezeichnet das Ausführen einer Transaktion. Transaktionen sind eine Möglichkeit, die Konsistenz des Datenbestandes zu sichern. Im Beispiel der doppelten Kontenführung wird durch das Verhindern von ungültigen Teilbuchungen eine ausgeglichene Kontobilanz gewährleistet.

Datenbanken erlauben es zum Teil, bestimmte Befehle außerhalb einer Transaktion auszuführen. Darunter fällt insbesondere das Laden von Daten in Tabellen oder das Exportieren von Daten mittels Utilities. Manche DBMS erlauben das temporäre Abschalten der Transaktionslogik sowie einiger Kontrollen zur Erhöhung der Verarbeitungsgeschwindigkeit. Dies muss allerdings meist durch einen expliziten Befehl erzwungen werden, um ein versehentliches Ändern von Daten außerhalb einer Transaktion zu vermeiden. Solche Änderungen können, falls eine Datenbankwiederherstellung erforderlich ist, zu schweren Problemen oder gar Datenverlusten führen. Eine Transaktion wird mit der SQL-Anweisung Commit beendet. Alle Änderungen der Transaktion werden persistent gemacht, und das DBMS stellt durch geeignete (interne) Mittel (z. B. Logging) sicher, dass diese Änderungen nicht verloren gehen. Mit dem Befehl Rollback wird eine Transaktion ebenfalls beendet, es werden jedoch alle Änderungen seit Beginn der Transaktion rückgängig gemacht. Das heißt, der Zustand des Systems (in Bezug auf die Änderungen der Transaktion) ist der gleiche wie vor der Transaktion.

5.1.6 Beispiele

NoSQL Querys werden mit Hilfe von JSON-Objects erstellt.

```
{
  object: String,
  q: Expression,
  fields: Array of String,
  groupBy: Array of String,
  aggregation: Object mapping fields to aggregate functions
}
```

5.1.7 Leftjoin

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

5.1.8 Rightjoin

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

5.1.9 Innerjoin

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

5.2 Architekturen

5.2.1 3 Tier Architektur

Die 3 Tier Architektur besteht im Gegensatz zur 5 Tier Architektur nur aus den 3 Layern: GUI (Graphical User Interface), Logic und Data.

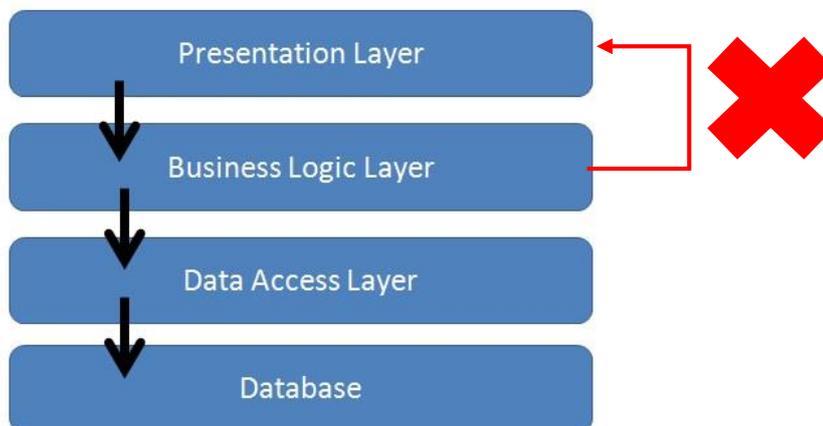
GUI: Hier werden alle Steuerelemente und grafischen Komponenten platziert. Zudem werden die Event-Handler (Button-Click, Mouse-Over usw.) definiert. In den Events werden jedoch selber keine Berechnungen vorgenommen. Diese rufen lediglich die Funktionen aus dem Logik-Layer auf.

Logic: Hier werden sämtliche Methoden zu Berechnungen und umgesetzte Algorithmen implementiert. Jedoch werden keine Datenquellen direkt angesteuert. Braucht man Daten aus einer Datenbank, oder von Webservices usw. wird die Datenschicht angesteuert.

Data Layer: In der Datenschicht werden alle Methoden verankert die Daten direkt aus einer Datenquelle lesen. Dies könnten Methoden sein die mittels SQL Datenbank-Abfragen machen, Methoden welche APIs von Webservices ansteuern oder sogar Methoden welche ganz simple Daten aus Dateien (Excel csv, Textdatei usw.) lesen.

5.2.2 4 Tier Architektur

Bei der 4Tier Architektur teilt man den Data Layer in den Data Access Layer- und Database auf.



5.2.3 5 Tier Architektur

Bei der 5-Tier Architektur wird so aufgebaut:

Layer	Beschreibung
Presentation	Ist eine Aufteilung des UI Layers, welche Client-seitige Scriptausführung, wie z.B. JavaScript
UI (User Interface)	Ist die Optische Darstellung, welche der Benutzer sieht
Buisness Logic	Die gesamte serverseitige Logik der Applikation befindet in diesem Layer
Data Acces	Schnittstelle zwischen der Applikation und Datenbank Storage Layer
Data Storage	Physische Datenbank, in welchem INSERT, DELETE etc. gemacht werden

5.2.4 Vorteile

Durch eine Schichtenarchitektur wird die Komplexität der Abhängigkeiten in einem System vereinfacht. Einzelne Schichten sind leicht veränderbar, bei der Einhaltung der Schichtenstellung/Interfaces. Bei Veränderungen/Interfaces sind nur die beiden angrenzenden Schichten betroffen. Schichtenarchitekturen kapseln Maschinenabhängigkeit und sind dadurch leicht portierbar. Nur die innerste Schicht muss neu implementiert werden.

5.2.5 Nachteile

Es ist schwierig Systeme sauber in Schichten zu Strukturieren. Wenn äussere Schichte Dienste der inneren benötigen, wird leicht die einfache Abhängigkeit von der nächst unteren Schicht zerstört. Es kann Perfomance Probleme geben, weil mit dem Zugriff auf die Dienste eine Schicht immer ein gewisser Overhead verbunden ist.

5.3 PHP 7.0

5.3.1 Geschichte

PHP ist eine Skriptsprache, die erstmals in 1995 vorgestellt wurde. Die erste Version von PHP wurde in 2004 veröffentlicht. PHP 5 brachte hierbei nicht nur performance-technische Verbesserungen, sondern auch einige Erweiterungen. PHP 5 hat jedoch auch innerhalb seiner Hauptversion einige Meilensteine hingelegt. Eine Version mit vielen Änderungen und vor allem auch Erweiterungen war unter anderem die Version 5.3. Bis heute wird auf einer Vielzahl an Servern PHP 5 eingesetzt. Die aktuelle Version von PHP 5 ist PHP 5.6. Für PHP 5.6 ist der aktive Support seit Anfang 2017 eingestellt, doch es werden noch Sicherheitsupdates in unregelmäßigen Abständen veröffentlicht.

Als Nachfolger von PHP 5 war PHP 7. Die Unterstützung von PHP 7 bei den Hosting - Providern ist in letzter Zeit stetig gestiegen und in der Zwischenzeit sehr hoch.

Einige Provider bieten jedoch die Möglichkeit, zwischen PHP 5 und PHP 7 zu wählen.

5.3.2 Syntax

Um den Grundsatz einer Syntax zu verstehen hilft es ungemein, die Bedeutung von "Programmiersprache" zu verstehen. Das Wort "Sprache" beim programmieren

wurde aus dem Grund gewählt, da es sehr viele Ähnlichkeiten mit der menschlichen Sprache gibt. Das trifft vor allem auf das Geschriebene zu. Wir Menschen führen einen Dialog, den man beim Programmieren auch mit dem Computer führt. Es gibt die Rechtschreibung sowie die Grammatik. Es gibt Ausdrücke, Fragen, Antworten und noch vieles mehr. Es gibt aber genau zwei Unterschiede, die man niemals vergessen sollte. Zum einen verzeiht der Computer keine Fehler und zum anderen ist er berechenbar.

Die Syntax bildet also etwas in der Programmiersprache ab, was in den menschlichen Sprachen als Grammatik bezeichnet wird. Diese Grammatik ist für die Interpreter/Compiler in der Programmierung unfassbar wichtig, da es den verwertenden Programmen ermöglicht, den Code besser in seine Bestandteile zu zerlegen. Für uns Programmierer ist die Syntax zudem ein gutes Mittel, die Übersicht über den Code zu behalten.

Die PHP Syntax ähnelt recht stark der Syntax von C oder Java. Die Unterschiede liegen vor allem im begrenzten Sprachumfang oder der einen oder anderen Designentscheidung.

```
1 // Eine Variable beginnt immer mit dem $ Zeichen
2 $variable;
3
4 // Eine Funktion wird ohne Vorzeichen verwendet, muss aber mit einem Satz
5 // runder Klammern abgeschlossen werden. Diese können Parameter enthalten.
6 getdate();
7
8 // Konstanten sind ebenfalls vorzeichenlos, benötigen keine Nachzeichen, können
9 // aber auch nur durch eine Funktion erzeugt werden.
10 TIMESTAMP_REFERENCE
```

5.3.3 PHP 7.0 vs PHP 5.6

- MYSQL Abfragen können nicht mehr getätigt werden
- Neu kann man MYSQLI verwenden, um Datenbank Abfragen zu machen
- Alle fatalen Fehler sind in PHP 7.0 Exceptions und können nun abgefragt werden, ohne für den sofortigen Abbruch des Programms zu sorgen
- Anonyme Klassen: Nützlich, wenn einfache Objekte zum einmaligen Gebrauch erzeugt werden müssen
- Return Type Declarations, neu können die Datentypen der zurückgegebenen Werte definiert werden

5.4 Webservices

5.4.1 Was ist ein Webservice?

Ein Webservice ist ein Dienst, der über ein Netzwerk angesprochen werden kann. Über den Webservice kommunizieren Maschinen oder Anwendungen miteinander. Die Services haben keine Benutzeroberfläche für Menschen. Für die Realisierung kommen serviceorientierte Architekturen (SOA) zum Einsatz.

Hierbei kommen definierte Schnittstellen, Protokolle und Standards zum Einsatz. Die Webdienste selbst können in den unterschiedlichsten Programmiersprachen geschrieben sein und verschiedene Hardware-Plattformen verwenden.

5.4.2 Merkmale

- der Webservice stellt seine Dienste und Funktionen über ein Netzwerk zur Verfügung

- es kommunizieren Anwendungen oder Maschinen, nicht Menschen mit dem Webservice
- die Realisierung des Webservices basiert auf der serviceorientierten Architektur
- es kommen definierte Schnittstellen, Protokolle und Standards zum Einsatz
- die Kommunikation mit dem Webservice ist automatisiert
- der Webservice kann mit unterschiedlichen Programmiersprachen und auf verschiedenen Hardwareplattformen realisiert sein
- die Kommunikationspartner authentifizieren sich gegenseitig
- übertragene Daten lassen sich per Verschlüsselung sichern

5.4.3 Abgrenzung des Webservices zur Webanwendung

Die Begriffe Webservice und Webanwendung werden oft analog verwendet, obwohl es sich grundsätzlich um verschiedene Funktionalitäten handelt. Webservices sind nicht für die Benutzung von Nutzern vorgesehen. Webservices dienen der reinen Computer-zu-Computer-Kommunikation. Die Webanwendung ist hingegen für den User entwickelt worden. Der User kommuniziert per Netzwerk direkt über die Benutzerschnittstelle mit der Anwendung. Im Hintergrund kann die Webanwendung Webservices nutzen, um dem Anwender die gewünschten Ergebnisse zu liefern.

5.4.4 Architektur

- **UDDI** dient als Verzeichnisdienst zur Registrierung von Webservices. Es ermöglicht das dynamische Finden des Webservices durch den Konsumenten. Allerdings wird **UDDI** nur in eher kleineren Firmennetzwerken verwendet.
- **WSDL** zur Beschreibung der unterstützten Methoden und deren Parametern für den Programmierer.
- **SOAP(oder XML-RPC)** zur Kommunikation. Der eigentliche Aufruf wird hier gestartet.

5.4.5 Vorteile

- Webservice verwenden Standards und ermöglichen die Kommunikation von Systemen auf unterschiedlichen Plattformen.
- Da überwiegend offene Standards genutzt werden, entstehen in der Regel kaum Software-Lizenzkosten zur Bereitstellung der Dienste.
- Das Internet macht die Services von überall aus nutzbar und erschließt ein riesiges Feld an Anwendungsmöglichkeiten.
- Die komplette Architektur der Webservices ist sehr flexibel und gestattet es, einzelne Protokolle und Standards variabel einzusetzen.

5.4.6 Nachteile

- Für die per Internet erreichbaren Webservices entsteht ein gewisser Aufwand zur Sicherung der Kommunikation und des Dienstangebots.
- Sichere Verschlüsselungsmechanismen erfordern eine hohe Rechenleistung.
- Der Schutz vor Angriffen aus dem Internet ist meist nur durch den Einsatz vorgeschalteter Systeme wie Firewalls möglich.

5.4.7 Beispiele

Webservices sind in vielen Bereichen zu finden. Für Onlineshops stellen sie beispielsweise Funktionen zur Verfügung, über die sich die Gültigkeit von Kreditkarten prüfen oder Versandkosten ermitteln lassen. Paketservices bieten für Shop Anwendungen Dienste zur Paketverfolgung an.

Google hat ein großes Angebot an Webservices. Unter anderem können fremde Anwendungen die Google-Suche oder Google-Maps per Webdienst ansprechen.

Weitere Beispiele sind Computerreservierungssysteme von Fluggesellschaften. Sie stellen Funktionen zur Verfügung, über die sich Flugdaten oder Flugverfügbarkeiten abrufen und Flüge buchen lassen. Viele Preisvergleichsportale nutzen ebenfalls Webservices zur Ermittlung von Preisen, die sie anschließend auf Vergleichsseiten dem Anwender aufbereitet darstellen.

5.4.8 Beispiel

Wenn man zum Beispiel alle Artikel anfragen möchte, kann man das mittels:

```
GET https://<your-domain>/api/articles
```

Auf diese Anfrage erhält man, wenn man korrekt authentisiert ist, alle Artikel im JSON-Format zurück.

```
[
  {
    "articleId": "fbc9f7b6-883a-4afd-8ac7-19472c6aeaf3",
    "created": "Tue Sep 24 09:40:39 CEST 2019",
    "lastEdited": "Tue Sep 24 09:43:29 CEST 2019",
    "author": "Ivo Cavelti",
    "title": "Content",
    "likes": 0,
    "dislikes": 0,
    "articleContent": "RandomContent"
  }
]
```

Wir haben uns hier für das JSON-Format entschieden, da dieses als «Mensch» sehr gut lesbar ist und es schon diverse Java-Libraries gibt, welche Java-Objecte in JSON umwandeln und umgekehrt.

SpringBoot wandelt schon automatisch alle gesendeten Java-Objekte in JSON um. Deshalb sieht die obenstehende Methode im Java-Code folgendermassen aus:

```
@GetMapping
public List<ArticleTO> getArticles() {
    log.info("Returning articles");
    return articleService.getArticles();
}
```

5.5 Datenbankhandling

5.5.1 Stored Procedures

Ein Stored Procedure ist ein Set aus SQL-Abfragen, welche unter einem Namen zusammengefasst werden. Diese werden ebenfalls in der Datenbank eingespeichert, damit sie an verschiedensten Orten wiederverwendet werden können. Stored Procedures bringen einen erheblichen Vorteil in der Sicherheit, da es viel schwieriger ist die Integrität der Daten zu verletzen, wenn der Nutzer nur bereits vorbereitete Prozeduren ausführen kann. Im optimalfall werden dem Datenbanknutzer nur die Rechte gegeben, die Stored Procedures auszuführen. So kann sichergestellt werden, dass er nur die Daten bearbeiten kann, welche explizit in den Stored Procedures genannt werden.

5.5.2 Prepared Statements

Bei Prepared Statements, wird die SQL-Datenbankabfrage schon geschrieben, allerdings werden alle Parameter weggelassen. Diese werden dann, wenn die Abfrage ausgeführt wird, angegeben. Somit kann

man SQL-Injection Attacken verhindern. Stored Procedures sehen in PHP folgendermassen aus:

```
$stmt = $dbh->prepare("SELECT user, password FROM tbl_user  
WHERE  
(user=:user)");  
$stmt->bindParam(':user', $user);  
$user = 'Alice';  
$stmt->execute();
```

5.5.3 Database Session (beispiel)

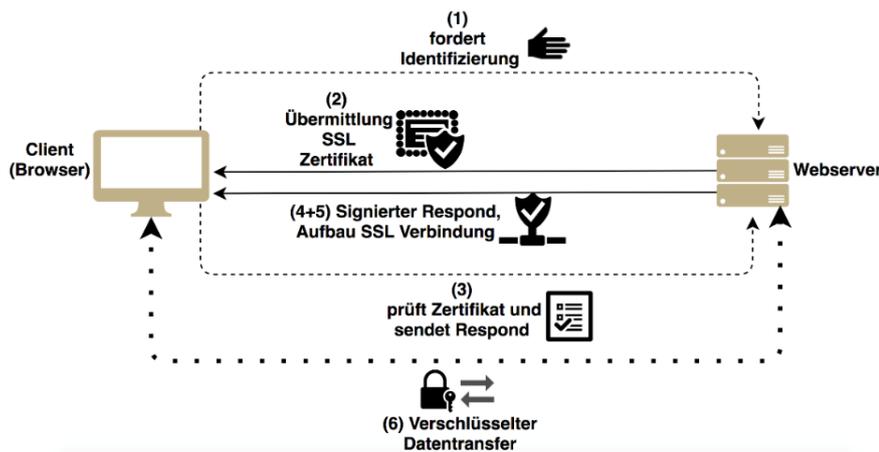
Die Datenbank Session ist quasi der Behälter über einer Transaction. Sie beinhaltet alle Arbeit, die während der Transaktion geleistet wurde und sorgt dafür, das alles reibungslos commitet oder wieder zurückgesetzt werden kann. Eine Session wird meistens für eine oder wenige Abfragen verwendet und wenn diese sicher verarbeitet sind wieder verworfen. Eine Session besitzt nur eine kurze Laufzeit, deshalb kann sie nur kurz offengehalten werden und muss am besten bei jedem neuen Anfragenset neu generiert werden. Zur Abhilfe gibt es dafür «Session Pools» welche einem automatisch Sessions zuteilen und diese zu verwalten.

5.5.4 Transaction

Eine Transaction wird vor allem dann verwendet, wenn eine Datenbank Thread-Safe sein soll. Sie hat den Vorteil, dass nicht sofort die Daten in die Datenbank geschrieben werden. Sie stellt sicher, das alle ihre Statements sicher ausgeführt wurden und macht dann einen «Commit». Falls es einen Fehler gab, macht sie einen «Rollback» welches alles wieder auf den Ausgangszustand zurücksetzt. So können mehrere «Inserts» ausgeführt werden und man kann sicherstellen, das alle Daten korrekt eingetragen sind. Wenn nun jetzt zum Beispiel verschiedene Threads auf einen SQL-Datensatz zugreifen, wird sichergestellt das diese in der richtigen Reihenfolge ausgeführt werden, damit sie sich nicht gegenseitig beeinflussen.

5.6 TLS-/SSL-Verschlüsselung

SSL (Secure Sockets Layer) ist der Vorgänger von TLS (Transport Layer Security). Es ist ein Sicherheitsprotokoll zum Schutze des Datentransfers via eines Netzwerkes. SSL/ TLS basieren auf RSA/ ECDSA kryptographischen Verfahren. Das heisst SSL/ TLS verwendet asymmetrische Verschlüsselung. Eine Webseite gibt ihren Public Key, in Form eines Zertifikates, heraus. Ein Client kann mit Hilfe des Zertifikates verifizieren, dass die erhaltene Webseite nicht im Transport verändert wurde (man in the middle). Weiter wird der gesamte Datenverkehr verschlüsselt, so dass man auch sensitive Daten übermitteln kann. Um den verschlüsselten Datenverkehr aufzubauen muss zuerst ein Key-Exchange erfolgen. Dazu wird meistens der Diffie-Hellman Key-Exchange verwendet. Er ermöglicht es das Server und Client über ein Netzwerk ein gemeinsamen Key generieren ohne diesen überdas Netzwerk senden zu müssen. Wenn eine Webseite SSL/ TLS verwendet stellt sie damit Authentizität, Integrität und Verschlüsselung sicher.



6 Teamarbeit

Da Laurin die meiste Erfahrung in Angular hat, hat er dem Rest vom Team Kajendran und Rahel die Welt von Angular nähergebracht. Zuerst erarbeiteten wir gemeinsam alle Diagramme und einige Lösungsansätze, bevor wir uns dann an die gezielte Aufgabenteilung gemacht haben. Kajendran ging für 5 Wochen auf England und so haben wir ihm gewisse Aufgaben für da zugeteilt, welche er relativ unabhängig von uns lösen konnte. Die Teamarbeit war somit, trotz dem fehlenden Teammitglied sehr gut und wir konnten uns gut einigen und absprechen.

6.1.1 Konflikte

Unsere grössten Diskussionen fanden ganz am Anfang statt, da jeder eine eigene Vorstellung von dem Projekt hatte. Wir haben das ausführlich diskutiert und beruhigten uns gegenseitig, wenn eine Diskussion zu hitzig wurde.

6.1.2 Warum die Arbeit gut war

Teamwork makes the Dreamwork, kann ich nur sagen. Wir haben schon einige andere Projekte zusammen gemacht und kennen uns dementsprechend sehr gut. In anderen Modulen haben vor allem Laurin und ich einige in unserer Klasse unterstützt und so hätten wir auch kein grosses Problem gehabt mit neuen Teammitgliedern. Wir sind auch privat gute Kollegen, was die Zusammenarbeit sicher vereinfacht hat. Nach intensiven Codingsequenzen, machten wir kurze Pausen und redeten ein bisschen, was uns wieder auflockerte.

6.1.3 Stärken und Schwächen

Unsere Stärke lag sicherlich darin, dass wir uns alle bereits kannten und so wussten, was die andere Person gerne machen würde. Eine andere Stärke ist sicher, dass wir alle sehr motiviert sind und immer neues lernen wollen. Eine klare Schwäche wäre dann sicher, dass wir uns auch mal überfordern und dann lange Nachtschichten machen. Auch verfangen wir uns schnell in, zwar spannenden, aber auch zeitraubenden Diskussionen.

7 Reflexion

Da wir uns ein relativ komplexes und anspruchsvolles Projekt ausgesucht haben, brauchte die ganze Planung ein bisschen mehr Zeit als gedacht. Am Anfang haben wir die ganze Planung gemeinsam gemacht und ausdiskutiert, sodass wir schlussendlich jedem die Tasks zuweisen konnten, welche die jeweilige Person gerne machen wollte.



Es gab einige Diskussionen, wer jetzt was in welcher Reihenfolge machen soll/will, aber unser Teamleiter Laurin Wiederkehr hat die Gemüter immer wieder erfolgreich beruhigt. Nach der intensiven Planungszeit kam dann endlich alles ins Rollen und wir kamen sehr gut voran. Jeder hatte seine Aufgaben und so arbeiteten wir auch zuhause noch an dem Projekt oder wie Herr Pulendran es gemacht hat, von Kanada aus. Die Schwierigkeit war es natürlich, mit Herrn Pulendran in Kontakt zu bleiben, da wir ja in verschiedenen Ländern und verschiedenen Zeitzone waren, gelöst haben wir das mittels WhatsApp. Regelmässig hielten wir über den Gruppenchat kontakt. Trotz den anfänglichen Startschwierigkeiten lief das Projekt sehr flüssig ab und wir konnten uns gut ergänzen.