

Entwicklung eines interaktiven Kundeninformationssystems für eine Bäckerei

STUDIENARBEIT

Gregory Seibert

22. Februar 2019

Matrikelnummer, Kurs

9234269, TINF-16-ITA

Betreuer

Enrico Keil

Name, Vorname: Seibert, Gregory

Matrikelnummer: 9234269

Studiengang/Kurs: TINF-16-ITA

Titel der Arbeit: Entwicklung eines interaktiven Kundeninformationssystems für eine Bäckerei

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Falls sowohl eine gedruckte als auch elektronische Fassung abgegeben wurde, versichere ich zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Unterschrift

Abstract

Inhaltsverzeichnis

Glossar	V
Akronyme	VI
Abbildungsverzeichnis	VII
1 Einleitung	1
1.1 Stand der Technik	1
1.2 Zielsetzung	1
2 Theoretische Grundlagen	2
2.1 REST API	2
2.2 JSON	4
2.3 Dependency Injection	4
2.4 Spring Framework	5
2.5 PostgreSQL	6
2.6 MVC Pattern	7
2.7 Swift	8
3 Entwurf des Systems	9
4 Entwicklung des Backends	10
5 Entwicklung der iOS App	11
6 Fazit	12
Literatur	VII

Glossar

Backend	Das Backend verwaltet die Daten.
DELETE	Das Löschen eines bestehenden Datenobjektes.
Frontend	Das Frontend stellt die notwendigen Informationen dar und ist zur Interaktion mit dem Benutzer.
GET	Das Abfragen eines bestehenden Datenobjektes.
POST	Das Erstellen eines neuen Datenobjektes.
PUT	Das Aktualisieren eines bestehenden Datenobjektes.

.

Akronyme

ACID	Atomicity, Consistency, Isolation and Durability.
API	Application Programming Interface.
HTTP	Hypertext Transfer Protocol.
JSON	JavaScript Object Notation.
MVC	Model View Controller.
MVCC	Multi-Version Concurrency Control.
REST	Representational State Transfer.

Abbildungsverzeichnis

2.1	Das Programm Postman wird genutzt, um REST Schnittstellen zu testen .	3
2.2	Der Aufbau des MVC Patterns [Cho]	7

1 Einleitung

1.1 Stand der Technik

1.2 Zielsetzung

2 Theoretische Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen der Technologien, die für das Backend und die iOS App notwendig sind, vermittelt.

2.1 REST API

Representational State Transfer (REST) Application Programming Interfaces (APIs) spielen mittlerweile eine große Rolle in der Entwicklung und finden bereits in zahlreichen Projekten namhafter Unternehmen Anwendung. Diese APIs sind sogenannte Programmierschnittstellen über das Hypertext Transfer Protocol (HTTP). Mit deren Hilfe lassen sich Daten per GET, PUT, POST und DELETE abrufen und verwalten.

Der Sinn dahinter ist, dass man das Frontend, also das, was der Benutzer sieht, und das Backend, die Verarbeitung und Verwaltung der Daten, trennt. Dieses Vorgehen ist nicht nur sicherer, sondern auch im späteren Verlauf einfacher anzupassen, da man nun in modularen Gebieten des Projektes arbeiten kann. Deshalb bedienen sich große und bekannte Internetanbieter, wie Google und Amazon, diesem nützlichen Prinzip.

Um REST Schnittstellen komfortabel und zeitsparend testen zu können, bietet sich das Programm Postman [Posa] hervorragend an. Die Request-Methode und die URL müssen angegeben werden. Der Request-Body ist nur für Requests zum Anlegen oder Bearbeiten von Daten notwendig. Wie das Ganze in Postman aussieht, lässt sich der Abbildung 2.1 entnehmen.

2 Theoretische Grundlagen

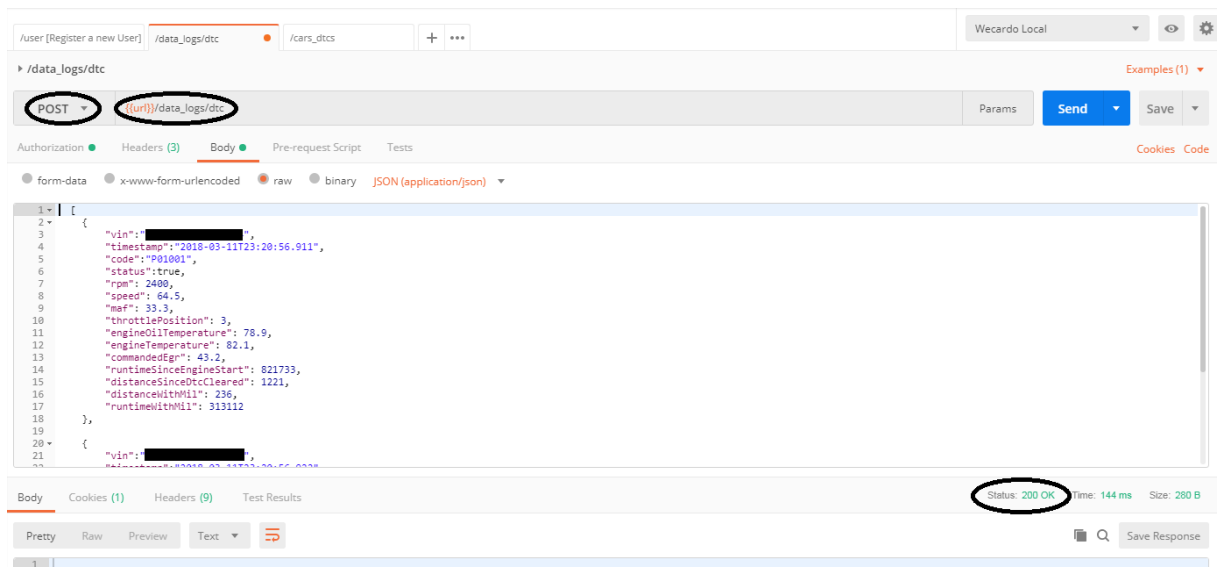


Abbildung 2.1: Das Programm Postman wird genutzt, um REST Schnittstellen zu testen

2.2 JSON

JavaScript Object Notation (JSON) ist ein leichtgewichtiges Format zum Austauschen von Daten, typischerweise zwischen Webserver und Client. Dieses Format wurde von der JavaScript Object Syntax abgewandelt, weswegen eine bidirektionale Umwandlung im JavaScript Code unkompliziert funktioniert. Doch auch in Programmiersprachen, wie zum Beispiel Java und Swift, gibt es standardmäßig eingebaute Funktionen, um JSON zu parsen. Die Syntax von JSON besteht aus folgenden Regeln:

- Arrays bestehen aus eckigen Klammern
- Objekte bestehen aus geschweiften Klammern
- Datenfelder von Objekten werden durch Key-Value Paaren dargestellt
- Datenfelder und Objekte werden durch Kommata getrennt

2.3 Dependency Injection

Dependency Injection ist ein Entwurfsmuster aus der objektorientierten Programmierung, welches dabei hilft, die Abhängigkeiten von Klassen zu organisieren [PP18, S. 27]. Die Abhängigkeiten werden an einem zentralen Ort einmalig instanziiert, identisch zum Singleton Pattern. Dabei werden die jeweiligen Klassen von ihren Abhängigkeiten entkoppelt und es wird verhindert, dass von ressourcen-intensiven Klassen mehrere Instanzen erzeugt wird. Des weiteren wird die Verwendung und Erstellung von Unit-Tests erleichtert.

Für die Umsetzung der Dependency Injection finden die drei Verfahren

- Constructor Injection
- Setter Injection
- Interface Injection

die meiste Verwendung.

Bei der Constructor Injection werden die Abhängigkeiten einer Klasse durch den Konstruktor übergeben und gesetzt [PP18, S. 119].

Durch jeweilige Methoden können die Abhängigkeiten bei der Setter Injection gesetzt werden [PP18, S. 120].

Die Interface Injection zeichnet sich dadurch aus, dass die abhängige Klasse eine Schnittstelle implementiert, durch die eine Methode vorgegeben wird, über die die Abhängigkeit zur Verfügung gestellt wird [PP18, S. 120].

2.4 Spring Framework

Das Spring Framework [Piv] ist ein Open-Source Framework für Java basierte Enterprise Projekte. Es wurde entworfen, um die Java EE Entwicklung zu vereinfachen und zu beschleunigen [Sch]. Spring implementiert Funktionalitäten wie

- Dependency Injection (siehe 2.3)
- Internationalisierung
- Datenbindung
- Validierung
- Typenkonvertierung

Insbesondere durch das Bereitstellen der Dependency Injection wird ein Einsatz von guten Softwarekonventionen gefördert, was für ein leicht wartbares und erweiterbares Projekt sorgt [Wol10, S. 20].

2.4.1 Spring Boot

Spring Boot ist eine Erweiterung zum Spring Framework mit dem Fokus auf Konventionen statt Konfiguration. Dadurch ist es deutlich schneller möglich, eine produktionsfähige Applikation zu erstellen, da bereits durch das Framework die meisten Entscheidungen getroffen wurden. Dennoch lassen sich jederzeit besagte Entscheidungen durch eine eigene Konfiguration überschreiben.

2.4.2 Spring Security

Spring Security erweitert das Spring Framework um einige Sicherheitsmechanismen, wie das Authentifizieren von Benutzern oder die Autorisierung in Form eines Rollenschemas mit verschiedenen Berechtigungen.

2.5 PostgreSQL

PostgreSQL, oft auch nur Postgres genannt, ist ein objektrelationales Datenbankmanagementsystem, welches als Open-Source Projekt angeboten wird. Es ist ein ehemaliges Projekt der „University of California, Berkeley“, das 1986 gestartet ist [Posb]. Als ein SQL-Interpreter im Jahre 1994 für Postgres geschrieben, woraufhin das gesamte Projekt als Open-Source unter dem Namen Postgres95 freigegeben wurde. Im Jahre 1996 wurde der aktuelle Name PostgreSQL gewählt, um die hinzugefügten SQL Fähigkeiten zu verdeutlichen [Posb].

2.5.1 Eigenschaften

Postgres ist [Atomicity, Consistency, Isolation and Durability \(ACID\)](#)-konform und fast vollständig mit dem SQL-Standard SQL:2011 konform, da mindestens 160 von 179 notwendige Funktionen erfüllt sind [Posc]. Es ist mit den Programmiersprachen C++, Delphi, Perl, Java, Lua, .NET, Node.js, Python, PHP, Lisp, Go, R, D und Erlang kompatibel [Bui]. Postgres unterstützt [Multi-Version Concurrency Control \(MVCC\)](#), ein Verfahren, um für eine gleichzeitige und Konsistenz wahrende Ausführung von konkurrierenden Zugriffen auf die Datenbank zu sorgen [Posc]. Die maximale Größe der Datenbanken wird entweder durch 32TB oder durch den tatsächlich verfügbaren Speicher begrenzt [Posc]. Postgres kann zudem durch selbst entworfene Funktionen, Operatoren und Datentypen erweitert werden und es unterstützt einige NoSQL Funktionen [Posc].

2.5.2 Vergleich zu MySQL

MySQL ist ebenfalls ein Open-Source Projekt. Es ist allerdings unter den relationalen Datenbankmanagementsystemen zu zählen. Genau wie PostgreSQL ist das oberste Element eine Tabelle. Die Funktionalitäten von MySQL und PostgreSQL sind beinahe identisch [Bui].

Zu den bekanntesten Unternehmen, die PostgreSQL als Datenbankserver verwenden, gehören unter anderen GitHub, US Navy, NASA, Tesla, YouTube, und Facebook verwenden MySQL als Datenbankserver [MyS]. Apple, Cisco, Skype, Uber, Groupon, Spotify, Netflix und Instagram verwenden PostgreSQL als Datenbankserver [Sta]. Auf beiden Seiten sind demnach größere und bekannte Unternehmen vertreten.

Da ein Vergleich der Performance stark abhängig von der jeweiligen Infrastruktur, des Einsatzzweckes und der Art der Abfragen ist, wird dies nicht weiter aufgeführt.

2.6 MVC Pattern

Das **Model View Controller (MVC)** Pattern ist ein Architekturmuster beziehungsweise Entwurfsmuster im Bereich der Software-Entwicklung. Hierbei wird das Programm in drei Schichten

- Model
- View
- Controller

aufgeteilt. Die Model Schicht enthält alle Klassen, die reale oder irreale Objekte modellieren, also jene, die nach der objektorientierten Programmierung entworfen wurden. Die View Schicht enthält die Oberfläche mit den Elementen zur Interaktion mit der Software. Da die Model Schicht und die View Schicht miteinander kommunizieren müssen und da die Eingaben des Benutzers verarbeitet werden müssen, ist die Controller Schicht gleichermaßen essenziell. Dieser Aufbau ist, wie beschrieben, in der Abbildung 2.2 ersichtlich.

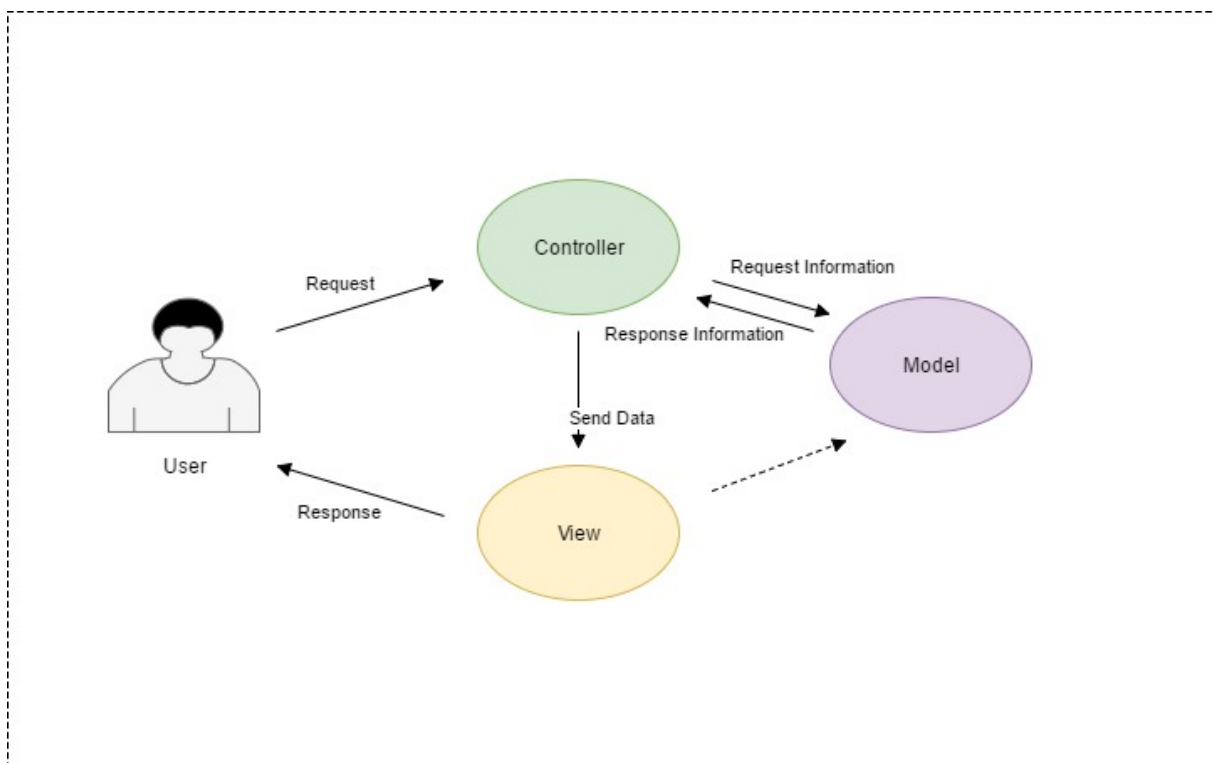


Abbildung 2.2: Der Aufbau des MVC Patterns [Cho]

Der Sinn dieses Architekturmusters ist eine lockere Kopplung der einzelnen Software-Module, um die Abhängigkeiten zu verringern und den Wartungsprozess, Erweiterungsprozess sowie Aktualisierungsprozess zu verbessern.

2.7 Swift

3 Entwurf des Systems

4 Entwicklung des Backends

5 Entwicklung der iOS App

6 Fazit

Literatur

- [Bui] An Bui. *PostgreSQL vs MySQL*. URL: <https://blog.panoply.io/postgresql-vs.-mysql> (besucht am 23.08.2018).
- [Cho] Kishan Choudhary. *Difference Between MVC And Web Forms*. URL: <https://www.c-sharpcorner.com/article/difference-between-mvc-and-web-forms/> (besucht am 20.08.2018).
- [Dat] Datical. *Liquibase*. URL: <https://www.liquibase.org/> (besucht am 03.09.2018).
- [MyS] MySQL. *MySQL Customers*. URL: <https://www.mysql.com/customers/> (besucht am 23.08.2018).
- [Piv] Pivotal. *Spring Framework*. URL: <https://spring.io/> (besucht am 27.08.2018).
- [Posa] Inc. Postdot Technologies. *Postman*. URL: <https://www.getpostman.com/> (besucht am 03.09.2018).
- [Posb] PostgreSQL. *A Brief History of PostgreSQL*. URL: <https://www.postgresql.org/docs/current/static/history.html> (besucht am 22.08.2018).
- [Posc] PostgreSQL. *About PostgreSQL*. URL: <https://www.postgresql.org/about/> (besucht am 22.08.2018).
- [PP18] Nilang Patel und Krupal Patel. *Java 9 Dependency Injection*. Packt Publishing Ltd, 2018. ISBN: 9781788296250.
- [Sch] Sebastian Schelter. *Das Spring Framework – eine Einführung*. URL: http://www.inf.fu-berlin.de/inst/ag-se/teaching/S-BSE/134_schelter_spring.pdf (besucht am 29.08.2018).
- [Sta] Stackshare. *Companies that use PostgreSQL*. URL: <https://stackshare.io/postgresql> (besucht am 23.08.2018).
- [Wol10] Eberhard Wolff. *Spring 3 - Framework für die Java-Entwicklung*. dpunkt.verlag, 2010. ISBN: 978-3-89864-572-0.