Dokumentation - Semesterarbeit

Karten Lern App

Studenten:	Alexander Studer
	Niklaus Hänggi
	Roman Joller
Klasse:	CAS OOP 22/14
Fachgebiet:	Objekt orientiertes Programmieren
Fachlehrer:	Mathias Bachmann
Datum:	Binningen, 20.01.2023

Management Summary

Ausgangslage und Problemstellung

Um zu Hause Sport zu treiben benötigt die Autorin diverse Apps (Musik, Timer und Pulstracker). Diese stehen zwar zur Verfügung, aber nur als eigenständige Apps. Es fehlt aus Sicht der Autorin eine All-in-One-Lösung. Dadurch muss während dem Training immer wieder zwischen den Apps hin- und hergewechselt werden, wodurch das Training unnötig unterbrochen wird.

Ziele der Arbeit

Um die oben dargelegte Problemstellung zu lösen, wurde im Rahmen dieser Diplomarbeit eine Web-Applikation mit nachfolgenden Zielen entwickelt:

- Im Frontend sollen Karten über ein vordefiniertes Formular angelegt und im Lernmodus werden können. Dabei kommt das TypeScript-basierte Frontend-Framework "Angular" zum Einsatz.
- Im Backend, welches auf dem Java-basierten Framework "Spring Boot" aufsetzt, dabei werden Entitäten wie Benutzer, Kategorien und Karten in die Datenbank gespeichert.
- Im Backend wird der Lernrhythmus berechnet und dem Benutzer entsprechend übers Frontend die zu lernenden Karten zur Verfügung gestellt.

Alle weiteren im Pflichtenheft definierten Ziele sind nicht Bestandteil der vorliegenden Arbeit.

Vorgehen und eingesetzte Methoden

In einem ersten Schritt wurden die Anforderungen an die Web-Applikation in einem Pflichtenheft festgelegt und ein Zeitplan mit definierten Meilensteilen aufgesetzt. Auf Basis der definierten Anforderungen wurde zu Beginn der Arbeit mithilfe der beiden Frameworks "Angular" und "Spring Boot" Lösungsvorschläge ausgearbeitet und anschliessend prototypenmässig umgesetzt. Das Frontend und das Backend in der IDE "IntelliJ" entwickelt.

Ergebnisse

Die definierten Level-1-Ziele konnten vollumfänglich umgesetzt werden. Es wurde eine Web-Applikation entwickelt, die die oben erwähnten Funktionen anbietet. Die Architektur basiert auf RESTful Web Services für den Datenaustausch. Das Backend wurde mithilfe des Frameworks "Spring Boot" entwickelt, das Frontend mit dem Framework "Angular".

Der Benutzer kann sich im Frontend anmelden und über ein vordefiniertes Formular Kategorien und Karten erstellen oder ändern.

Des Weiteren können Zu lernende Karten einer Kategorie im Lernmodus wiedergegeben werden.

Der Benutzer entscheidet, ob die Antwort korrekt war oder nicht, anschliessend wird diese Information über ein LearnDto zurück ans Backend gesendet, damit dort der Lernrhythmus aktualisiert werden kann.

Inhaltsverzeichnis

1. Einleitung

Auftraggeber

Bei der vorliegenden Semesterarbeit handelt es sich um ein Projekt des CAS Objekt orientiertes Programmieren der ZHAW.

Ausgangslage und Problemstellung

Eine altbekannte Lernmethode sind Karteikarten, welche eine Frage und auf der Rückseite die Antwort enthalten. Diese Lernmethode ist zwar sehr effektiv, allerdings muss Student diese Karten immer bei sich haben, wenn er lernen will und er muss auch Übersicht über den Lernrhythmus haben.

Dies werde in der heutigen Zeit in digitaler Form viel komfortabler.

Ziele der Arbeit

Um die oben dargelegte Problemstellung zu lösen, wurde im Rahmen dieser Diplomarbeit eine Web-Applikation mit nachfolgenden Zielen entwickelt:

- Im Frontend sollen Karten angelegt, geändert und im Lernmodus angezeigt werden können. Dabei kommt das TypeScript-basierte Frontend-Framework "Angular" zum Einsatz.
- Im Backend, welches auf dem Java-basierten Framework "Spring Boot" aufsetzt, sollen einerseits die Karten in einer Datenbank abgespeichert werden, andererseits soll im Backend Benutzer- und Kategorienverwaltung erfolgen

In einem ersten Schritt ist die Applikation darauf ausgerichtet, dass sie nur einem Benutzer zur Verfügung steht. Daher ist die Benutzerverwaltung noch nicht auf einen Mehrbenutzerbetrieb ausgerichtet. Ebenfalls nicht zwingender Bestandteil der vorliegenden Arbeit sind die im Pflichtenheft aufgeführten Level-2- und Level-3-Funktionen.

Vorgehen und eingesetzte Methoden

In einem ersten Schritt wurden die Anforderungen an die Web-Applikation in einem Pflichtenheft festgelegt und ein Zeitplan mit definierten Meilensteilen aufgesetzt. Auf Basis der definierten Anforderungen wurde zu Beginn der Arbeit mithilfe der beiden Frameworks "Angular" und "Spring Boot" Lösungsvorschläge ausgearbeitet und anschliessend prototypenmässig umgesetzt. Das Frontend und das Backend in der IDE "IntelliJ" entwickelt.

Aufbau der Arbeit

Nach einer Einleitung wird zuerst der Projektverlauf aufgezeigt. Im Folgenden ist die Arbeit in fünf Hauptkapitel unterteilt. Im ersten Kapitel werden die Use Cases dargestellt und erläutert. In den beiden nachfolgenden Kapiteln werden die Systemarchitektur und die Applikationsarchitektur detailliert beschrieben. Im vorletzten Kapitel folgt die Beschreibung der Test Cases. Das letzte Kapitel beinhaltet Anmerkungen zur Implementierung. Die Arbeit schliesst mit einem Fazit.

2. Projektverlauf

Nachfolgend werden die Meilensteine, unterteil in drei Phasen, mit dem jeweiligen Abgabedatum (SOLL / IST) aufgelistet. Zudem wird grob angegeben, wie viel Zeit für die Umsetzung der Meilensteine benötigt wurde.

Übersicht der Meilensteine

Vorbereitungsphase		
SOLL	**IST**	**Meilenstein**
02.12.2022	**02.12.2022**	Kick Off
16.12.2022	**16.12.2022**	Einarbeitung Spring Boot
16.12.2022	**9.12.2020**	Festlegung des Gui
Implementierung und Test		
SOLL	**IST**	**Meilenstein**
23.12.2022	**09.01.2023**	Datenbank funktionsfähig
30.12.2022	**09.01.2023**	Restschnittstelle definiert
24.01.2023	**18.12.2022**	Automatisiertes Docker-Deployment eingerichtet
30.12.2022	**17.01.2023**	Rest-Schnittstelle fertig
29.01.2023	**27.01.2023**	Backend fertig
29.01.2023	**31.01.2023**	Frontend fertig
28.01.2023	**23.12.2023**	Automatisiertes Github-Deployment fertig
01.10.2020	**01.10.2020**	Präsentation erstellen
Einführung		
SOLL	**IST**	**Meilenstein**
03.02.2023	03.02.2023	Abgabe Unterlagen (Code und Dokumentation)
03.02.2023	03.02.2023	Präsentation

Tabelle 1: Meilensteine

Zeitplan

Aufwand		**Aufgabe**
SOLL	**IST**	**Beschreibung**
10 h	10 h	Präsentation erstellen
40 h	40 h	Vorbereitung (Einarbeitung Spring Boot / Angular, API Dokumentation, Auswahl Datenbank, Auswahl Authentifizierung)
70 h	80 h	Programmierung Backend
10 h	15 h	Datenbank aufsetzen und anbinden
30 h	40 h	Frontend-Programmierung
20 h	25 h	Docker einrichten
15 h	15 h	Automatisiertes Deployment mit Github einrichten
30 h	60 h	Testing und Problembehebung
30 h	30 h	Dokumentation, Präsentation

Tabelle 2: Zeitplan

Geplant waren gemäss Pflichtenheft rund 180 Stunden für die Umsetzung der Level-1-Funktionen. Tatsächlich wurden für die Umsetzung rund 220 Stunden benötigt. Gründe hierfür sind:

- Für die Einarbeitung in die verschiedenen Themen (Spring Boot, Angular, Fitbit-API, Authentifizierung) wurden rund zehn Stunden mehr benötigt. Dies ist darauf zurückzuführen, dass die Autorin auch Level-2- und Level-3-Ziele (Fitbit-Integration, Authentifizierung) implementiert hat.
- Der erweiterte Funktionsumfang führte dazu, dass für die Implementierung und Dokumentation mehr Zeit investiert werden musste. Konkret wurden für die Erstellung der Views, das Aufsetzen und Anbinden der Datenbank sowie für die Programmierung der Eventhandler inkl. Datenspeicherung jeweils zehn Stunden mehr in Anspruch genommen.

Daraus abgeleitet mussten diverse Abgabetermine in der Meilensteinplanung verschoben werden.

3. Use Cases

Nachfolgend werden die Funktionalitäten aus Anwendersicht beschrieben:

Use Case UC01: Login

- Der Benutzer meldet sich mit Benutzername und Passwort an.

Use Case UC02: Benutzer anlegen

- Es kann ein neuer Benutzer angelegt werden.

Use Case UC03: Kategorie erstellen

- Der Benutzer kann eine Kategorie erstellen

Use Case UC04: Karten erstellen

- Der Benutzer kann Karten erstellen

Use Case UC05 Übersicht Anzeigen:

- Der Benutzer kann eine Übersicht mit seinen Kategorien und den zu lernenden Karten anzeigen

Use Case UC06: Lernen

- Der Benutzer kann die zu Lernenden Karten einer Kategorie im Lernmodus auf rufen und abarbeiten

Use Case UC07: Karten editiren

- Der Benutzer kann sich abmelden.

3.1. Use Case Diagramm

Abbildung 1: Use Case Diagramm

3.2. Use Case Details

Nachfolgend werden die definierten Use Cases im Detail beschrieben:

Use Case UC01: Login	
Beschreibung	Der Benutzer meldet sich mit Benutzername und Passwort an.
Akteure	Benutzer, Backend, Frontend
Auslöser	Der Benutzer möchte sich anmelden.
Eingaben	Benutzername und Passwort
Ablauf	1. Der Benutzer öffnet die Webseite und gelangt direkt zur Login-Seite. 2. Der Benutzer gibt den Benutzernamen und das Passwort ein. 3. Der Benutzer klickt auf den Button "Login".
Nachbedingungen	Der Benutzer ist angemeldet und hat Zugang zu seinen Lernkarten

Tabelle 3: Use Case UC01

Use Case UC02: Benutzer anlegen	
Beschreibung	Es kann ein neuer Benutzer angelegt werden.
Akteure	Benutzer, Frontend
Auslöser	Der Benutzer möchte sich registrieren
Eingaben	Benutzerdaten
Ablauf	1. Der Benutzer ruft die Registrierung-Seite auf 2. Der Benutzer gibt über ein Formular im Frontend Benutzername, Vorname, E-Mail-Adresse und Password ein. 3. Nach der Prüfung der Eingaben, kann der Ok-Button angewählt werden. 4. Drückt der Benutzer den Ok-Button, werden ans Backend gesendet und in der Datenbank abgespeichert.
Nachbedingungen	Alle Pflichtfelder sind korrekt ausgefüllt

Tabelle 4: Use Case UC02

Use Case UC03: Kategorie erstellen	
Beschreibung	Der Benutzer kann eine Kategorie erstellen
Akteure	Benutzer, Backend, Frontend
Auslöser	Benutzer möchte eine neue Kategorie anlegen.
Eingaben	Name der Kategorie
Ablauf	1. Der Benutzer gibt den Namen der Kategorie ein
Nachbedingungen	Die Kategorie ist in der Datenbank gespeichert und dem Benutzer zugeordnet. Nach folgend können Karten in der Kategorie angelegt werden.

Tabelle 5: Use Case UC03

Use Case UC04: Karten anlegen	
Beschreibung	Der Benutzer kann Karten erstellen
Akteure	Benutzer, Backend, Frontend
Auslöser	Der Benutzer möchte innerhalb einer Kategorie Karten anlegen
Eingaben	Frage und Antwort
Ablauf	1. Der angemeldete Benutzer wählt eine seiner Kategorien aus. 2. Er geht in den Bearbeitungsmodus der Kategorie 3. Wählt Karte erstellen an 4. Gibt Frage und Antwort ein 5. Bestätigt mit speichern
Nachbedingungen	Die Karte ist in der Datenbank gespeichert und der Kategorie zugeordnet.

Tabelle 6: Use Case UC04

Use Case UC05: Übersicht anzeigen	
Beschreibung	Der Benutzer kann eine Übersicht mit seinen Kategorien und den zu-lernenden Karten anzeigen
Akteure	Benutzer, Backend, Frontend
Auslöser	Übersichtseite wird nach dem Einloggen angezeigt
Eingaben	Benutzerlogin oder Homebutton
Ablauf	1. Einloggen oder Homebutton anklicken
Nachbedingungen	Eine Benutzerspezifische Übersicht wird aus dem Backend abgerufen und dem Benutzer im Frontend dargestellt

Tabelle 7: Use Case UC05

Use Case UC06: Lernen	
Beschreibung	Der Benutzer kann die zu Lernenden Karten einer Kategorie im Lernmodus aufrufen und abarbeiten
Akteure	Benutzer, Backend, Frontend
Auslöser	Der Benutzer mit dem Lernen starten
Eingaben	Klick auf eine Kategorie
Ablauf	1. Der Benutzer wählt eine Kategorie aus in welcher lernen will 2. Die Frage der 1. Karte (Reihenfolge wird vom Backend berechnet) wird dem Benutzer angezeigt. 3. Wenn der Benutzer die Antwort sehen möchte, kann er auf den Button «show answer» klicken und ihm wird die Antwort angezeigt 4. Der Benutzer kann mit den Buttons entscheiden ob die Antwort korrekt oder falsch war.
Nachbedingungen	Die Information wird ans Backend gesendet, damit die Karte neu eingeordnet werden kann und dem Benutzer wird die nächste Frage angezeigt

Tabelle 8: Use Case UC06

Use Case UC07: Abmelden	
Beschreibung	Der Benutzer meldet sich ab
Akteure	Benutzer, Backend, Frontend
Auslöser	Der Benutzer klickt auf den «Logout» Button
Eingaben	Klick auf den «Logout» Button
Ablauf	5. Der Benutzer wählt eine Kategorie aus in welcher lernen will 6. Die Frage der 1. Karte (Reihenfolge wird vom Backend berechnet) wird dem Benutzer angezeigt. 7. Wenn der Benutzer die Antwort sehen möchte, kann er auf den Button «show answer» klicken und ihm wird die Antwort angezeigt 8. Der Benutzer kann mit den Buttons entscheiden ob die Antwort korrekt oder falsch war.
Nachbedingungen	Die Information wird ans Backend gesendet, damit die Karte neu eingeordnet werden kann und dem Benutzer wird die nächste Frage angezeigt

Tabelle 9: Use Case UC06

Use Case UC07: Logout	
Beschreibung	Der Benutzer kann sich abmelden.
Akteure	Benutzer, Frontend
Auslöser	Der Benutzer möchte sich abmelden.
Eingaben	Keine Eingaben nötig
Ablauf	1. Der angemeldete Benutzer klickt auf die Menüoption "Logout".
Nachbedingungen	Der Benutzer ist abgemeldet und hat keinen Zugriff auf seine Daten.

Tabelle 10: Use Case UC07

Use Case UC08: Fitbit-Daten speichern	
Beschreibung	Der Benutzer kann Fitbit-Daten beziehen und in der Datenbank speichern.
Akteure	Benutzer, Backend, Frontend, Fitbit-API
Auslöser	Der Benutzer möchte Fitbit-Daten beziehen.
Eingaben	Fitbit-Login, vorerst keine weiteren Eingaben (siehe Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** "**Fehler! Verweisquelle konnte nicht gefunden werden.**")
Ablauf	1. Der angemeldete Benutzer wählt die Menüoption "Fitbit" aus und kann sich dort über den Button "Login" bei Fitbit anmelden. 2. Nach erfolgtem Login kann der Benutzer über den Button "Get Fitbit-Data" Pulsdaten einer bestimmten Zeitspanne von der Fitbit-API beziehen.
Nachbedingungen	Die Fitbit-Daten sind in der Datenbank gespeichert und werden angezeigt.

Tabelle 11: Use Case UC08

Use Case UC09: Fitbit-Daten anzeigen	
Beschreibung	Der Benutzer kann sich die im Backend gespeicherten Fitbit-Daten anzeigen lassen.
Akteure	Benutzer, Backend, Frontend
Auslöser	Der Benutzer möchte Fitbit-Daten anzeigen lassen.
Eingaben	Keine Eingaben nötig
Ablauf	1. Der angemeldete Benutzer wählt die Menüoption "Fitbit" aus und kann dort über den Button "Show" Fitbit-Daten anzeigen lassen.
Nachbedingungen	Die Details des ausgewählten Fitbit-Datensatzes werden tabellarisch und grafisch angezeigt.

Tabelle 12: Use Case UC09

4. Systemarchitektur

Die Architektur basiert auf RESTful Web Services für den Datenaustausch, auf dem Framework "Spring Boot" für das Backend und dem Framework "Angular" für das Fronend. Die Architektur ist in Abbildung 2 vereinfacht dargestellt.

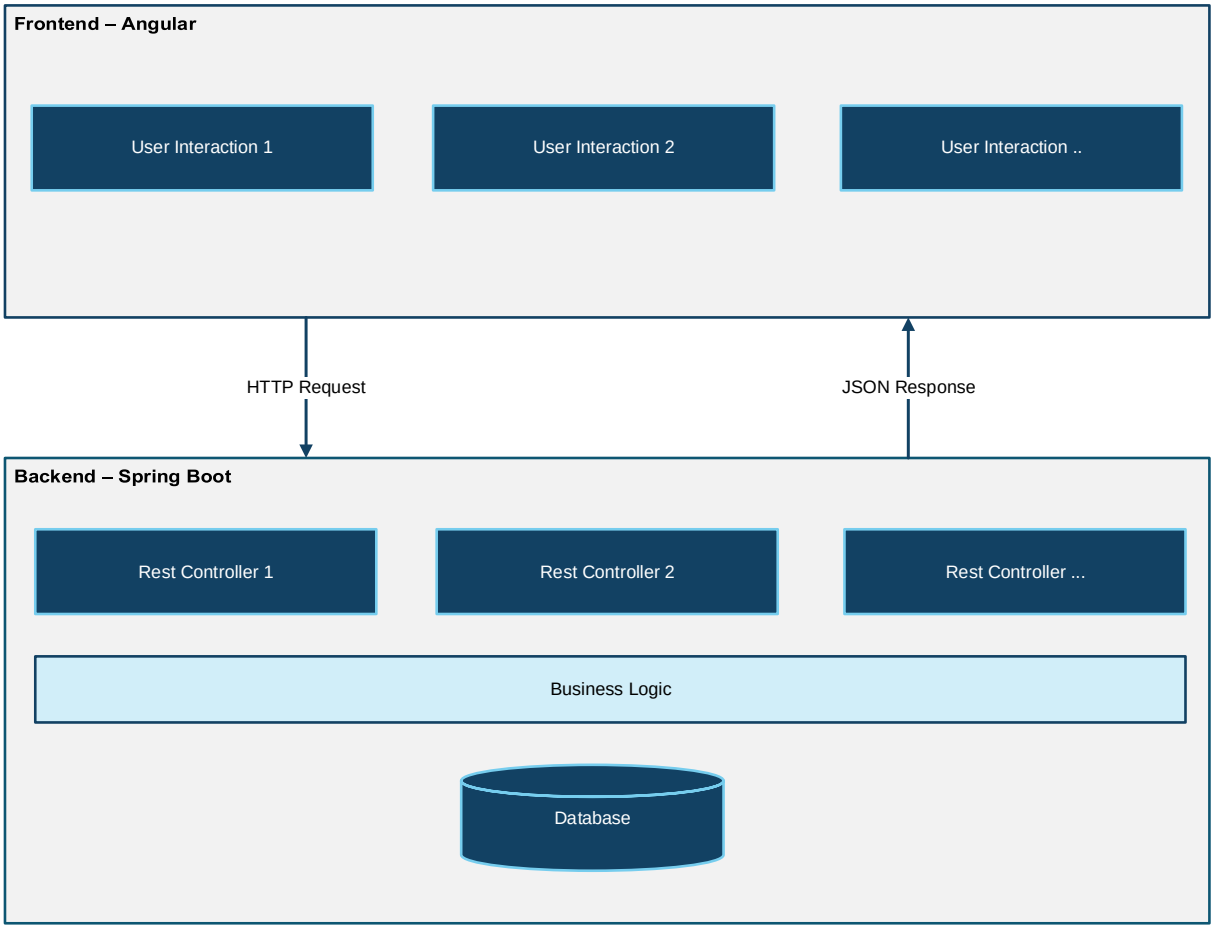

Abbildung 2: Systemarchitektur (IT-P, o.J.)

RESTful Web Service

Bei RESTful Web Service handelt es sich um leichte, wartbare und skalierbare Services, die auf der REST-Architektur basieren. REST (Representational State Transfer) ist eine Möglichkeit, auf Ressourcen zuzugreifen, die sich in einer bestimmten Umgebung befinden. Jede Ressource ist über einen eindeutigen Unique Resource Identifier (URI) identifizierbar. Dabei wird stets über einen einheitlichen Satz von Standardmethoden auf eine Ressource zugegriffen, wie z. B. die Standard-HTTP-Methoden GET, POST, PUT usw.. Ein Client kann somit etwa eine Ressource explizit beispielsweise im XML- oder JSON-Format anfordern. Ressourcen werden von ihrer Darstellung entkoppelt, sodass auf ihre Inhalte in verschiedenen Formaten wie HTML, XML, Klartext, PDF, JPEG, JSON und anderen zugegriffen werden kann (Guru99, o.J.; Oracle, 2013).

Spring Boot

Spring Boot ist eine Open Source "Konvention vor Konfiguration"-Lösung für das Java-Framework Spring, die die Komplexität der Konfiguration neuer Spring-Projekte reduziert. Zu diesem Zweck legt Spring Boot eine Grundkonfiguration inklusive Richtlinien für die Nutzung des Frameworks sowie aller relevanten Drittanbieter-Bibliotheken fest und gibt damit den Weg vor, um den Einstieg in neue Projekte so mühelos wie möglich zu gestalten (IONOS, 2020).

Angular

Angular von Google ist eine Open Source Plattform und ein Framework zum Erstellen von Single-Page-Client-Anwendungen mit HTML und TypeScript. Es implementiert Kernfunktionen und optionale Funktionen als eine Reihe von TypeScript-Bibliotheken, die importiert werden können (Docker, 2022).

Docker

er vereinfacht die Bereitstellung von Anwendungen, weil sich Container, die alle nötigen Pakete enthalten, leicht als Dateien transportieren und installieren lassen. Container gewährleisten die Trennung und Verwaltung der auf einem Rechner genutzten Ressourcen. Das umfasst laut Aussage der Entwickler: Code, Laufzeitmodul, Systemwerkzeuge, Systembibliotheken – alles was auf einem Rechner installiert werden kann

(Docker, 2022).

Weitere Frameworks, Tools und Bibliotheken

Konstruktoren, Setter und Getter mit dem Lombock Framework erstellt.

Für die Serialisierung und De- Serialisierung der JSON-Dateien wurde Jackson verwendet.

Für das Deployment durch Maven

Release Plugin über Maven.. H2 als Dev DB und MySql als Prod DB

4.1. Applikationsdiagramm

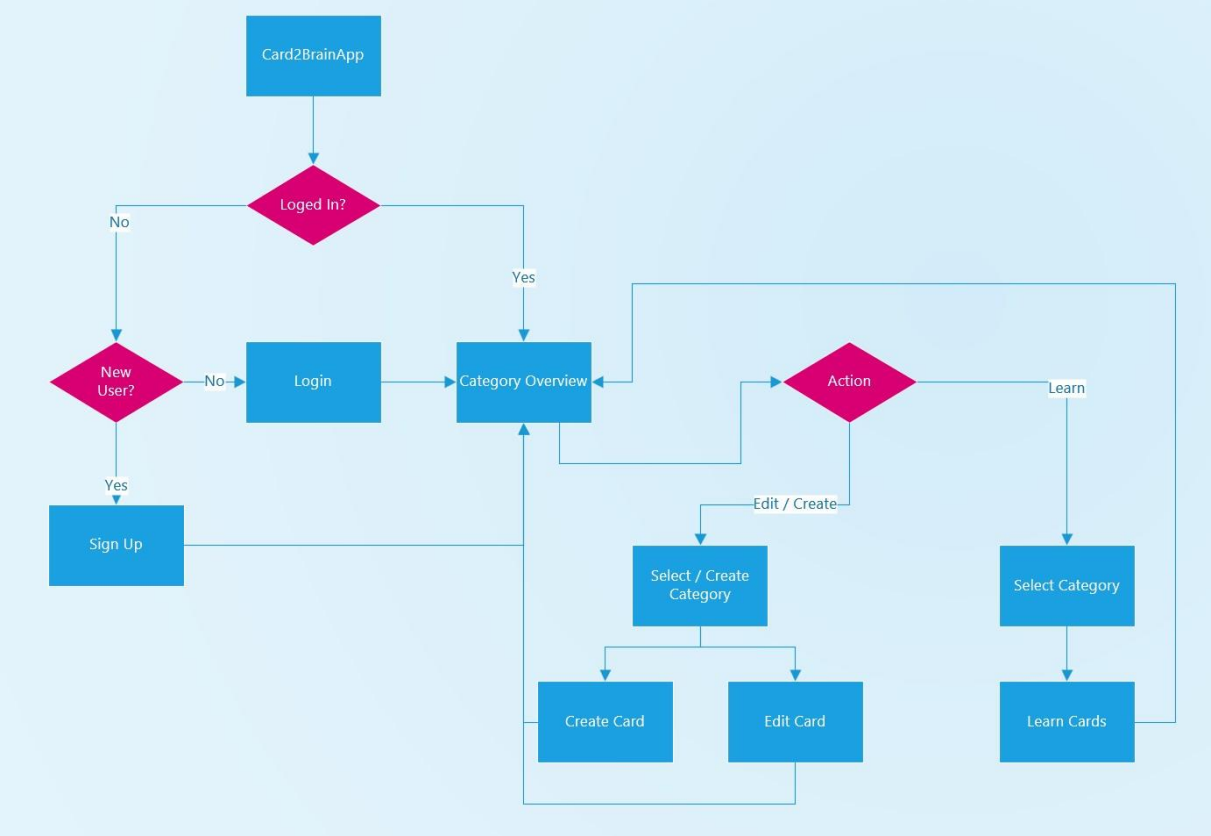

Abbildung 3 Applikation

4.2. Entitäten Diagramm

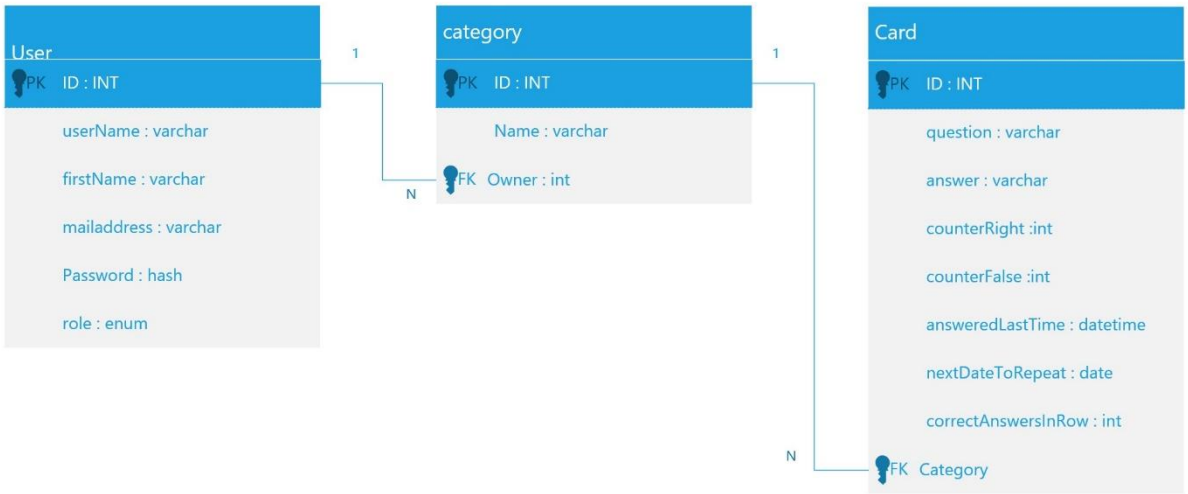

Abbildung 4 Entitäten Diagramm

4.3. Klassen Diagramm

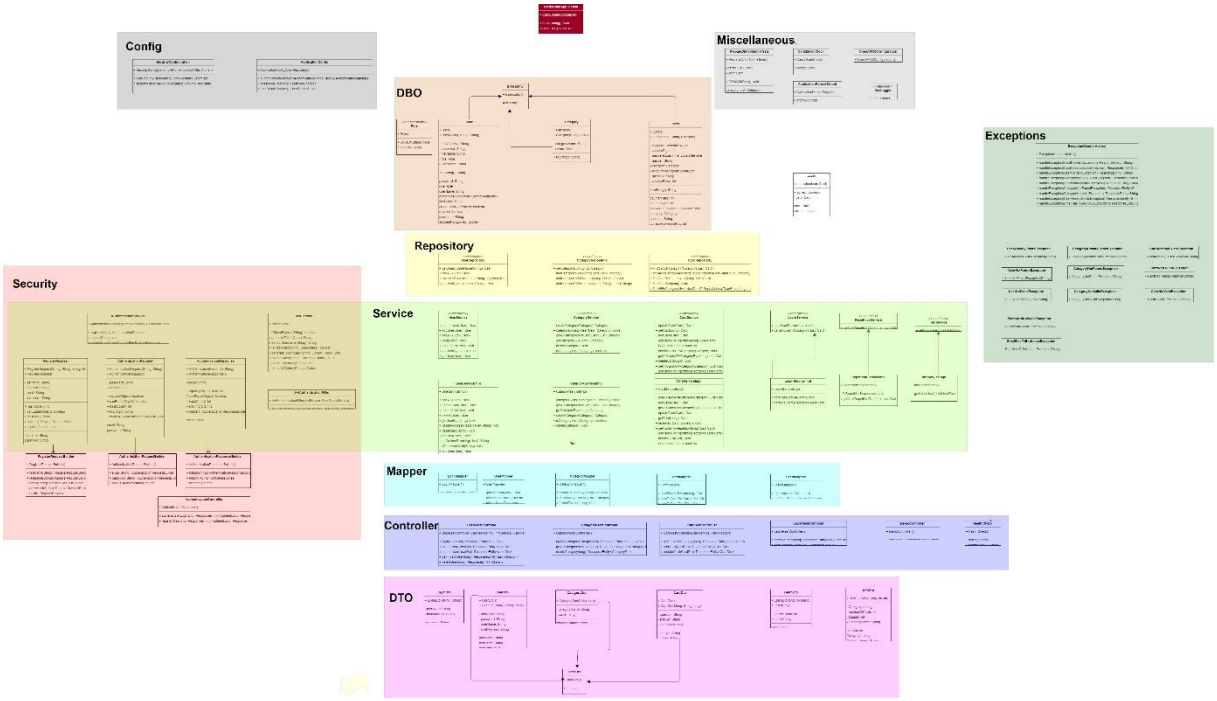

Abbildung 5 Klassen Diagramm

Literaturverzeichnis

Docker. (2022). *Package Software into Standardized Units for Development*. Abgerufen am 12. 09 2020 von https://www.docker.com/resources/what-container/

Fitbit. (o.J.). *Inside our APIs*. Abgerufen am 20. 09 2020 von https://dev.fitbit.com/build/reference/

Fitbit. (o.J.). *Web API Reference*. Abgerufen am 19. 09 2020 von https://dev.fitbit.com/build/reference/web-api/

Guru99. (o.J.). *RESTful Web Services Tutorial with Example*. Abgerufen am 12. 09 2020 von https://www.guru99.com/restful-web-services.html

In28Minutes. (05. 07 2020). *Spring Boot and Embedded Servers - Tomcat, Jetty and Undertow*. Abgerufen am 22. 09 2020 von https://www.springboottutorial.com/spring-boot-with-embedded-servers-tomcat-jetty

IONOS. (24. 07 2020). *JSON Web Token (JWT) vorgestellt*. Abgerufen am 22. 09 2020 von https://www.ionos.de/digitalguide/websites/web-entwicklung/json-web-token-jwt-vorgestellt/

IONOS. (16. 03 2020). *Spring-Boot-Tutorial: Java-Apps mit Spring Boot erstellen*. Abgerufen am 12. 09 2020 von https://www.ionos.de/digitalguide/websites/web-entwicklung/spring-boot-tutorial/

IT-P. (o.J.). *Warum eigentlich ein Spring-Boot Backend?* Abgerufen am 23. 08 2020 von https://www.it-p.de/newsblog/177-angular-5-meets-spring-boot-java

Oracle. (2013). *What Are RESTful Web Services?* Abgerufen am 12. 09 2020 von https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

RIP Tutorial. (o.J.). *Getting started with jpa*. Abgerufen am 14. 09 2020 von https://riptutorial.com/jpa

tutorialspoint. (o.J.). *H2 Database Tutorial*. Abgerufen am 12. 09 2020 von https://www.tutorialspoint.com/h2_database/index.htm

webpack. (o.J.). *Development*. Abgerufen am 22. 09 2020 von https://webpack.js.org/guides/development/#using-webpack-dev-server

Wikipedia. (13. 04 2020). *CRUD*. Abgerufen am 16. 09 2020 von https://de.wikipedia.org/wiki/CRUD

ZetCode. (06. 07 2020). *Spring Boot JpaRepository tutorial*. Abgerufen am 16. 09 2020 von http://zetcode.com/springboot/jparepository/

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CRUD	Create, Read, Update und Delete
DB	Datenbank
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JWT	JSON Web Token
REST	REpresentational State Transfer
TCP/IP	Transmission Control Protocol/Internet Protocol
UC	Use Case
URI	Unique Resource Identifier

Anhang

Installation Guide

Nachfolgend wird kurz auf die Systemvoraussetzungen eingegangen. Unter den angegebenen Links sind detaillierte Installationsanleitungen verfügbar.

Backend

Spring Boot kann mit "klassischen" Java-Entwicklungstools verwendet oder als Befehlszeilentool installiert werden. Unabhängig davon wird Java SDK v1.6 oder höher benötigt. Weitere Informationen sind unter https://docs.spring.io/spring-boot/docs/1.0.0.RC5/reference/html/getting-started-installing-spring-boot.html erhältlich.

Sobald die Applikation in einer IDE gestartet wurde, ist der embedded Tomcat-Server auf http://localhost:8080 erreichbar, die integrierte H2-Datenbankkonsole auf http://localhost:8080/h2-console.

Frontend

Um Angular auf dem lokalen System zu installieren, wird Folgendes benötigt:

- Node.js
- npm package manager
- Angular CLI

Weitere Informationen zur Angular-Installation sind unter https://angular.io/guide/setup-local erhältlich.

Über den Angular-Befehl "ng serve" im Workspace-Ordner ("Diplomarbeit\frontend\fitnessapp") kann die Angular-Applikation auf http://localhost:4200 gestartet werden.