Modul 151

Thema: LB2

Dokumentinformationen

Dateiname: Modul 151.docx

Speicherdatum: 01.11.2022

Autoreninformationen

Autor: Anojan Pirabakaran | Shawn Lacarta

Inhaltsverzeichnis

[1 Projektbeschreibung 3](#_Toc118195047)

[2 Architektur 3](#_Toc118195048)

[3 Frameworks 3](#_Toc118195049)

[3.1 Presentation Tier 3](#_Toc118195050)

[3.2 Sevice Tier 4](#_Toc118195051)

[3.3 Data Tier 4](#_Toc118195052)

[4 ERM 4](#_Toc118195053)

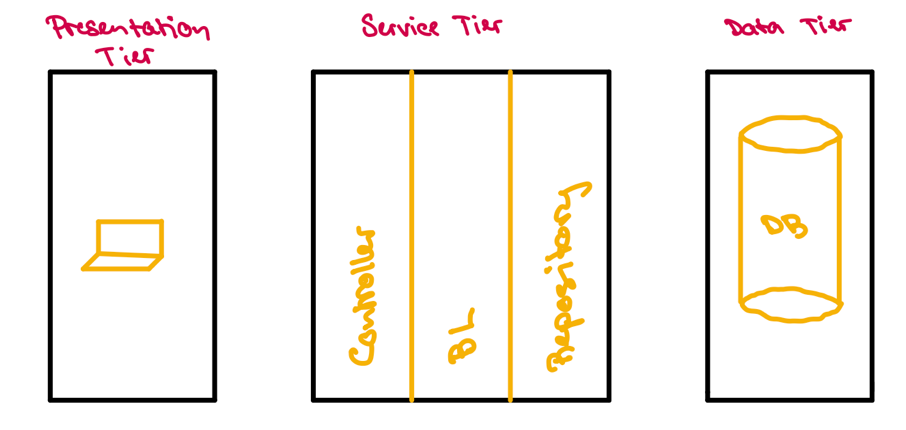
[5 Fazit Projekt 5](#_Toc118195054)

[6 Reflexion Gruppenarbeit 5](#_Toc118195055)

# Projektbeschreibung

In unserem Projekt geht es darum, ein Webshop zu simulieren. Verschiedene User können sich einloggen und Daten manipulieren, wenn die entsprechenden Rechte vorhanden sind. Zum einen kann man als User seine eigenen Daten abändern. Anderseits kann man als Administrator neue Produkte in den Shop hinzufügen.

# Architektur

Für die Architektur des Projektes haben wir uns für eine hybride Form zwischen dem MVC und N-Tier entschieden.

Wir haben den Presentation Tier für die Darstellung der Komponenten.

Den Service Tier haben wir in drei Layers unterteilt für eine klare Struktur.

Zuletzt haben wir den Data Tier, worin wir unsere Daten abspeichern.

# Frameworks

Wir arbeiten mit Frameworks, damit wir mit Funktionen zur Verfügung bekommen und nicht alles selbst programmieren können. Heutzutage arbeiten fast alle Firmen mit Frameworks und wir haben uns für diese entschieden:

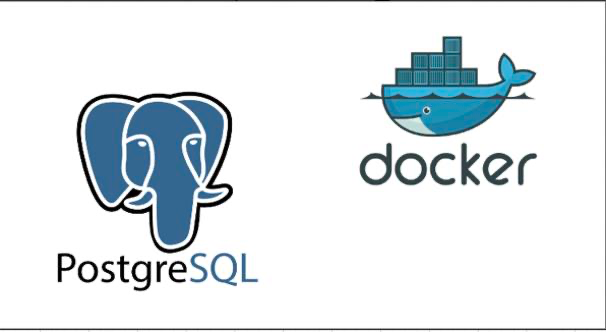
## Presentation Tier

Hier verwenden wir Angular als Framework. Angular bietet viele Libraries an, welches wir verwenden können, um effizient zu arbeiten. Die Programmiersprache ist TypeScript. Es ist vorteilhaft, da wir unsere Eingabefelder Type-Sicher machen können. Die Verbindung zum Backend machen wir mit dem http-Client, was von Angular zur Verfügung gestellt wird.

## page6image10194592Sevice Tier

Im Backend verwenden Spring Boot mit Java. Hier haben wir mit dem REST-Prinzip gearbeitet. Das Ganze läuft über Gradle. Als ORM haben wir Hibernate.

## Data Tier

Für die Datenbank verwenden wir eine PostgreSQL Datenbank, welches wir in einem Docker-Container haben.

# ERM

Ein Bild, das Text, drinnen, Screenshot enthält.

Automatisch generierte Beschreibung

Das ERM beinhaltet alle Tabellen, die in der Datenbank vorhanden sind. Das ERM zeigt auch alle Kardinalitäten zwischen den Tabellen, damit es für eine nicht-technische Person verständlich ist.

# Testkonzept

Um das Spring Boot Projekt zu testen, hat man mehrere Möglichkeiten, aber die einfachste Methode ist es JUnit-Tests zu schreiben. Um alle Controller-Klassen zu testen, kann man mit Hilfe von JUnit und MockMvc einen Request auf die URL senden und mit einem JSONAssert einen ganzen JSON-String mit dem expected String vergleichen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Um die Service-Klasse auszutesten, kann man auch mit JUnit Tests schreiben. Man erstellt Mockdaten und mockt den Repository. Mit der gemockten Repository, ruft man die Methode, die getestet werden soll. Mit der verfiy()-Methode kann man berechnen, wie viel Mal die Methode aufgerufen wurde.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

# Fazit Projekt

Das Projekt ist gut relativ gegangen. Wir haben die Zeit ein bisschen unterschätzt und hatten gegen das Ende etwas stress. Jedoch ist es trotz allen recht gut gegangen. Wir beide fanden, dass das Projekt recht spass gemacht hat. Da wir beide mit den gleichen Frameworks arbeiten, konnten wir viele neue Sachen, welches wir gelernt haben, anwenden.

Was uns am meisten gefallen hat war, dass wir während dem Projekt an sehr viele kleine Fehler angestossen sind und sie fixen konnten. So haben wir unser Wissen erweitert. Was uns nicht so gefallen hat war, dass wir mit dem Login im Frontend recht Mühe hatten, da wir mit CORS Errors uns beschäftigen mussten. Jedoch haben wir das Beste draus gemacht.

# Reflexion Gruppenarbeit

Die Gruppenarbeit war sehr gut. Wir hatten schon an vorigen Modulen zusammengearbeitet und haben die Chemie für das Zusammenarbeiten. Wir haben uns auch so aufgeteilt, dass Anojan eher Backendtätig war und Shawn eher Frontend. Klar haben wir uns immer gegenseitig unterstützt, falls wir an Probleme geraten sind. Die Kommunikation war bei uns Top, da wir jederzeit erreichbar waren und auch in der Freizeit regelmässig Calls hatten.