

2021-02-22

# Dokumentation zum CocktailCalculator

# Einführung und Ziele

## Aufgabenstellung

Entwicklung einer Webanwendung, die bei Angabe eines Cocktails den Preis für die Zutaten pro Cocktail berechnet.

Hintergrund ist der Wunsch von Studenten, die wissen wollen, wie viel sie ihr Lieblingscocktails kosten wird. Diese können die entwickelte Webanwendung dann nutzen, um schnell die Kosten für ihren gewünschten Cocktail einsehen zu können.

Der User kann per Browsereingabe seinen gewünschten Cocktail angeben. Daraufhin wird ihm als Ergebnis der Programmberechnung der Preis der Zutaten des angegebenen Cocktails zurückgegeben.

Aufgrund von Komplexitätsgründen kann derzeitig nur nach Margherita gesucht werden, woraufhin Tommys Margherita ausgegeben wird.

## Stakeholder

|  |  |  |
| --- | --- | --- |
| **Rolle** | **Name(n)** | **Erwartungshaltung** |
| Auftraggeber | Ralf Klemmer | Entwicklung eines lauffähigen Programms inklusive Projektdokumentation und Veröffentlichung des Projekts auf GitHub |
| Entwicklerteam | Dennis, Jan, Nikolay, Tanja und Jessica | Erwartungshaltung des Auftraggebers fristgerecht erfüllen |
| User | Vor allem Studenten | Nutzbarkeit der lauffähigen Webanwendung, die den Use-Case korrekt ausführt |

# Randbedingungen

* Einbindung einer REST-API an die eigene Anwendung
* Hochladen des Projekts auf GitHub
* (optional) automatisierte Testfälle
* Verwendung von Domain Driven Development

# Kontextabgrenzung

Zum Kontext des Systems gehört der von uns geschriebene Anwendungscode, sowie die angebundene API („TheCocktailDB.com“). Außerhalb des Kontextes liegen somit alle anderen vorgeschlagenen APIs. Außerdem liegt ein mögliches Frontend außerhalb des Kontextes. Dieses wurde jedoch nicht implementiert.

# Lösungsstrategie

* Nutzung des Spring-Frameworks → Erleichtert die Entwicklung einer Webanwendung und war den Entwicklern bereits bekannt
* Implementierung in Java → Meiste Erfahrung bei den Entwicklern vorhanden
* Ausschließliche Entwicklung eines Back-Ends, das Aus- und Eingaben über den Browser abwickelt (spätere Entwicklung eines Front-Ends wäre möglich) → Zeitgründe
* Architekturmuster: Monolith → Vereinfachung der Softwarearchitektur

# Bausteinsicht

Verwendete Bausteine:

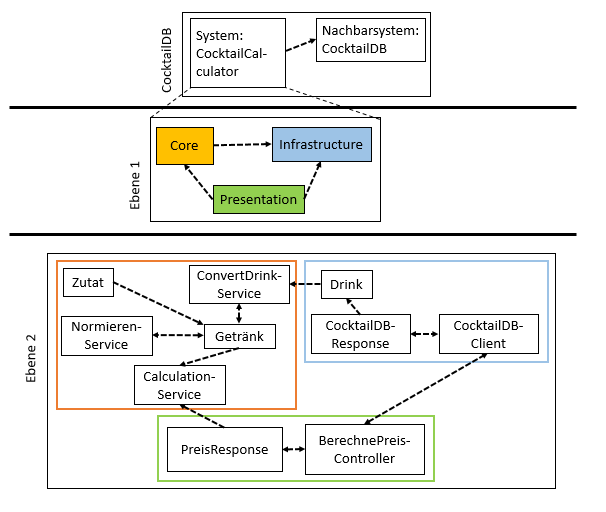
* Framework: Spring Boot
* Datenstrukturen:
  + public class Zutat {

private String name;

private String menge;

private double mengeInMl;

}



# Laufzeitsicht

1. User stellt Anfrage per Browsereingabe
2. Anfrage an die API
3. API sendet angefragte Daten zurück
4. Empfangene Daten werden als Getränk mit der Liste Zutaten gespeichert
5. Preiskalkulation
6. Ausgabe des berechneten Preises im Browser

# Risiken und technische Schulden

* Falls die API einmal nicht mehr öffentlich zugänglich ist, funktioniert die Anwendung nicht mehr
* Software läuft aktuell nur auf einem privaten Rechner. Wenn dieser durch Störungen ausfällt, ist die Software nicht mehr nutzbar oder könnte bei fehlender Sicherung auf GitHub nicht mehr vorhanden sein
* Durch monolithische Struktur könnte die Software bei vielen Usern an ihre Grenzen stoßen, sodass die Umstellung auf Micro Services notwendig wäre