

# Projektarbeit

## Coctailmaschiene

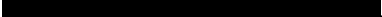
Johanna Reidt, Freya Dorn, Amir Rabieyan Nejad und Phillip Wieber  
5. März 2024

### **Betreuer**

Prof. Dr. Thorsten Thormählen

Philipps-Universität Marburg  
Fachbereich Mathematik und Informatik  
Hans-Meerwein-Straße  
35032 Marburg

---





---

# Inhaltsverzeichnis

<b>Abstract</b>	<b>I</b>
<b>Inhaltsverzeichnis</b>	<b>I</b>
<b>1 Planung</b>	<b>1</b>
1.1 Kurzbeschreibung . . . . .	1
1.1.1 Cocktailmaschine . . . . .	1
1.1.2 Funktion der App . . . . .	1
1.2 Entwicklungsprozess . . . . .	2
1.3 Team . . . . .	2
1.4 Risikomanagement . . . . .	2
1.5 Zeitplan . . . . .	2
<b>2 Anforderungsanalyse</b>	<b>3</b>
2.1 Definition des Zielsystems . . . . .	3
2.2 Funktionale Anforderungen . . . . .	3
2.3 Nicht-funktionale Anforderungen . . . . .	3
<b>3 Entwurf</b>	<b>4</b>
3.1 Grober Entwurf (Architektur) . . . . .	4
3.2 Technologien . . . . .	4
3.3 Detaillierter Entwurf . . . . .	4
3.3.1 ESP: Befehle . . . . .	4

3.3.2	ESP: Fehler . . . . .	10
3.3.3	Fehler: Kalibrierung . . . . .	10
3.3.4	ESP: Services . . . . .	12
3.3.5	ESP: Zustände . . . . .	14
3.3.6	GUI: Öffnung der App . . . . .	15
3.3.7	GUI: Keine Kalibrierung . . . . .	16
3.3.8	GUI: Dialog: Anmelden . . . . .	16
3.3.9	GUI: Dialog: Automatische Kalibrierung . . . . .	18
3.3.10	GUI: Darstellung Liste . . . . .	19
3.3.11	GUI: Anzeigen einzelner Elemente . . . . .	20
3.3.12	GUI: Ändern oder Hinzufügen . . . . .	21
3.3.13	GUI: Menü . . . . .	22
3.3.14	GUI: Einstellung . . . . .	23
3.3.15	Datenbank . . . . .	23
3.3.16	DB: Zustände . . . . .	26
3.3.17	DB: Administrator . . . . .	26
3.3.18	Schnittstellen . . . . .	26
<b>4</b>	<b>Qualitätssicherung</b>	<b>31</b>
4.1	Testplan . . . . .	31
4.2	Testprotokoll . . . . .	31
<b>5</b>	<b>Abschlussbericht</b>	<b>32</b>
5.1	Zusammenfassung . . . . .	32
5.2	Beispielanwendungen . . . . .	32
5.3	Benutzerdokumentation . . . . .	32
5.4	Entwicklerdokumentation . . . . .	32
5.5	Erfahrungsbericht . . . . .	32
	<b>Literaturverzeichnis</b>	<b>IV</b>
	<b>Tabellenverzeichnis</b>	<b>VI</b>
	<b>Abbildungsverzeichnis</b>	<b>VII</b>
	<b>Abkürzungsverzeichnis</b>	<b>VIII</b>

<b>Listings</b>	<b>IX</b>
<b>Appendix</b>	<b>X</b>
<b>Eidesstattliche Erklärung</b>	<b>XI</b>



---

## 1.1 Kurzbeschreibung

Die Funktion der App ist die Steuerung einer Cocktailmaschine.

### 1.1.1 Cocktailmaschine

Die Cocktailmaschine ist ein Gerät, welches über  $k$  Pumpen verfügt. Diese Pumpen werden von einem Micro-Chip gesteuert. Jede Pumpe führt von einem Flüssigkeitsbehälter zu einer kaffeemaschinenähnlichen Ausgabe, unter der man ein Glas stellen kann. Der Micro-Chip ist mit Bluetooth steuerbar.

### 1.1.2 Funktion der App

Die Funktion der App ist die Steuerung einer Cocktailmaschine. Dabei sollen angezeigt werden:

- die Flüssigkeiten, die an der Cocktailmaschine angeschlossen sind (Wodka, Rum, Cola, etc. ),
- die damit erstellbaren Cocktails (Mochito, Long Island Ice Tea, etc.),
- Angaben zu zusätzlichen Zutaten, die nicht von Cocktailmaschine steuerbar sind (Eis-Würfel, Zitronenscheiben, etc.)

Die Funktionen der App gehen über die reine Darstellung hinaus:

- die Neuerstellung von Cocktails mit den verfügbaren Zutaten
- einen Auftrag aufgeben den ausgesuchten Cocktail zu mixen (Zitronenscheiben, etc.)

Ein Administrator sollte zudem:

- die Flüssigkeiten einer Pumpe ändern können (Nachfüllen, andere Zutat)
- Zugriff auf alle auch nicht verfügbaren Zutaten und Rezepte haben
- das Hinzufügen neuer Zutaten Zitronenscheiben, etc.)

---

## **1.2 Entwicklungsprozess**

---

## **1.3 Team**

---

## **1.4 Risikomanagement**

---

## **1.5 Zeitplan**



---

# 2

# Anforderungsanalyse

---

## 2.1 Definition des Zielsystems

Das Ziel ist eine App als Fernbedienung einer Cocktailmaschine zu entwickeln. Dabei sollen Rezepte erstellt und gemixt werden.

---

## 2.2 Funktionale Anforderungen

Ein grober Überblick über die funktionalen Anforderungen:

- Anzeige der Rezepte, Zutaten, Serviervorschläge und Pumpen im Einzelnen und gelistet
- Erstellen von Rezepten, Zutaten, Serviervorschläge und Pumpen
- Mixen von Cocktails
- Cocktailmaschine einrichten und Zustand abfragen

---

## 2.3 Nicht-funktionale Anforderungen

---

## 3.1 Grober Entwurf (Architektur)

---

## 3.2 Technologien

- Bluetooth
- SQL Datenbank
- JAVA (Android)

---

## 3.3 Detaillierter Entwurf

### 3.3.1 ESP: Befehle

Nachrichten werden als JSON-Maps (in UTF8) kodiert und folgen alle dem selben Format.

Für unverschlüsselte Nachrichten ist das Format:

```
*{"cmd: "der Befehl", üser: "die User-ID", ärgument: "weitere Werte", ...} *
```

Das "cmdFeld ist verpflichtend und enthält den Namen des Befehls. Die meisten Nachrichten benötigen auch ein üserFeld mit der User-ID. Eventuelle weitere Argumente werden als weitere Felder in der Map gesendet.

Für verschlüsselte Nachrichten wird die Nachricht in einen Wrapper gepackt:

```
* {"msg: "verschlüsselte JSON-Nachricht", üser: "die User-ID"} *
```

Der Key zum Entschlüsseln ist mit der User-ID verbunden und wird benutzt, um die Nachricht in "msgbu ver-/entschlüsseln. Die entschlüsselte Nachricht ist ein normaler Befehl der Form "cmd: ...".

Momentan unterstützt der ESP noch keine verschlüsselten Nachricht.

### **3.3.1.1 Allgemeine Befehle**

#### **3.3.1.1.1 test (USER): Dummy-Befehl, der nichts macht** JSON-Beispiel:

```
* {"cmd: "test"} *
```

#### **3.3.1.1.2 init\_user (USER): als neuer Benutzer registrieren und eine User-ID erhalten** - name: str

JSON-Beispiel:

```
* {"cmd: "init_user", "name: "test-user"}
```

```
-i {user: 100} *
```

#### **3.3.1.1.3 reset (ADMIN): die Maschine zurücksetzen** - user: User

Der Befehl entfernt den aktuellen Cocktail und setzt die Maschine wieder in den normalen Zustand zurück. Das ist nur beim Testen notwendig.

JSON-Beispiel:

```
* "cmd: "reset", user: 0 *
```

#### **3.3.1.1.4 reset\_error (ADMIN): gespeicherten Fehler zurücksetzen** - user: User

Der Befehl entfernt den aktuellen Fehler und macht im normalen Betrieb weiter. Das kommt meistens nur vor, wenn ein Rezept ein Problem hatte und die Maschine das Problem nicht beheben konnte.

JSON-Beispiel:

```
* "cmd: "reset_error", user: 0 *
```

#### **3.3.1.1.5 clean (ADMIN): reinigt die Maschine** - user: User

JSON-Beispiel:

```
* "cmd: "clean", user: 0 *
```

#### **3.3.1.1.6 restart (ADMIN): startet die Maschine neu** - user: User

JSON-Beispiel:

```
* "cmd: "restart", user: 0 *
```

**3.3.1.1.7 factory\_reset (ADMIN): setzt alle Einstellungen zurück** - user: User

JSON-Beispiel:

```
* "cmd: "factory_reset", user: 0 *
```

### 3.3.1.2 Rezepte definieren

**define\_recipe (USER): definiert ein neues Rezept**

- user: User - name: str - ingredients: List[Tuple[str, float]]

JSON-Beispiel:

```
* {"cmd: "define_recipe", user: 0, "name: "radler", ingredients: [{"beer", 250}, {"lemonade", 250}]}
```

**3.3.1.2.1 edit\_recipe (USER): editiert ein Rezept** - user: User - name: str  
- ingredients: List[Tuple[str, float]]

JSON-Beispiel:

```
* "cmd: edit_recipe", user: 0, "name: "radler", ingredients: [{"beer", 250}, {"lemonade", 250}] *
```

**3.3.1.2.2 delete\_recipe (USER): löscht ein Rezept** - user: User - name: str

JSON-Beispiel:

```
* {"cmd: "delete_recipe", user: 0, "name: "radler"}
```

### 3.3.1.3 Rezepte machen

Der Ablauf um ein Rezept zu machen ist:

1. Das Rezept mit 'queue\_recipe' in Auftrag geben. Das Rezept kommt in die Warteschlange. Die aktuellen Benutzer in der Warteschlange können mit dem Status 'user' ausgelesen werden.
2. Wenn die Maschine bereit ist, wird der Status 'state' auf 'waiting for container' gesetzt. Sobald ein Gefäß in der Maschine ist, kann das Rezept mit dem Befehl 'start\_recipe' gestartet werden.
3. Wenn das Rezept fertig ist, wird der Status 'state' auf 'cocktail done' gesetzt. Der Cocktail kann entnommen werden. Das Rezept wird mit 'take\_cocktail' beendet.
4. Die Maschine beginnt dann mit dem nächsten Rezept in der Warteschlange.

**3.3.1.3.1 queue\_recipe (USER): gibt ein Rezept in Auftrag** - user: User  
- recipe: str

JSON-Beispiel:

```
* {"cmd: "queue_recipe", user: 8858, "recipe: "radler"} *
```

**3.3.1.3.2 start\_recipe (USER): fängt das Rezept an, wenn die Maschine bereit ist** - user: User

JSON-Beispiel:

```
* {"cmd: "start_recipe", user: 8858} *
```

**3.3.1.3.3 cancel\_recipe (USER): bricht das aktuelle Rezept ab** - user: User

JSON-Beispiel:

```
* {"cmd: "cancel_recipe", user: 483} *
```

**3.3.1.3.4 take\_cocktail (USER): gibt Bescheid, dass der Cocktail entnommen wurde** - user: User

JSON-Beispiel:

```
* {"cmd: "take_cocktail", user: 483} *
```

**3.3.1.3.5 add\_liquid (USER): fügt Flüssigkeit zum aktuellen Rezept hinzu** - user: User - liquid: str - volume: float

JSON-Beispiel:

```
* {"cmd: "add_liquid", user: 0, "liquid: "water", "volume: 30} *
```

### **3.3.1.4 Pumpen**

**3.3.1.4.1 define\_pump (ADMIN): fügt Pumpe zu ESP hinzu** - user: User  
- liquid: str - volume: float - slot: int

JSON-Beispiel:

```
* {"cmd: "define_pump", user: 0, "liquid: "water", "volume: 1000, slot: 1} *
```

**3.3.1.4.2 define\_pumps (ADMIN): fügt mehrere Pumpe zu ESP hinzu**  
- user: User - liquid: str - volume: float - quantity: int

JSON-Beispiel:

```
* {"cmd: "define_pumps", user: 0, "liquid: "water", "volume: 0, "quantity: 3}
*
```

**3.3.1.4.3 edit\_pump (ADMIN): editiert eine Pumpe** - user: User - liquid: str - volume: float - slot: int

Nur die Flüssigkeit und das Volumen werden angepasst. Die Kalibrierung bleibt erhalten.

JSON-Beispiel:

```
* {"cmd: "edit_pump", user: 0, "liquid: "water", "volume: 1000, slot: 1} *
```

**3.3.1.4.4 refill\_pump (ADMIN): füllt Pumpe auf** - user: User - liquid: str - slot: int

JSON-Beispiel:

```
* {"cmd: "refill_pump", user: 0, "volume: 1000, slot: 1} *
```

### **3.3.1.5 automatische Kalibrierung**

siehe [Kalibrierung.md]()

**3.3.1.5.1 calibration\_start (ADMIN): Kalibrierung anfangen** - user: User

JSON-Beispiel:

```
* {"cmd: "calibration_start", user: 0} *
```

**3.3.1.5.2 calibration\_cancel (ADMIN): Kalibrierung abbrechen** - user: User

JSON-Beispiel:

```
* {"cmd: "calibration_cancel", user: 0} *
```

**3.3.1.5.3 calibration\_finish (ADMIN): Kalibrierung fertig** - user: User

JSON-Beispiel:

```
* {"cmd: "calibration_finish", user: 0} *
```

**3.3.1.5.4 calibration\_add\_empty (ADMIN): leeres Gefäß ist bereit** - user: User

JSON-Beispiel:

```
* {"cmd: "calibration_add_empty", user: 0} *
```

**3.3.1.5.5 calibration\_add\_weight (ADMIN): Gefäß ist mit einer Menge Wasser gefüllt** - user: User - weight: float (in Gramm)

JSON-Beispiel:

```
* {"cmd: "calibration_add_weight", user: 0, "weight: 100.0} *
```

### 3.3.1.6 manuelle Kalibrierung

**run\_pump (ADMIN): lässt die Pumpe für eine bestimmte Zeit laufen**

- user: User - slot: int - time: int

Die Zeit wird in Millisekunden angegeben.

JSON-Beispiel:

```
* {"cmd": "run_pump", "user": 0, "slot": 1, "time": 1000} *
```

**3.3.1.6.1 calibrate\_pump (ADMIN): kalibriert die Pumpe mit vorhandenen Messwerten**

- user: User - slot: int - time1: int - time2: int - volume1: float - volume2: float

Zur Kalibrierung müssen zwei Messwerte vorliegen, bei denen die Pumpe für eine unterschiedliche Zeit gelaufen ist. Daraus wird dann der Vorlauf und die Pumprate berechnet.

Die Zeiten werden in Millisekunden und die Flüssigkeiten in Milliliter angegeben.

JSON-Beispiel:

```
* {"cmd": "calibrate_pump", "user": 0, "slot": 1, "time1": 10000, "time2": 20000, "volume1": 15.0, "volume2": 20.0} *
```

**3.3.1.6.2 set\_pump\_times (ADMIN): setzt die Kalibrierungswerte für eine Pumpe**

- user: User - slot: int - time\_init: int - time\_reverse: int - rate: float

‘time\_init‘ ist die Vorlaufzeit und ‘time\_reverse‘ die Rücklaufzeit in Millisekunden. Normalerweise sollten diese Werte ähnlich oder gleich sein. Die Rate wird in mL/ms angegeben.

JSON-Beispiel:

```
* {"cmd": "set_pump_times", "user": 0, "slot": 1, "time_init": 1000, "time_reverse": 1000, "rate": 1.0} *
```

**3.3.1.6.3 tare\_scale (ADMIN): tariert die Waage**

- user: User

JSON-Beispiel:

```
* {"cmd": "tare_scale", "user": 0} *
```

**3.3.1.6.4 calibrate\_scale (ADMIN): kalibriert die Waage**

- user: User - weight: float

Das Gewicht wird in Milligramm angegeben.

JSON-Beispiel:

\* {"cmd: "calibrate\_scale", user: 0, "weight: 100.0} \*

### 3.3.1.6.5 set\_scale\_factor (ADMIN): setzt den Kalibrierungswert für die Waage

- user: User - factor: float

JSON-Beispiel:

\* {"cmd: "set\_scale\_factor", user: 0, "factor: 1.0} \*

## 3.3.2 ESP: Fehler

Mögliche Fehler, die vom ESP zurückgegeben werden können. (s. [Befehle.md]())

- 'init': ESP wird noch initialisiert
- 'ok': alles in Ordnung
- 'processing': Befehl wird geparsed (kein Fehler; kommt nur, wenn der Wert zu früh ausgelesen wird)
- 'unsupported': Befehl noch nicht implementiert
- 'unauthorized': Befehl nur für Admin verfügbar
- 'invalid json': ungültiges JSON
- 'message too big': JSON-Nachricht zu lang
- 'missing arguments': Argumente im JSON-Befehl fehlen
- 'unknown command': unbekannter Befehl
- 'command missing even though it parsed right': Befehl wurde im ESP fehlerhaft implementiert :)
- 'wrong comm channel': falscher Channel für den Befehl (Admin vs. User)
- 'invalid pump slot': ungültige Pumpe ausgewählt
- 'invalid pump quantity': ungültige Anzahl an Pumpen ausgewählt
- 'invalid volume': ungültige Menge (z.B. '-5')
- 'invalid weight': ungültiges Gewicht (z.B. '-5')
- 'invalid times': ungültige Zeit (z.B. '-5')
- 'insufficient amounts of liquid available': nicht genug Flüssigkeit vorhanden
- 'liquid unavailable': Flüssigkeit fehlt im ESP
- 'recipe not found': unbekanntes Rezept
- 'recipe already exists': Rezept mit dem gleichen Namen existiert bereits
- 'missing ingredients': Zutaten fehlen im Rezept
- 'invalid calibration data': Kalibrierungs-Werte sind ungültig (z.B. 2x die gleichen Werte)
- 'can't start recipe yet': Rezept ist noch nicht bereit
- 'can't take cocktail yet': Cocktail ist noch nicht fertig
- 'calibration command invalid at this time': Kalibrierungsbefehl zur falschen Zeit geschickt

## 3.3.3 Fehler: Kalibrierung

### 3.3.3.1 automatische Kalibrierung

Die automatische Kalibrierung kalibriert die Waage und die Pumpen relativ selbstständig. Es muss hauptsächlich nur das Gefäß geleert werden.

Der aktuelle Zustand kann im Status 'state' ausgelesen werden. Siehe [Zustände.md]() für eine Übersicht der Zustände.

Ablauf:

1. Kalibrierung mit 'calibration\_start' anfangen.



2. Die Maschine braucht ein leeres Gefäß. Der Zustand ist 'calibration empty container'. Sobald das Gefäß hingestellt wurde, kann mit dem Befehl 'calibration\_add\_empty' weitergemacht werden.
3. Die Maschine tariert die Waage. Danach braucht sie ein bekanntes Gewicht (z.B. 100ml Wasser). Das Gefäß sollte gefüllt werden. Der Zustand ist 'calibration known weight'. Sobald das Gewicht bereit ist, kann mit 'calibration\_add\_weight' weitergemacht werden.
4. Die Maschine kalibriert die Waage und bereitet die Pumpen vor. Das Gefäß muss wieder geleert werden und es wird mit 'calibration\_add\_empty' weitergemacht.
5. Es wird jetzt jede Pumpe zweimal gepumpt. Nach jedem Pumpen muss das Gefäß geleert werden. Der Zustand ist immer 'calibration empty container', wenn geleert werden muss. Danach wird mit 'calibration\_add\_empty' weitergemacht.
6. Wenn alle Pumpen fertig sind, berechnet die Maschine die neuen Werte. Danach ist die Kalibrierung fertig und der Zustand ist 'calibration done'. Das Gefäß kann entfernt werden. Der letzte Befehl ist 'calibration\_finish'.

### 3.3.3.2 manuelle Kalibrierung

**3.3.3.2.1 Ablauf der Kalibrierung für eine Pumpe** 1. Eine Messung kann durchgeführt werden, indem die Pumpe manuell mit dem Befehl 'run\_pump' angesteuert wird. Beispielsweise für 10s:

```
* "cmd: "run_pump", user: 0, blot: 1, "time: 10000 *
```

Danach muss das Volumen, das insgesamt gepumpt wurde, gemessen werden.

2. Dann muss der Vorgang nochmal mit einer anderen Zeit wiederholt werden, z.B. für 20s:

```
* "cmd: "run_pump", user: 0, blot: 1, "time: 20000 *
```

3. Anschließend kann die Pumpe mit 'calibrate\_pumps' kalibriert werden. Wenn die Volumen z.B. 15mL und 20mL waren:

```
* "cmd: "calibrate_pump", user: 0, blot: 1, "time1: 10000, "time2: 20000, "volume1: 15.0, "volume2: 20.0 *
```

Die berechneten Werte werden im Debug-Log auch angezeigt.

4. Der Füllstand der Pumpe ist nach dem Kalibrieren üblicherweise in einem falschen Zustand. Am besten wird der Füllstand mit 'refill\_pump' zurückgesetzt.

5. Die Zeiten werden auf dem ESP gespeichert.

6. Alternativ können die Werte auch direkt mittels 'set\_pump\_times' gesetzt werden.

**3.3.3.2.2 Ablauf der Kalibrierung für die Waage** 1. Der Befehl ‘tare\_scale’ tariert die Waage neu aus, wenn (außer dem Gefäß) nichts vorhanden ist. Die Tarierung wird auch automatisch vor jedem Rezept ausgeführt.

2. Die Waage kann mit einem bekannten Gewicht und dem Befehl ‘calibrate\_scale’ kalibriert werden. Wenn das Gewicht z.B. 100mg ist:

\* "cmd: "calibrate\_scale", user: 0, "weight: 100.0 \*

3. Alternativ kann der Skalierungsfaktor auch manuell mit ‘set\_scale\_factor’ gesetzt werden.

### **3.3.4 ESP: Services**

#### **3.3.4.1 Unterstützte Services**

**3.3.4.1.1 Allgemein** Der ESP nutzt kein Pairing. Stattdessen wird jedem Benutzer eine User-ID zugewiesen, die dann bei den Befehlen mit angegeben werden muss.

Um allgemeine Werte (z.B. den Zustand der Pumpen) auszulesen, reicht es, die entsprechenden Characteristics auszulesen. Die Characteristics unterstützen Notifications, damit sie nur neu ausgelesen werden müssen, wenn sie sich geändert haben. Die verfügbaren Services und Characteristics stehen im Abschnitt Status-Services.

Zur Kommunikation wird ein Kommunikations-Service benutzt. Der grundlegende Aufbau ist:

1. ein Write in die passende Characteristic, zu der eine Nachricht gesendet werden soll 2. der ESP sendet die Antwort mit Read+Notify zurück

Nachrichten sind dabei immer JSON-Objekte (in UTF8-Kodierung), üblicherweise ein Map. Das Kommunikationsprotokoll ist:

1. Der Client registriert, dass er eine Notification auf der Characteristic erhalten will. 2. Der Client schreibt die Nachricht in die Characteristic. 3. Wenn der Server die Nachricht erhalten und verarbeitet hat, schreibt er eine Antwort in die Characteristic. Der Client bekommt eine Notification und kann die Antwort auslesen.

Die meisten Nachrichten senden nur als Antwort, ob es einen Fehler gab, und wenn ja, welchen. Der genaue Aufbau der Nachrichten steht in [(Befehle.md)].

Um mit dem ESP zu kommunizieren, muss ein Client zuerst eine User-ID erhalten. Die ID ist dauerhaft gültig und muss für einen Benutzer nur einmal registriert werden. Die ID funktioniert auch bei zukünftigen Verbindungen noch.

Um eine ID zu erhalten, sendet der Client den Befehl ‘init\_user’. Eine typische Nachricht ist:

\* {"cmd: "init\_user", "name: "Jane"} \*

Die Antwort ist dann:

\* üser: 100 \*

Die ID ist also 100 und kann dann in anderen Befehlen mit angegeben werden.

**3.3.4.1.2 ID-Service** - Name: Cocktail Machine ESP32 - UUID des Service: 8ccbf239-1cd2-4eb7-8872-1cb76c980d14 - UUID des Namens: c0605c38-3f94-33f6-ace6-7a5504544a80

**3.3.4.1.3 Kommunikations-Service** - UUID des Service: dad995d1-f228-38ec-8b0f-593953973406 - UUID user: eb61e31a-f00b-335f-ad14-d654aac8353d - UUID admin: 41044979-6a5d-36be-b9f1-d4d49e3f5b73

**3.3.4.1.4 Status-Services** - UUID des Service: 0f7742d4-ea2d-43c1-9b98-bb4186be905d

**3.3.4.1.5 Pumpen** - UUID Characteristic: 1a9a598a-17ce-3fcd-be03-40a48587d04e

Wert: Map aller verfügbaren Pumpen und deren Füllstand und Kalibrierung.  
"cal", ist dabei ein Array aus [Rate, Zeit davor, Zeit danach], wenn die Pumpe kalibriert ist, und sonst ein leeres Array.

Beispiel:

\* {"1":{"liquid":"water","volume:1000.0,"cal:[0.0,1000,1000]}} \*

**3.3.4.1.6 Flüssigkeiten** - UUID Characteristic: fc60afb0-2b00-3af2-877a-69ae6815ca2f

Wert: Map aller verfügbaren Flüssigkeiten und deren Menge

Beispiel:

\* "beer: 200, "lemonade: 2000, örange juice: 2000 \*

**3.3.4.1.7 Zustand** - UUID Characteristic: e9e4b3f2-fd3f-3b76-8688-088a0671843a

Wert: Der aktuelle Zustand der Cocktail-Maschine und was sie macht.

Beispiel:

\* "ready\*

**3.3.4.1.8 Rezepte** - UUID Characteristic: 9ede6e03-f89b-3e52-bb15-5c6c72605f6c

Wert: alle gespeicherten Rezepte und deren Namen

Beispiel:

\* [{"name: "radler", "liquids: [{"beer", 250}, {"lemonade", 250}], "name: Bpe-zi", "liquids: [{"cola", 300}, [örange juice", 100]]] \*

#### **3.3.4.1.9 Cocktail** - UUID Characteristic: 7344136f-c552-3efc-b04f-a43793f16d43

Wert: Der Inhalt des aktuellen Cocktails, der gemischt wird.

Beispiel:

\* "weight: 500.0, "content: [{"beer", 250}, {"lemonade", 250}] \*

#### **3.3.4.1.10 Benutzer in der Warteschlange** - UUID Characteristic: 2ce478ea-8d6f-30ba-9ac6-2389c8d5b172

Wert: alle Benutzer in der Warteschlange, für die ein Cocktail gemacht wird

Wenn kein Benutzer aktiv ist, ist der Wert '[]'.

Beispiel:

\* [1, 4, 2] \*

#### **3.3.4.1.11 Letzte Änderung** - UUID Characteristic: 586b5706-5856-34e1-ad17-94f840298816

Wert: Timestamp der letzten Änderung

Wenn sich der Timestamp nicht geändert hat, sind die verfügbaren Rezepte und Zutaten noch die gleichen.

Der Timestamp ist ein interner Wert des ESP und hat keinen Bezug zur echten Zeit.

Beispiel:

\* 275492 \*

#### **3.3.4.1.12 Waage** - UUID Characteristic: ff18f0ac-f039-4cd0-bee3-b546e3de5551

Wert: Zustand der Waage

Beispiel:

\* "weight:0.0,"calibrated:true \*

#### **3.3.4.1.13 Fehler** - UUID Characteristic: 2e03aa0c-b25f-456a-a327-bd175771111a

Wert: aktueller Fehler (falls vorhanden)

Beispiel:

invalid volume"

### **3.3.5 ESP: Zustände**

Der aktuelle Zustand kann im Status-Service ausgelesen werden.

#### **3.3.5.1 Rezepte**

- 'ready': Maschine ist bereit einen Cocktail zu machen

- ‘waiting for container’: Maschine wartet auf ein leeres Gefäß
- ‘mixing’: Maschine macht einen Cocktail
- ‘pumping’: Maschine pumpt Flüssigkeiten
- ‘cocktail done’: Cocktail ist fertig zubereitet und kann entnommen werden. Danach sollte ‘take\_cocktail’ ausgeführt werden.

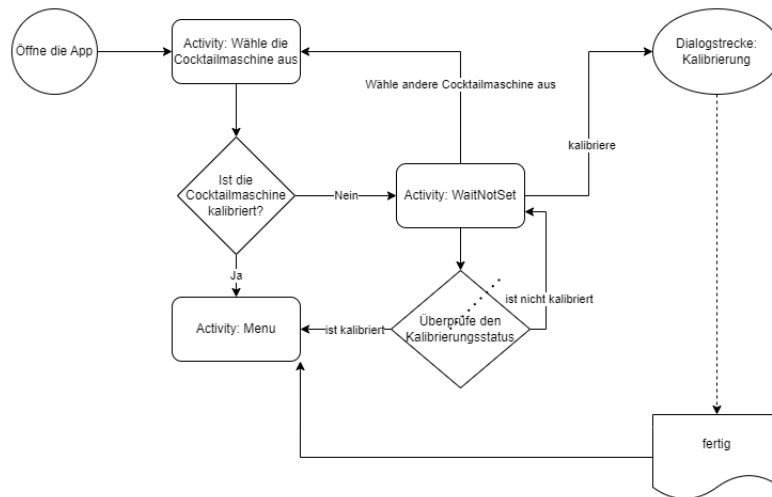
### 3.3.5.2 Kalibrierung

- ‘ready’: Maschine ist bereit für eine Kalibrierung
- ‘calibration empty container’: Kalibrierung wartet auf ein leeres Gefäß. Es sollte ‘calibration\_add\_empty’ ausgeführt werden.
- ‘calibration known weight’: Kalibrierung wartet auf ein Gewicht. Es sollte ‘calibration\_add\_weight’ ausgeführt werden.
- ‘calibration pumps’: Kalibrierung pumpt Flüssigkeiten
- ‘calibration calculation’: Kalibrierung berechnet die Werte
- ‘calibration done’: Kalibrierung fertig. Es sollte ‘calibration\_finish’ ausgeführt werden.

### 3.3.6 GUI: Öffnung der App

Die Öffnung der App resultiert erstmal darin die verfügbaren Cocktailmaschinen zu listen und den Nutzer wählen zu lassen. Ist die gewählte Cocktailmaschine kalibriert, wird der Nutzer sofort in das Menü versetzt. Ist die Maschine nicht kalibriert, wird der Nutzer in einen Wartebereich geschoben. Dort hat der Nutzer die Wahl:

- den Kalibrierungsstatus erneut abzufragen und bei positiver Bestätigung in dem Menü zu landen,
- die Cocktailmaschine zu wechseln
- oder die Cocktailmaschine zu kalibrieren.



### 3.3.7 GUI: Keine Kalibrierung

Drei mögliche Aktionen:

- anmelden
- falsche Passwort: zurück in die Activity
- und einleiten der automatischen Kalibrierung: Dialog: Kalibrierung
- wähle eine andere Cocktailmaschine aus: Cocktailmachinewahlactivity öffnet.
- laden (Überprüfen, ob ein Admin die Kalibrierung vorgenommen hat)
  - Bei Erfolg (Admin hat die Kalibrierung getätigt. ): Hauptmenü öffnet
  - Bei Misserfolg: Toast "Cocktailmaschine ist noch nicht bereit."

Nach der Auswahl eines der Möglichkeiten werden die Button auf nicht verfügbar gestellt, damit die Button beim Laden zwischen den Dialogen nicht angeklickt werden können.

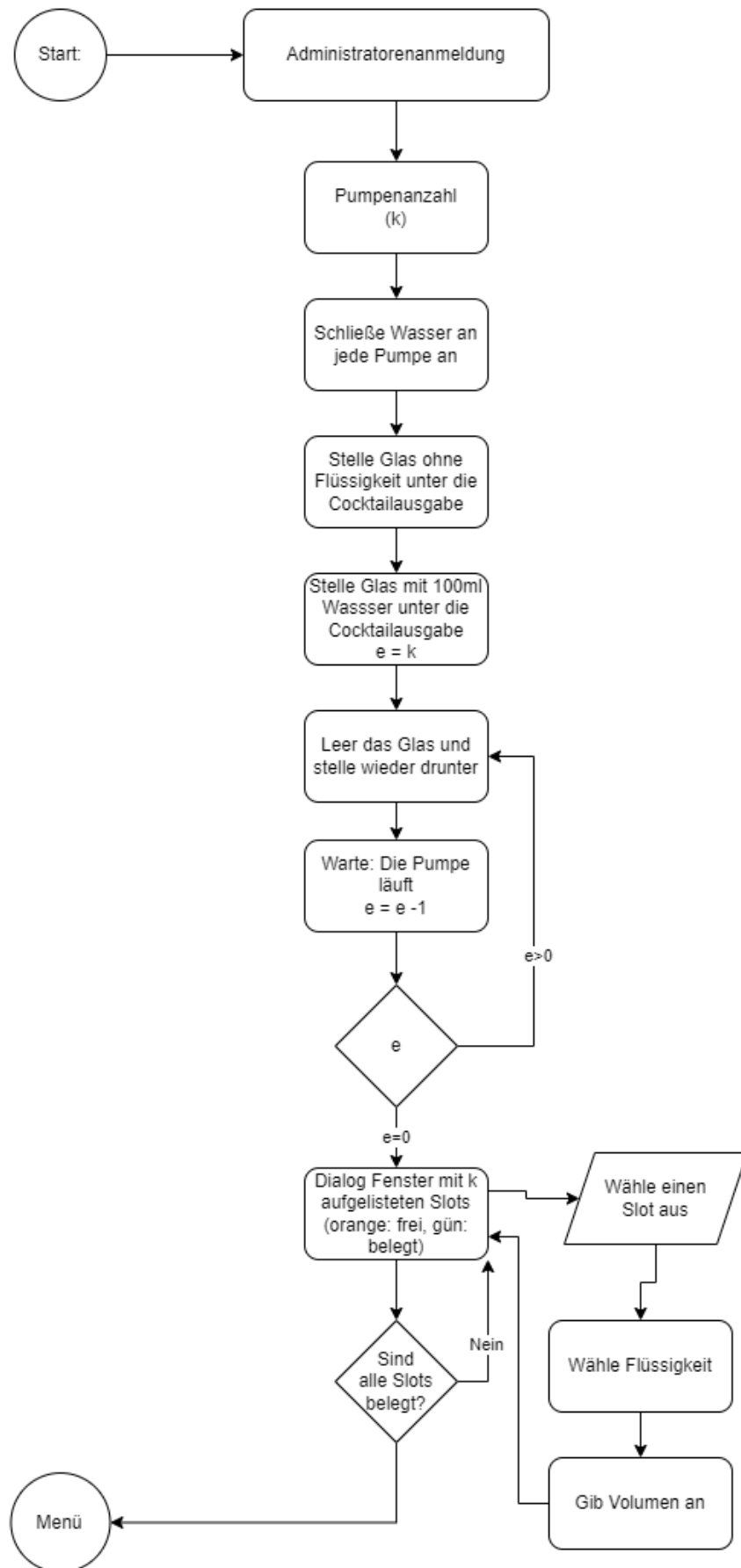
### 3.3.8 GUI: Dialog: Anmelden

Dialogfenster:

Passwort angeben ("admin"). Gelange in den angemeldeten Zustand.



### 3.3.9 GUI: Dialog: Automatische Kalibrierung





1. Gebe die Anzahl (k) der Pumpen an.
  - (a) Toast "Es lädt"
2. Ok-Fenster: Schließe Wasser an die Pumpen.
3. Ok-Fenster: Glas ohne Flüssigkeit
4. Ok-Fenster: Glas mit 100ml Wassre
5. Ok-Fenster: Leere das Glas
6. Warte-Fenster: Pumpe läuft.  $k=k-1$ 
  - (a) zu 5.: falls  $k \neq 0$
  - (b) zu 7.: sonst
7. Dialog mit Liste von k Pumpen sind mit einer Slotnummer markiert und orange hinterlegt, wenn noch keine Flüssigkeit hinterlegt ist, und grün wenn eine Flüssigkeit hinterlegt ist.
8. Wähle orangene Pumpe.
9. Wähle Flüssigkeit.
10. Gib Volumen an:
  - (a) zu 7.: wenn noch nicht alle Flüssigkeiten ausgewählt wurden
  - (b) zum Hauptmenü: sonst

### **3.3.10 GUI: Darstellung Liste**

Liste von Elementen

Titel: Rezept, Zutaten, Serviervorschläge, Pumpen

Haus: Gehe zum Hauptmenü.

drehender Pfeil: Lade die Datenbank neu!

#### **3.3.10.1 Pumpen**

Statt dem Namen der Pumpen sind die Namen der zugeordneten Zutaten und die Slotnummer sichtbar.

#### **3.3.10.2 Rezepte**

Plus Zeichen: Gehe zur Erstellung eines Rezeptes.

### 3.3.11 GUI: Anzeigen einzelner Elemente

Für alle Elemente gilt: unter dem Titel ist eine Zeile mit drei Buttons. Der erste und linke Button ist ein Stift und öffnet die Änderungsseite für das Element. Der zweite und mittige Button ist ein drehender Pfeil. Er lädt die gesamte Datenbank neu und setzt die Seite neu auf. Der dritte und rechte Button ist eine Liste und öffnet die Liste der Elemente. Wird also eine Zutat angezeigt, so öffnet sich die Zutatenliste. Im weiteren wird von dieser Zeile als Menüzeile in der Einzelansicht gesprochen.

#### 3.3.11.1 Zutat

Als Titel steht der Name der Zutat. Darunter kommt die Menüzeile. Unter der Menüzeile ist eine Gruppe von drei Sternen. Die Sterne haben die Farbe der Zutat. Darunter wird entweder angezeigt, dass die Zutat verfügbar oder nicht verfügbar ist. Dabei wird innerhalb einer Zeile links ein grüner Punkt für verfügbar und dann neben der Text "verfügbar" angezeigt oder ein graues Kreuz und der Text "nicht verfügbar".

In der nächsten Zeile steht, ob die Zutat alkoholisch ist oder nicht. Dabei wird bei der Angabe alkoholisch eben der Text links angezeigt mit einem roten Warndreieck auf der rechten Seite oder "nicht alkoholisch" mit einem grauen Warndreieck. Die Volumenanzeige gibt die Milliliterangabe an.

#### 3.3.11.2 Serviervorschlag

Als Titel steht der Name des Serviervorschlags. Darunter ist die Menüzeile und da runter die Beschreibung.

#### 3.3.11.3 Pumpe

Als Titel steht Slot und die Slotnummer. Die zugehörige Zutat wird angezeigt und das Volumen in Millilitern. Ein großer Button "Lass die Pumpe laufen".

**3.3.11.3.1 Dialog: Pumpe laufen lassen** Hier steht eine Aufforderung zur Milliliterangabe, um die Pumpe laufen zu lassen. Bei Bestätigung wird die Pumpe solange laufen gelassen.

#### 3.3.11.4 Rezepte

Die Verfügbarkeitsanzeige ist ein grüner Punkt, wenn es verfügbar ist und rot, wenn es nicht verfügbar ist. Die Alkoholgehaltsanzeige ist ein Dreieck mit weißem Ausrufzeichen. Wenn das Dreieck rot ist, hat das Rezept alkoholische Zutaten. Wenn es grau ist, ist es nicht alkoholisch. Außerdem werden die Zutaten mit Milliliterangabe gelistet. Jede Zutat führt zur jeweiligen Zutatendarstellung. Die Serviervorschläge werden genauso gelistet. Zu letzt ist ein großer Button "Mix den Cocktail" führt zu, Dialog: Mixen.

**3.3.11.4.1 Dialog: Mixen/Activity:Simulation des Befüllens** Beginnt mit einem Countdown mit der Anzahl der noch zu mixenden Cocktails bis zum eigenen Cocktail. Wenn der Nutzende dann dran ist, kommt die Aufforderung das eigene Glas runterzustellen, was bestätigt werden muss. Gehe zur Darstellung (Activity): Simulation des Befüllens, wenn das Mixen abgeschlossen ist, wird mit einem Dialog aufgefordert, den Cocktail abzuholen. Nach der Bestätigung des Abholens wird das Hauptmenü geöffnet.

### **3.3.12 GUI: Ändern oder Hinzufügen**

Für alle Elemente gilt:

Möchte man ein neues Element hinzufügen. Geht man in die entsprechende Liste und wählt den Floatingbutton in der rechten unteren Ecke mit dem Plus-symbol aus.

Jedes Element braucht einen Namen. Unter dem Titel gibt es ein Eingabefeld. Als Aufforderung steht in dem Feld in grauer Schrift "Füge einen Namen ein!". Wählt man das Feld aus, öffnet sich die Tastatur. Jetzt kann der Name eingegeben werden. Der eingegebene Name ist in schwarzer Schrift und nachdem ersten Einfügen eines Buchstabens ist die graue Aufforderung nicht mehr sichtbar. Handelt es sich hierbei allerdings um eine Änderung an einer bestehenden Element, dann ist zunächst die nicht die Aufforderung zu sehen, sondern direkt der Name in schwarzer Schrift. Entfernt man alle Buchstaben in dem Feld, egal ob es sich um eine Änderung oder eine Neuerstellung handelt. Dann erscheint wieder die graue Aufforderung.

Unter all den Eingabefeldern. Stehen zwei Button "Abbrechen" und "Speichern". Wählt man den Abbruch gelangt man in die Liste der Elementen. Wählt man den Speichervorgang. So wird die Detailansicht der neu erstellten Zutat sichtbar.

Der Speichervorgang wird abgebrochen, sollte beim Auslösen das Namensfeld leer sein.

#### **3.3.12.1 Zutat**

Daraufhin öffnet sich die AddAktivität. Der Titel heißt "SZutat".

Unter dem Namensfeld ist ein Switchbutton. Ist dieser aktiviert, steht direkt unter dem Button ein Feld "alkoholisch" mit einem roten Warndreieck davor. Ist der Button nicht aktiviert, ist das Warndreieck grau und es steht "nicht alkoholisch" da. Bei der Neuerstellung ist zunächst der Switchbutton im inaktiven Zustand und das Feld mit dem Warndreieck ist nicht sichtbar. Erst nachdem der Nutzende mit dem Switchbutton interagiert wird die Warndreieckzeile sichtbar. Bei der Änderung ist das Warndreieck direkt sichtbar und der Switchbutton ist im aktiven oder inaktiven Zustand je nachdem ob die zu ändernde Zutat alkoholisch oder nicht ist.

Unter der Alkoholgehaltabfrage steht eine weitere Aufforderung "Ändere die Farbe!". Sie steht in einem Button. Wählt man diesen aus öffnet sich ein Dialog, in dem eine Farbe ausgewählt werden kann. Dieser Dialog kann abgebrochen werden mit "Abbruch". Mit dem "Speichern" wird der Zutat eine Farbe zu geordnet. Handelt es sich eine Neuerstellung und das erste Anklicken der Farbauswahl, so ist eine zufällige Farbe hinterlegt. Wurde bereits eine Farbe ausgewählt oder es ist eine Änderung, dann ist die derzeit zugeordnete Farbe der Ausgangspunkt. Der Button ändert seine Farbe mit der ausgewählten Farbe. Ist noch keine ausgewählt worden bei der Neuerstellung, so hat der Button die Standardbuttonfarbe.

### **3.3.12.2 Serviervorschlag**

Als Titel steht "Serviervorschlag". Unter dem Namensfeld steht ein großes editierbares Textfeld. Dort steht in schwarz entweder die vorhandene Beschreibung bei einer Änderung oder in schwarz die eingegebene Beschreibung. Ist es eine Neuerstellung oder die gesamte Textfeldeingabe wurde entfernt (also die schwarzen Buchstaben), dann ist in grauen Buchstaben der Text "Füge eine Beschreibung ein." zu lesen, der so gleich wieder nicht sichtbar wird, sollte auch nur ein Buchstabe in diesem Feld eingegeben werden. Sind die Titelangabe oder die Beschreibungsangabe leer, dann wird das Speichern abgebrochen und eine Toastaufforderung zur Ergänzung angezeigt.

### **3.3.12.3 Rezepte**

Unter dem Namensfeld steht für die Zutaten und die Serviervorschläge je eine Aufforderung des Einfügens mit einem grünen Plus daneben. Das Plus löst als Button einen Dialog aus, indem je entweder die Namen der verfügbaren Zutaten oder die der Serviervorschläge gelistet sind. Nach der Wahl einer Zutat wird in einem Dialog die Milliliterangabe gemacht und die Zutat mit Volumenangabe unter der Aufforderung hinzugelistet. Genauso sammeln sich die Namen der gewählten Serviervorschläge.

### **3.3.12.4 Pumpe**

Es gibt kein Namensfeld. Es gibt zwei verpflichtende Felder: Ein Textfeld zur Angabe des Zutatennamens und der des Volumens. Beide müssen mit gültigen Werten ausgefüllt sein, um gespeichert zu werden.

## **3.3.13 GUI: Menü**

Im Menü werden die wichtigsten Funktionen dem Nutzer angezeigt. Übersicht

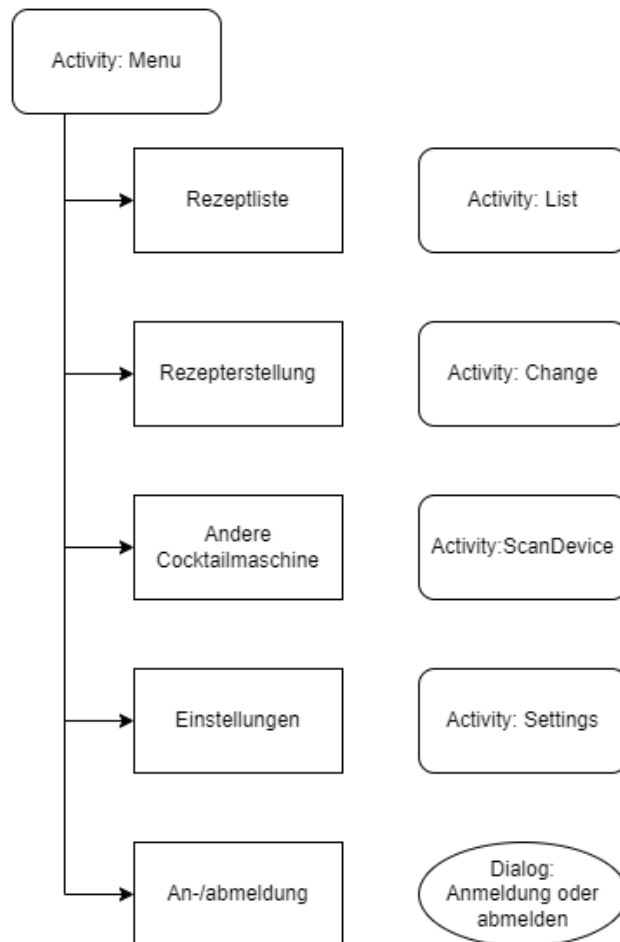


Abbildung 3.1: Menü

### 3.3.14 GUI: Einstellung

#### 3.3.14.1 Die Haupteinstellung:

Übersicht

#### 3.3.14.2 Die Cocktailmaschineneinstellungen:

(Erreichbar nur im **angemeldeten** Zustand) Übersicht

#### 3.3.14.3 Die Waagenkalibrierung:

(Erreichbar nur im **angemeldeten** Zustand) Übersicht

#### 3.3.14.4 Die Pumpenkalibrierung:

(Erreichbar nur im **angemeldeten** Zustand) Übersicht

### 3.3.15 Datenbank

Die Datenbank verwendet die von Android zur Verfügung stehende Standardimplementierung.

Die Datenbank ist über ein paar simplen Designentscheidungen implementiert. Es gibt eine Standardimplementierung von Grundfunktionen für jede Tabelle die als Oberklasse aller Tabellen fungiert. Die Unterklassen werden dann um

Tabelle 3.1: Menü

Symbol	Nächste Activity	Funktion	Sichtbar	ausgelöste Reaktion
Liste	ListActivity	Liste der Rezepte	immer	Aktivitätenwechsel
Liste mit Plusymbol	AddActivity AddActivity	Erstellung eines neuen Rezepts	immer	Aktivitätenwechsel
Bluetooth mit Lupe	BluetoothScan	Suche eine neue Cocktailmaschine	immer	Aktivitätenwechsel
Zahnrad	Settingsactivity	Einstellungs- optionen	immer	Aktivitätenwechsel
Pfeil in eine Tür	-	Administratoren- anmeldung	Nur im abgemeldeten Zustand	Zeigen eines Anmeldedialogs, Wechsel in den angemeldeten Zustand
Pfeil aus einer Tür	-	Administratoren- abmeldung	Nur im angemeldeten Zustand	Wechsel in den abgemeldeten Zustand

die spezifische Implementierungen erweitert. Eine Standardabfrage einer Tabelle ist die Frage nach dem Namen oder den Spaltennamen. Die Abfrage nach Elementen mit einer bestimmten ID, etc.

Die für diese Anwendung benötigten Tabellen sind.

- der Serviervorschlag (Titel, Beschreibung)
- die Zutat (Titel, Farbe, Verfügbarkeit)
- die Pumpe (Slotnummer, Zutat Referenz, Volumen)
- die Rezept (Titel, Verfügbarkeit)
- die Information ob eine Zutat in einem Rezept ist (Rezept Referenz, Zutat Referenz, Volumen)
- die Information ob ein Serviervorschlag in einem Rezept ist (Rezept Referenz, Serviervorschlag Referenz)

Gebündelt ist der Zugriff auf die Tabellen in einer Klasse zur Erleichterung. Dort ist dann eine Tabelle als statisches Objekt hinterlegt.

Jede Tabelle hat ein Format seiner Objekte. Diese Datenbankelemente müssen programmatisch hinterlegt werden und die Tabellenspalten abbilden.

Um zu erkennen, wo und welche Fehler auftreten, wie zum Beispiel die Doppelnennung einer Zutat in einem Rezept, gibt es eine Reihe von spezifische Exception.

- die Ablehnung des Zugriffs, weil der App-Nutzer kein Administrator ist
- die fehlende Zutat in einer Pumpe
- die Spalte existiert nicht
- die Tabelle existiert nicht
- die Datenbank ist nicht erreichbar
- die Zutat ist bereits im Rezept erwähnt
- die Zutat ist nicht im Rezept erwähnt, obwohl explizit nach ihr gefragt wird
- die Pumpe ist leer, bzw. wurde gerade leer gemacht

Es gibt 4 Klassen, die auf die Datenbank zugreift.

- eine Klasse, um ein Element zu ändern oder hinzuzufügen
- eine Klasse, um ein Element zu löschen
- eine Klasse, um Elemente abzufragen
- eine Klasse für die Extrabefehle der Datenbank, wie die Erstellung und Löschung von Tabellen oder das Einlesen vorgefertigter Rezepte.

- die Datenbankverbindung, welche die Standardimplementierung einer SQL-Datenbank darstellt
- eine Hilfsklasse, die beim Auslesen hilft in den unterschiedlichen Androidversionen

### **3.3.16 DB: Zustände**

Es gibt mehrere Zustände, die die Cocktailmaschine ein nehmen kann. Programmtechnisch ist es sinnvoll diese in Enums zu verpacken, wobei es mehrere Enum-Pakete gibt. Die Standardzustände, wie das Mixen eines Cocktails und die Zustände während bzw. kurz vor der Kalibrierung und die auftretenden Fehler. In den Enums werden alle Zustände gelistet und die Bluetoothabfrage und die Anzeige via eines Dialogs ermöglicht.

### **3.3.17 DB: Administrator**

Es gibt neben den ESP-Zustände noch einen weiteren App internen Zustand und zwar den des Nutzers. Dieser kann ein User oder ein Admin sein, wobei Admin die Abkürzung für Administrator ist. Innerhalb der App wird nach der Anmeldung des Nutzers die erweiterten Funktionen freigeschaltet, wie die Kalibrierung der Maschine.

### **3.3.18 Schnittstellen**

Es gibt 5 Hauptschnittstellen, eine für Pumpen, für die Rezepte, die Zutaten, die Serviervorschläge und die Cocktailmaschine selbst. Die ersten vier Komponenten von Pumpen, Rezepten, Zutaten und Serviervorschläge ähneln sich. Die Schnittstelle dient als Dataformat für ein einzelnes Objekt wie ein Rezept mit Namen, Zutaten und ID oder in statischen Funktionen zur Abfrage der Datenbank. Finde zum Beispiel alle Zutaten, die "Wodim Namen tragen. Die Schnittstelle der Cocktailmaschine diene zunächst als Dummy für die Bluetoothverbindung, weil diese noch nicht implementiert war und trotzdem aufgerufen werden sollte, um die GUI die Cocktailmaschinenfunktionen simulieren zu lassen und diene im Anschluss als Oberfläche die Bluetoothverbindung einzelner Cocktailmaschinenfunktionen zu erstellen, sowie die automatische Kalibrierung zu starten. Dabei ist zum Beispiel die Anmeldung oder der Start des Reinigungsprogramms eine Variante.



Tabelle 3.2: Haupteinstellung

<b>Element</b>	<b>Nächste Activity</b>	<b>Funktion</b>	<b>Sichtbar</b>	<b>ausgelöste Reaktion</b>
Anmelden	- - -	Administratorenanmeldung: Zeigen eines Anmeldedialogs, Wechsel in den angemeldeten Zustand	Nur im abgemeldeten Zustand	Funktion
Abmelden	- -	Administratorenabmeldung: Wechsel in den abgemeldeten Zustand	Nur im angemeldeten Zustand	Funktion
Pumpe	ListActivity	Liste der Pumpen	Nur im angemeldeten Zustand	Aktivitätenwechsel
Zutaten	ListActivity	Liste der verfügbaren Zutaten	immer	Aktivitätenwechsel
Zutaten(Alle)	ListActivity	Liste der Zutaten	Nur im angemeldeten Zustand	Aktivitätenwechsel
Rezepte	ListActivity	Liste der verfügbaren Rezepte	immer	Aktivitätenwechsel
Rezepte(Alle)	ListActivity	Liste der Rezepte	Nur im angemeldeten Zustand	Aktivitätenwechsel
Serviovorschläge	ListActivity	Liste der Serviovorschläge	immer	Aktivitätenwechsel
Synchronisieren (Bluetooth)	- -	Synchronisiere Rezepte und die Pumpen.	immer	Funktion
Scannen (Bluetooth)	ScanActivity	Scanne nach der Cocktailmaschine	immer	Aktivitätenwechsel
Cocktailmaschinen- einstellung	MachineSettings	weitere Einstellungen	Nur im angemeldeten Zustand	Aktivitätenwechsel
Komplett neuladen (Datenbank)	-	Synchronisiere Pumpenstand und damit die Verfügbarkeit.	immer	Funktion
Füge vorbereitete Rezepte hinzu (Datenbank)	-	Lade Rezepte aus Json und CSV-Dateien.	immer	Funktion

Tabelle 3.3: Cocktailmaschineneinstellungen

<b>Element</b>	<b>Nächste Activity</b>	<b>Funktion</b>	<b>ausgelöste Reaktion</b>
Neukalibrierung (Kalibrierung)	WaitNotSet	Setze die Kalibrierung zurück und kalibriere die Cocktailmaschine neu. Geh zu WaitNotSet.	Aktivitätenwechsel
Wagenkalibrierung (Kalibrierung)	ScaleSettingsActivity	weitere Einstellungen	Aktivitätenwechsel
Pumpenkalibrierung (Kalibrierung)	PumpSettingsActivity	weitere Einstellungen	Aktivitätenwechsel
Statusabfrage (Zustand)	-	Zeige in einem Dialog den derzeitigen Zustand an, wie von Freya beschrieben.	Funktion
Reinigung (Zustand)	-	Schickt Reinigungsbefehl	Funktion
Neustart (Neustart)	-	Schickt Neustartbefehl. (Kein Zurücksetzen!)	Funktion
Werkseinstellungen (Neustart)	WaitNotSet	Setze die Kalibrierung zurück. Geh zu WaitNotSet.	Aktivitätenwechsel

Tabelle 3.4: Waagenkalibrierung

<b>Element</b>	<b>Nächste Activity</b>	<b>Funktion</b>	<b>ausgelöste Reaktion</b>
Kalibriere	-	Setze das Gewicht mit Hilfe eines Dialogs. Schicke Gewichtsangabe.	Funktion
Tariere	-	Schicke Tariierungsbefehl.	Funktion
Skaliere	-	Setze Saklierungsfaktor mit des Dialogs.	Funktion
derzeitiges Gewicht	-	Zeige in einem Dialog das derzeitiges Gewicht an.	Funktion

Tabelle 3.5: Pumpenkalibrierung

<b>Element</b>	<b>Nächste Activity</b>	<b>Funktion</b>	<b>ausgelöste Reaktion</b>
Automatische Kalibrierung	WaitNotSet	Setze die Kalibrierung zurück und kalibriere die Cocktaillmaschine neu. Geh zu WaitNotSet.	Aktivitätenwechsel
Setze Pumpzeiten (Kalibrierung)	- - - -	Dialog mit vier Eingabefeldern: Zeit1, Zeit2, Volumen1, Volumen2, wie von Freya beschrieben. Wird für alle Pumpen gleichzeitig verwendet.	

---

# 4

## Qualitätssicherung

---

### 4.1 Testplan

---

### 4.2 Testprotokoll

---

# 5

## Abschlussbericht

---

### 5.1 Zusammenfassung

---

### 5.2 Beispielanwendungen

---

### 5.3 Benutzerdokumentation

---

### 5.4 Entwicklerdokumentation

---

### 5.5 Erfahrungsbericht







---

# Literaturverzeichnis

---

# Tabellenverzeichnis

3.1	Menü . . . . .	24
3.2	Haupteinstellung . . . . .	27
3.3	Cocktailmaschineneinstellungen . . . . .	28
3.4	Waagenkalibrierung . . . . .	29
3.5	Pumpenkalibrierung . . . . .	30

---

# Abbildungsverzeichnis

3.1	Menü . . . . .	23
-----	----------------	----

---

# Abkürzungsverzeichnis

**DNA** Desoxyribonukleinsäure

**A** Adenin

**T** Thymin

**G** Guanin

**C** Cytosin

**AT** Adenin und Thymin

**GC** Guanin und Cytosin

**MESA** Mosla Error Simulator

**URL** Uniform Resource Locator

**API** Anwendungsprogrammierschnittstelle

**TIFF** Tagged Image File Format

**PNG** Portable Network Graphics

**RS-Code** Reed-Solomon-Code

---

# Listings

---

# Appendix

---

# Eidesstattliche Erklärung

Ich erkläre, dass ich meine Thesis *Coctailmaschine* selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe, und dass ich alle Stellen, die ich wörtlich oder sinngemäß aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner Prüfungsbehörde vorgelegen.

Ich versichere, dass die eingereichte schriftliche Fassung der auf dem beigefügten Medium gespeicherten Fassung entspricht.

Marburg,  
den  
5. März  
2024

---

Johanna Reidt, Freya Dorn, Amir Rabieyan Nejad und Phillip Wieber