*TAU INF202 Software Engineering Projektarbeit*

Pflichtenheft

Projekt Dokumentation Version: 2023.v3 Status: Abgegeben

ORDER CORNER

Verantwortliche/r:

ANIL CEMAL YAŞAR < [e200503022@stud.tau.edu.tr](mailto:e200503022@stud.tau.edu.tr) > BARIŞ CAN FIRAT < [e180503009@stud.tau.edu.tr](mailto:e180503009@stud.tau.edu.tr) >

Stakeholder:

Fulya Yenilmez < [fulya.yenilmez@tau.edu.tr](mailto:fulya.yenilmez@tau.edu.tr) > DI. Ömer Karacan < [omer.karacan@tau.edu.tr](mailto:omer.karacan@tau.edu.tr) >

INHALTSVERZEICHNIS

# Ausgangssituation und Ziele

1. **Gesamtarchitektur**

# Funktionale Anforderungen

1. **Nichtfunktionale Anforderungen**

# Referenzen

Gerade in Zeiten, in denen die Technologie rasante Fortschritte macht, möchten Kunden ihre Produkte schnell und einfach bestellen können. Aus diesem Grund wurde die neue Bestell-App "ORDER CORNER" entwickelt, um den Bedürfnissen der Kunden gerecht zu werden. Die Merkmale, die "ORDER CORNER" hervorheben möchte, sind ein rückmeldungbasiertes Abonnement-System, ein sicheres Zahlungssystem und eine benutzerfreundliche Oberfläche

# Ausgangssituation und Ziele

Zuerst erleichtert die einfache und verständliche Oberfläche den Kunden, schnell Produkte auszuwählen und Bestellungen aufzugeben. Die App ermöglicht es auch Kunden, ihren Bestellverlauf, letzte gesuchte Produkte und frühere Einkäufe leicht nachzuverfolgen.

Das sichere Zahlungssystem ist ein weiteres wichtiges Merkmal von "ORDER CORNER". Kunden können sicher bezahlen, ohne sich Sorgen machen zu müssen. ---vor dem Einkauf wird dem Verbraucher ihr Passwort gefragt–

Schließlich ist ein wichtigstes Merkmal das rückmeldungsbasierte Abonnement-System, das "ORDER CORNER" von anderen Bestell-Apps unterscheidet. Der Kunde Verlaufsdaten der Produkte, die er regelmäßig kauft, aufbewahrt werden. Und mit einer Reihe von Empfehlungsalgorithmen kann ihm ein Produktabonnement vorgeschlagen werden.

Auf diese Weise kann "ORDER CORNER" ständig Feedback sammeln und sich verbessern, um die Erwartungen der Kunden zu erfüllen.

Mit all diesen Funktionen bietet "ORDER CORNER" Kunden eine einfache Möglichkeit, Bestellungen aufzugeben, während sie ihnen ein sicheres und benutzerfreundliches Erlebnis bieten. Aus diesem Grund ist "ORDER CORNER" eine ideale Wahl für jeden, der nach einer neuen Bestell-App sucht.

## Problemstellung (Funktionalität)

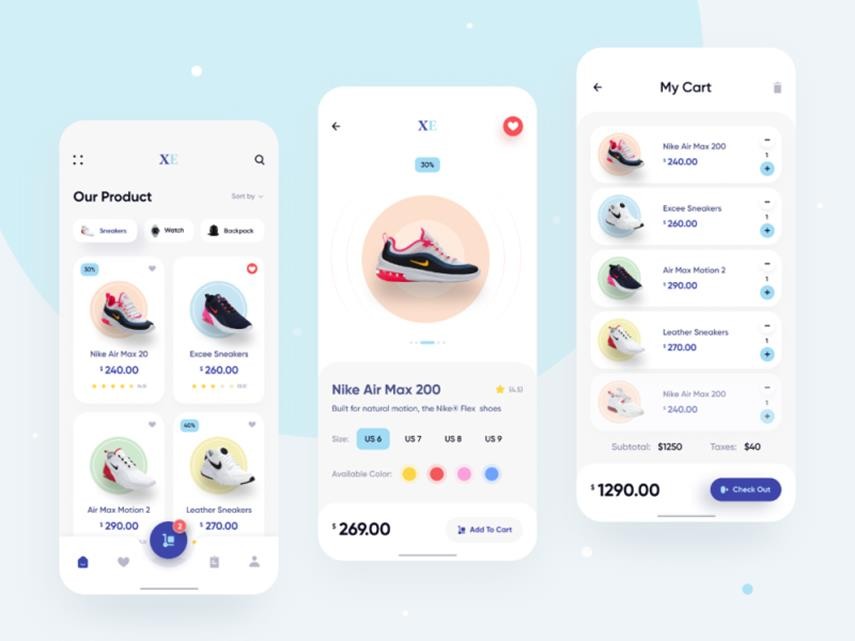
Es gibt einige Möglichkeiten, die von unserer Anwendung angeboten werden. Diese sind:

Bei der Konsumenten Seite

* Das letzte gesuchte Produkt wird auf der Startseite (Homepage) mit höchster Priorität in einem eigenen Teil als “Letzte gesucht” gezeigt.
* Diese App bietet die Möglichkeit Verbrauchern an, dass die Produkte, die von einem Verbraucher routiniert eingekauft werden, im Warenkorb des Verbrauchers autonom hinzugefügt werden, dadurch wird dem Verbraucher kein Zeitverlust erspart.
* Außerdem verhindert dieses System, dass Käufer Produkte vergessen, da Produkte automatisch an den Kunden innerhalb eines vorher festgelegten Zeitraums gesendet werden. Wenn der Verbraucher dieses Angebot nicht mehr verwenden möchte, dann wird das Abonnement abgesagt.

Bei der Hersteller Seite:

* Die Anzahl der Produkte nach einem Einkauf, sowohl in der Anwendung, als auch im Lagerhaus des Herstellers, aktualisiert werden. Durch dieses System wird der Lagerbestand von Produkten autonom verfolgt werden. Das kann ein Vorteil für Hersteller sein, dass sie im Lagerhaus Mitarbeiterkosten reduzieren können.
* Darüber hinaus steigt die Kundenbindung, da Kunden regelmäßig Produkte kaufen und somit ihre Loyalität zu den Produkten der Verkäufer erhöhen.
* Das Abonnement-Produkt-System ist in den letzten Jahren immer beliebter geworden. Dieses System bietet sowohl für Käufer als auch für Verkäufer viele Vorteile. Käufer können für einen bestimmten Zeitraum und gegen eine bestimmte Gebühr auf das Produkt zugreifen und dadurch von der regelmäßigen Bestellung des Produkts befreit werden. Gleichzeitig können Verkäufer ein regelmäßiges Einkommen erzielen und von der Leichtigkeit profitieren, Produkte regelmäßig an Kunden zu liefern.



## Rahmenbedingung (Einschränkungen)

* Die wichtigsten Einschränkungen befinden sich in der Wahl der Software Tools.

●

* Als Software-Entwicklungstool soll entweder Eclipse oder IntelliJ verwendet werden.

●

* die Backend Applikationen sollen mit Java Framework und Spring Boot Framework,

●

* die Frontend Applikationen mit React.js realisiert werden,

●

* die persistenten Daten sollen in einer MySQL-Datenbank gespeichert werden.

## Ziele (Lösung)

Es soll ein Prototyp entwickelt werden, der die automatisierte E-Commerce Anwendung simuliert und den Ablauf visuell demonstriert.

Darüber hinaus wird der Prototyp mit unterschiedlichen Test Units(z.B. JUnit,Mockito) geprüft.

Es soll auch möglich sein, mit allen Komponenten über eine Schnittstelle(GUI) die autonome Verwaltung der Daten zu erleichtern, zwischen Datenbank und Webservices mittels REST API zu kommunizieren.

Die externen Schnittstellen sollen mit Hilfe von RESTful Services(API) definiert werden. Die Persistenz soll mit einem Datenbanksystem erfolgen.

# Gesamtarchitektur

Die Gesamtarchitektur wird unter folgenden Aspekten betrachtet:

### Komponente <x>

**BAU (Bestellung Automation Unit) Komponent**

BAU ist die Steuereinheit und Information-Gateway der Automation E-Commerce Apps.

Die hervorgehobenen Komponenten und Komponenten Schnittstellen sollen hier beschrieben und verwaltet werden.

BAU kommuniziert, überwacht und steuert alle Komponenten Automation E-Commerce Apps. BAU stellt die KI-Services(Kontrolle-Interface) über die KI-Schnittstelle zur Verfügung, mit deren Hilfe der Anwender Informationen über die anderen PAI-Einheiten abrufen kann. Die Produkte auf der Startseite des Apps. Darüber hinaus stellt er die Services zur Verfügung, damit der Anwender einige Systemparameter konfigurieren und sie überwachen kann.

Die BAU bietet dem Anwender alle Informationen über der Komponente

Die Systemparameter und ihre Konfiguration und Überwachung soll im Laufe der Entwicklung definiert werden.

### Komponente NGP

Dieser Komponente sorgt dafür, dass nach dem gewünschten Produkt mit Filtern schneller zu schaffen und dem Verbraucher auf der Oberfläche zu zeigen.

### Komponente LGP

Dieser Komponente sorgt dafür, dass das letzte gesuchte Produkt dem Verbraucher auf der Startseite wieder zu zeigen. Dadurch kann der Verbraucher viel Zeit sparen. Im Worst-Case- Szenario, wenn die App ausgefallen ist, kann der Verbraucher schnell das letzte gesuchte Produkt finden.

### Komponente SB

Diese Komponente fordert den Benutzer nach einem Passwort für den erfolgreichen Einkauf und gewährleistet Zahlungssicherheit.

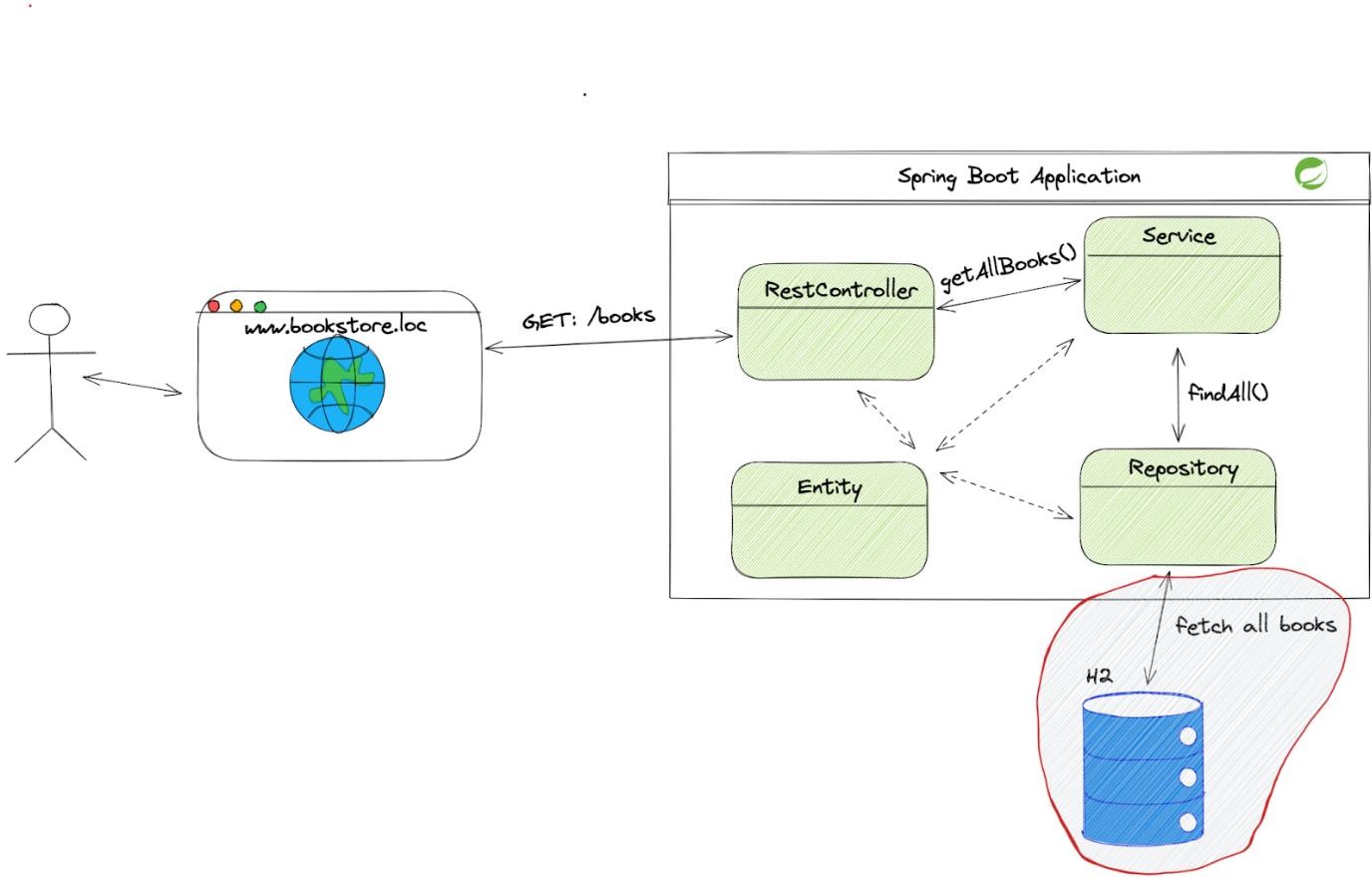
Die „Repository“ Sub-Komponente ist verantwortlich für die Kommunikation mit der externen Datenbank.

**Externe Schnittstellen**

**Die externen Schnittstellen sind:**

1. **Kontrollsschnittstelle(KI):** Schnittstelle zur Verwaltung von Systemkomponenten wie NGP auf der Hauptoberfläche.
2. **Produkt-Abonnement-Interface (PAI):** Schnittstelle zum Abonnieren von Verbrauchern für routinierte gekaufte Produkte.

Die Komponentenarchitektur soll nach dem Architekturmuster Model-View-Controller (MVC) entworfen werden, siehe das Spring Boot Framework für Detailinformation.



# Funktionale Anforderungen

In diesem Kapitel sind Anforderungen (inklusive User Stories und Use Cases) ein Gesamtsystem, aber auch an die einzelnen Systemkomponenten definiert.

### USE CASE - 1

**<< SUCHE NACH DEM GEWÜNSCHTEN PRODUKT”>> (NGP)**

### INITIAL ASSUMPTION / PRE-CONDITION

Ein erfolgreiches Einloggen ins System mit richtigem Passwort und Username

### NORMAL OPERATION

**Step 1:** Verbraucher sollten die Eigenschaften des gewünschten Produkts in die Suchleiste eingeben.

**Step 2:** Nach den Informationen von Produkten in der Datenbank wird kontrolliert, ob so ein Produkt mit bestimmten Eigenschaften im Lagerhaus von Herstellern vorhanden ist.

**Step 3:** Wenn so ein Produkt mit bestimmten Eigenschaften vorhanden ist(auf dem Markt), dann auf der nächsten Seite wird dem Verbraucher dieses Produkt mit seinem Preis gezeigt.

**Step 4:** Wenn der Verbraucher aufs Produkt anklickt, werden die Eigenschaften des Produkts detailliert gezeigt.

### WHAT CAN GO WRONG

( kein gültiger Eingabe vom Verbraucher)

kein Produkt gefunden im Lagerbestand (aufgebraucht)

### SYSTEM STATE ON COMPLETION / POST-CONDITION

Das Produkt erscheint im “letzte gesuchte” Teil, wenn es später abgeholt werden soll

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

### USE CASE - 2

**<< “LETZTE GESUCHTE PRODUKTE AUF DER STARTSETE”>> (LGP)**

### INITIAL ASSUMPTION / PRE-CONDITION

Ein erfolgreiches Einloggen ins System mit richtigem Passwort und Username

- Die Überprüfung eines Produkts des Verbrauchers

### NORMAL OPERATION

**Step 1:** Der Verbraucher überprüft ein Produkt nach seiner Suche zum Einkauf.

**Step 2:** Das letzte gesuchte Produkt von einem Verbraucher wird durch Web Services und Rest API in der Datenbank gespeichert.

**Step 3:** Auf der Startseite des Anwendung wird ein eigenes Teil als “letztes gesucht” erschaffen. Hier wird dem Anwender das letzte gesuchte Produkt mit dem Bild gezeigt. Dadurch spart der Anwender viel Zeit

**Step 4:** Wenn der Verbraucher aufs Produkt anklickt, werden die Eigenschaften des Produkts detailliert gezeigt.

### WHAT CAN GO WRONG

Wenn die Anwendung plötzlich ausgefallen ist, dann kann der Verbraucher einfach auf der Startseite das letzte gesuchte Produkt sehen.

Wenn der Anwender mehr als ein Produkt zur gleichen Zeit überprüft hat, dann können diese Produkte in der Datenbank nicht in der richtigen Ordnung gespeichert werden.

### SYSTEM STATE ON COMPLETION / POST-CONDITION

Das Produkt erscheint im “letzte gesuchte” Teil, wenn es später abgeholt werden soll

### USE CASE - 3

**<< “SICHERES BEZAHLUNGSSYSTEM”>> (SB)**

### INITIAL ASSUMPTION / PRE-CONDITION

Ein erfolgreiches Einloggen ins System mit richtigem Passwort und Username

- Ausreichendes Budget fürs Einkaufen

### NORMAL OPERATION

**Step 1:** Der Verbraucher überprüft ein Produkt nach seiner Suche zum Einkauf.

**Step 2:** Wenn das Produkt dem Verbraucher gefallen und das Budget vom Verbraucher ausreichend ist, kann er das Produkt in den Warenkorb zum Einkaufen hinzufügen.

**Step 3:** Der Verbraucher klickt am Knopf “Kauf ein” an

**Step 4:** Dem Verbraucher wird sein eigenes Passwort fürs Einkaufen gefragt.

**Step 5:** Wenn das eingegebene Passwort dem richtigen Passwort entspricht, wird das Einkaufen des Produkts mit Erfolg abgeschlossen.

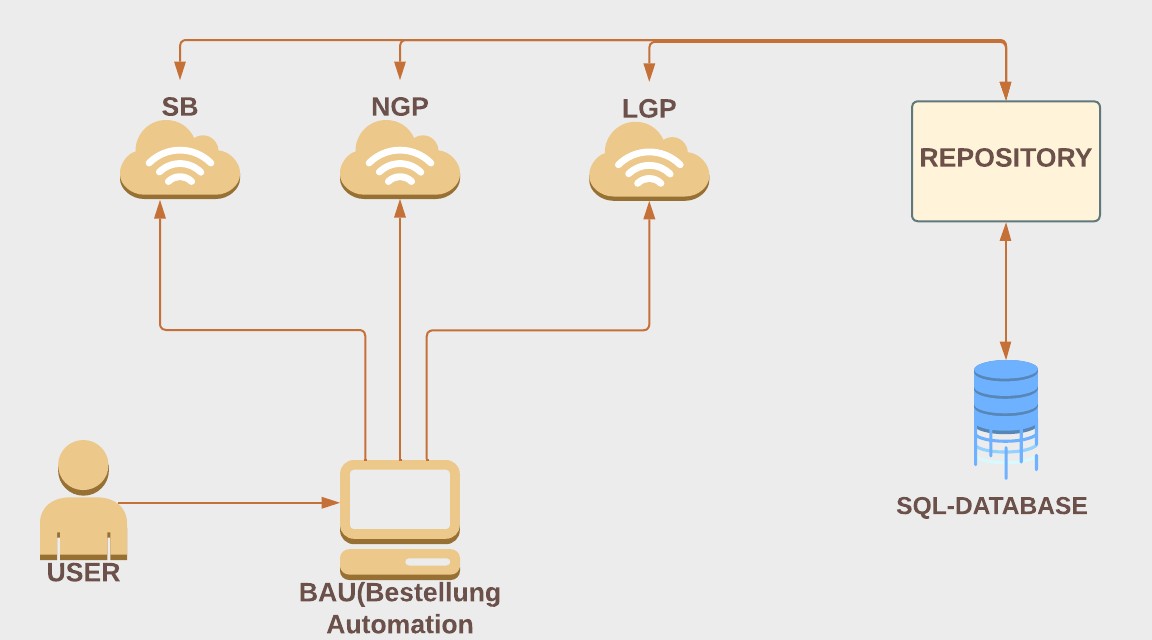
**Step 6**: Auf der Seite vom Hersteller wird die Anzahl des gekauften Produkts nach dem Einkauf aktualisiert und die Nachverfolgung des Lagerbestands erleichtert.

### WHAT CAN GO WRONG

Wenn das eingegebene Passwort dem richtigen Passwort nicht entspricht, wird der Einkauf gleich abgebrochen.

### SYSTEM STATE ON COMPLETION / POST-CONDITION

Mit dem REST API Post-, Update- und Delete-Service sollte die Anzahl des gekauften Produkts unbedingt aktualisiert werden. Sonst kann der Hersteller den Lagerbestand des Produkts nicht korrekt verwalten. Darüber hinaus kann ein Anwender ein ausverkauftes Produkt kaufen.



**Technischen und fachliche Anforderungen & Physikalische Systemarchitektur**

### BAU(Bestellung Automation Unit) KOMPONENT

**/BAU 1:/** Auf der Oberfläche sollen alle Produkte gezeigt werden. Darüber hinaus soll dem Anwender eine Möglichkeit gegeben werden, das gewünschte Produkt mit Hilfe unserer NGP- Komponente einfacher zu finden.

**/BAU 2:/** Durch die NGP-Komponente soll dem Verbraucher das gewünschte Produkt gefunden und gezeigt werden.

**/BAU 3:/** Wenn der Verbraucher das Produkt kaufen möchte, soll die Bezahlung nach dem SB- Komponent gemacht werden.

**/BAU 4:/** Nach dem Einkauf des Produkts soll der Lagerbestand durch SB-Komponent Post- Condition sowohl auf der Oberfläche als auch im Lagerhaus vom Hersteller aktualisiert werden.

**/BAU 5:/** Durch LGP-Komponent soll das letzte gesuchte Produkt in der Datenbank gespeichert und auf der Oberfläche gezeigt werden.

**/BAU 6:/** Mit dem PAI(Produkt Abonnement Interface) soll dem Verbraucher die Möglichkeit gegeben werden, für routinierte gekaufte Produkte abonniert zu sein.

## Andere technischen und fachliche Anforderungen

**/SYS-1/** Das Smart Home System soll eine verteilte Architektur aufweisen. Die

Komponente BAU ist die zentrale Komponente, die die restlichen Komponenten wie die NGP, LGP Komponente(Master-Slave Beziehung) steuert, siehe Abbildung 6 Physikalische Systemarchitektur .

**/SYS-2**/ Die „Slave“ Komponenten(z.B. NGP) sollen nur mit BAU und DBMS

kommunizieren. Zwischen BAU und DBMS sollen RestControllers die Produkte an die Oberfläche durch WebServices vermitteln.

**/SYS-3**/ Der Datenaustausch zwischen BAU und den „Slave“-Komponenten soll ausschließlich über Rest Controllers, Repositories und Services erfolgen.

**/SYS-4/** Jede Komponente soll für die persistenten Daten die DBMS-Komponente durch Repositories nutzen.

### Datenmodell

**/DAT-1/** Das Datenmodell muss die Parameter der API-Schnittstellen berücksichtigen.

/**DAT-2/** Das Datenmodell sollte mit dem UML-Klassendiagramm modelliert werden.

# Nichtfunktionale Anforderungen

**Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)**

**/SYS-1/** Deployment-Architektur: Alle Systemkomponente (BAU, NGP,LGP …) sollen in einem eigenen OS-Prozess lauffähig sein, somit ist das System eine verteilte Architektur aufweist.

**/SYS-2**/ Die Kommunikation der Systemkomponenten untereinander unterliegt der Beschränkungen der REST-Architektur und die Interaktionen den RESTful Webservices.

**Nicht-funktionale Anforderungen an die Entwicklungsumgebung**

Die nicht-funktionalen Anforderungen an die Entwicklungsumgebung sollen hier aufgelistet sein.

**/DEV-1/** Die Entwicklungsumgebung ist frei wählbar!

### Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)

Die nicht-funktionalen Anforderungen an die Entwicklungswerkzeuge sollen hier aufgelistet sein.

**/TOL-1**/ Die Backend Applikationen (NGP, LGP, SB) sollen mit Java Spring Boot Framework implementiert

werden.

**/TOL-2/** Die Frontend Applikation soll mit React Rich Client Technologie realisiert werden.

**/TOL-3/** Die persistenten Daten sollten in einer MySQL-Datenbank gespeichert werden.

### Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

Die nicht-funktionalen Anforderungen an die Teststrategie sollen hier beschrieben sein.

**/TEST-1/** Jeder System-Service soll Black-Box getestet werden

**/TEST-2**/ Alle Use Cases, User Stories und Anforderungen sollen mit Hilfe JUnit und Mockito Test Units getestet und berichtet werden.

# Referenzen

\*inf202-smarthome-pkw ladestation-pflichtenheft-2023.v0.1.doc

\*INF202-VO#5#6-VO-System Requirements WS-Pflichtenheft.v1