*TAU INF202 Software Engineering*

*Individuelles Projekt*

***Pflichtenheft***



Projektdokumentation

Version: 2023.v1.0

Status: fertig

Stockwell

Verantwortliche/r:

Çağlar Kabaca, e200503053@stud.tau.edu.tr

Burak Arslan, e200503051@stud.tau.edu.tr

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

Template: Projektdokumentation.doc / Version vom 29.09.2020

**Dokumentenverwaltung**

**Dokument-Historie**

| **Version** | **Status \*)** | **Datum** | **Verantwortlicher** | **Änderungsgrund** |
| --- | --- | --- | --- | --- |
| v1.0 | freigegeben | 25.02.2023 | Ö. Karacan | Vorlage wurde für die Studentenprojekte freigegeben |
| v1.0 | Entwurf | 01.04.2023 | atom-Team | Einführung in die Umsetzung des Projekts |
| v1.0 | fertig | 17.04.2023 | atom-Team | UML-Diagramm und Meilensteine |

*\*) Sofern im Projekt nicht anders vereinbart, sind folgende Statusbezeichnungen zu verwenden   
(in obiger Tabelle und am Deckblatt):*

***Dokument-Status: Entwurf / in Review / freigegeben (abgegeben)***

**Dokument wurde mit folgenden Tools erstellt:**

Microsoft Office Word

Google Docs

Lucidchart

**Inhaltsverzeichnis**

[**1.**](#_1ksv4uv) **Einleitung 4**

[**2.**](#_2et92p0) **Ausgangssituation und Ziele 5**

[**3.**](#_3dy6vkm) **Gesamtarchitektur 6**

[**4.**](#_1t3h5sf) **Funktionale Anforderungen 7**

[**5.**](#_17dp8vu) **Nichtfunktionale Anforderungen 8**

[**6.**](#_26in1rg) **Abnahmekriterien 9**

[**7.**](#_lnxbz9) **Projekt Meilensteine 10**

[**8.**](#_35nkun2) **Referenzen 11**

# **Einleitung**

Ziel dieses Dokuments ist es, die Struktur der ERP-Software Stockwell umfassend und konsistent zu beschreiben.

Das Kapitel 2 “Ausgangssituation und Ziele” erläutert ausführlich die Ausgangssituation und die Gründe für die Entscheidung zugunsten von Stockwell.

Im Kapitel 3 "Gesamtarchitektur" werden die physische und konzeptionelle Architektur des Systems sowie die wichtigsten Komponenten, Benutzer und notwendigen Kommunikationsschnittstellen detailliert beschrieben. Zusätzlich werden hier auch weitere Anforderungen an die Architektur und Komponenten aufgeführt.

Das Kapitel 4 "Funktionale Anforderungen" beschreibt die funktionalen Anforderungen durch Anwendungsfälle (Use Cases) sowie technische und fachliche Anforderungen. Die Datenmodelle definieren alle relevanten Daten für den Betrieb.

Das Kapitel 5 "Nichtfunktionale Anforderungen" ergänzt die funktionalen Anforderungen um solche Anforderungen, die nicht funktionaler Natur sind.

Im Kapitel 6 "Abnahmekriterien" werden die Abgabeartefakte definiert, die ohne Zustimmung der Stakeholder nicht verändert werden dürfen.

Das Kapitel 7 "Projektmeilensteine" listet die wichtigsten Termine auf, welche den Fortschritt und die Teilergebnisse des Projekts definieren.

Im Kapitel 8 "Referenzen" sind die wichtigsten Quellen aufgeführt.

# **Ausgangssituation und Ziele**

**Einleitung**

In diesem Kapitel geht es um einige Details des Projekts und darum, warum dieses Projektthema ausgewählt wird.

**Problemstellung (Funktionalität)**

Eines der wichtigsten Probleme im Arbeitsleben ist die Planung der Ressourcen. Stockwell hilft Unternehmen genau in diesem Punkt.

**Stakeholder (Anwender):**

Von kleinen Geschäften bis zu großen Unternehmen kann Stockwell jedem Unternehmen dienen. Eigentlich ist Stockwell für jedes Unternehmen, das seinen Arbeitsfluss optimieren und Personalkosten sparen möchte. Außerdem kann fast jedes Personal durch Stockwells benutzerfreundlicher Oberfläche das Programm benutzen.

**Systemumfeld (Einsatzumgebung)**

Das System kann auf Servern oder auf lokalen Computern von Unternehmen funktionieren. Als Betriebssystem kann man Windows, Linux oder macOS verwenden, weil Java eine plattformunabhängige Programmiersprache ist.

**Rahmenbedingung (Einschränkungen)**

Für die Projektdurchführung gibt es die Einschränkung, dass das System auf zu alter Hardware nicht so effizient funktionieren kann. Um diese Einschränkung zu beseitigen, wurde die Struktur von Stockwell so gebildet, dass die Arbeitsbelastung des Systems so gering wie möglich ist.

**Ziele (Lösung)**

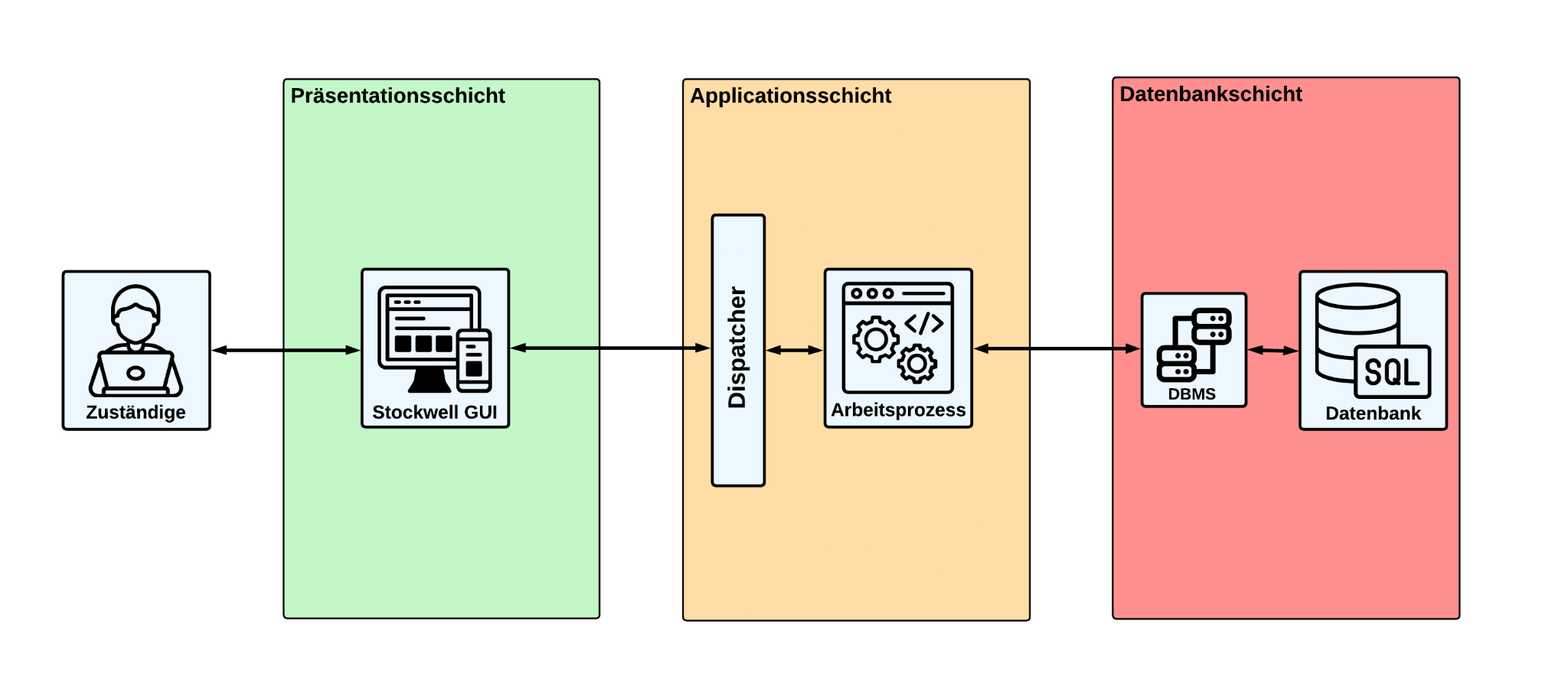
Durch das System ist es gezielt, dass Unternehmen ihre Ressourcen besser verwalten können, damit sie die Digitalisierung-Ära nicht verpassen.

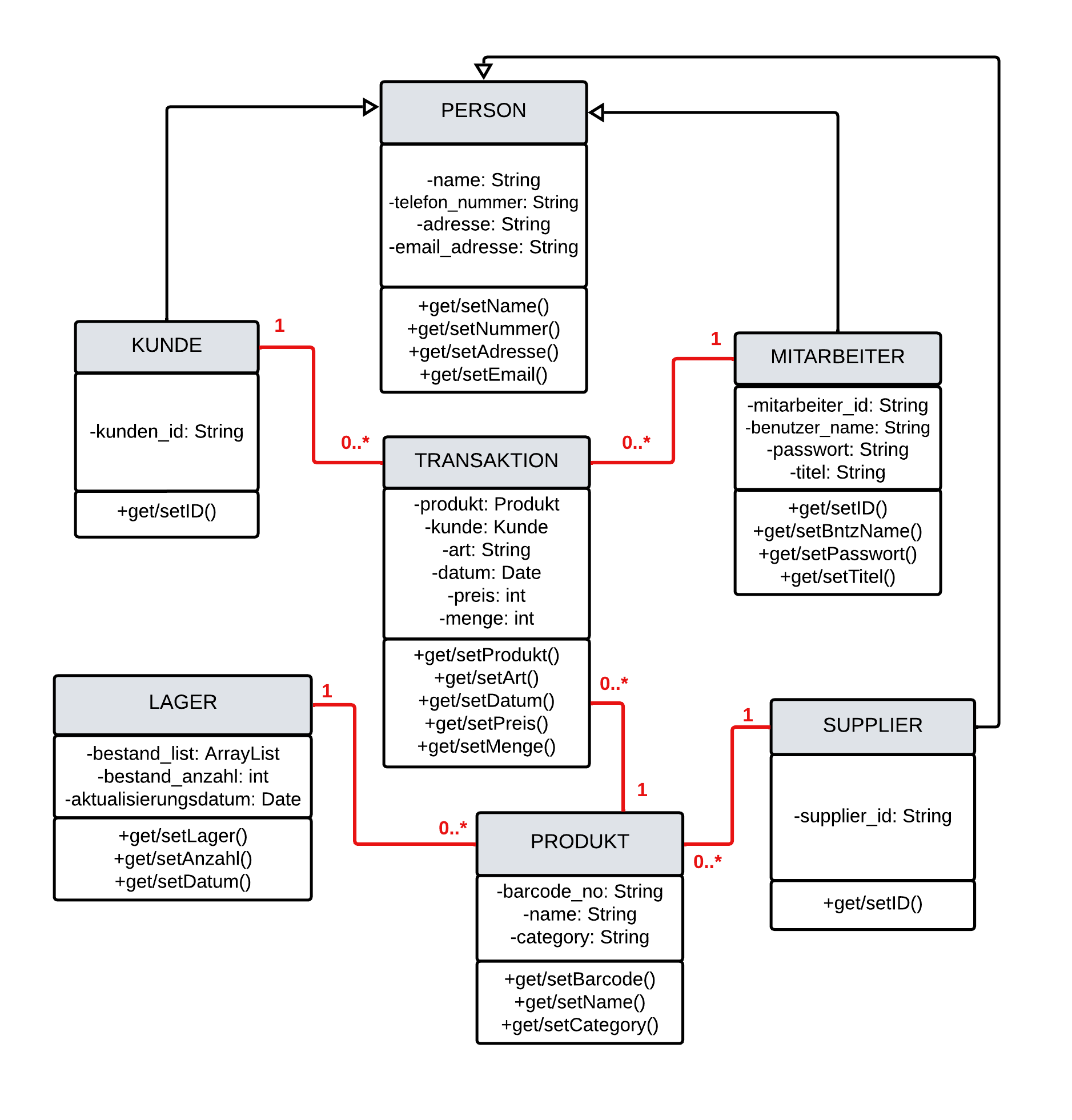
# 

# **Gesamtarchitektur**

**Einleitung**

Stockwell hat genau drei Hauptkomponente in seiner Architektur. Diese Komponenten, die auch als Schicht genannt werden, heißen Präsentationsschicht, Applikationsschicht und Datenbankschicht. Alle drei Komponenten führen eigentlich verschiedene Operationen aus, aber sie benötigen sich gegenseitig für eine präzise Umsetzung.

***Gesamtarchitektur***



***Komponente <x>***

**Präsentationsschicht**

Diese Komponente führt die Beziehung zwischen Benutzer und Applikation aus. Alle Operationen über Benutzeroberfläche setzen in dieser Schicht um.

**Applikationsschicht**

Applikationsschicht kontrolliert Datenaustausch zwischen Präsentationsschicht und Datenbankschicht. Was der Benutzer zeigt oder welche Daten in die Datenbank eingegeben werden, die Applikationsschicht ist dafür verantwortlich, wie diese Operationen durchgeführt werden.

**Datenbankschicht**

In dieser Komponente gibt es Datenbank und Datenbankmanagementsystem (DBMS) darin. Alle wichtigen Daten werden von Benutzer in dieser Schicht gespeichert.

***Externe Schnittstellen***

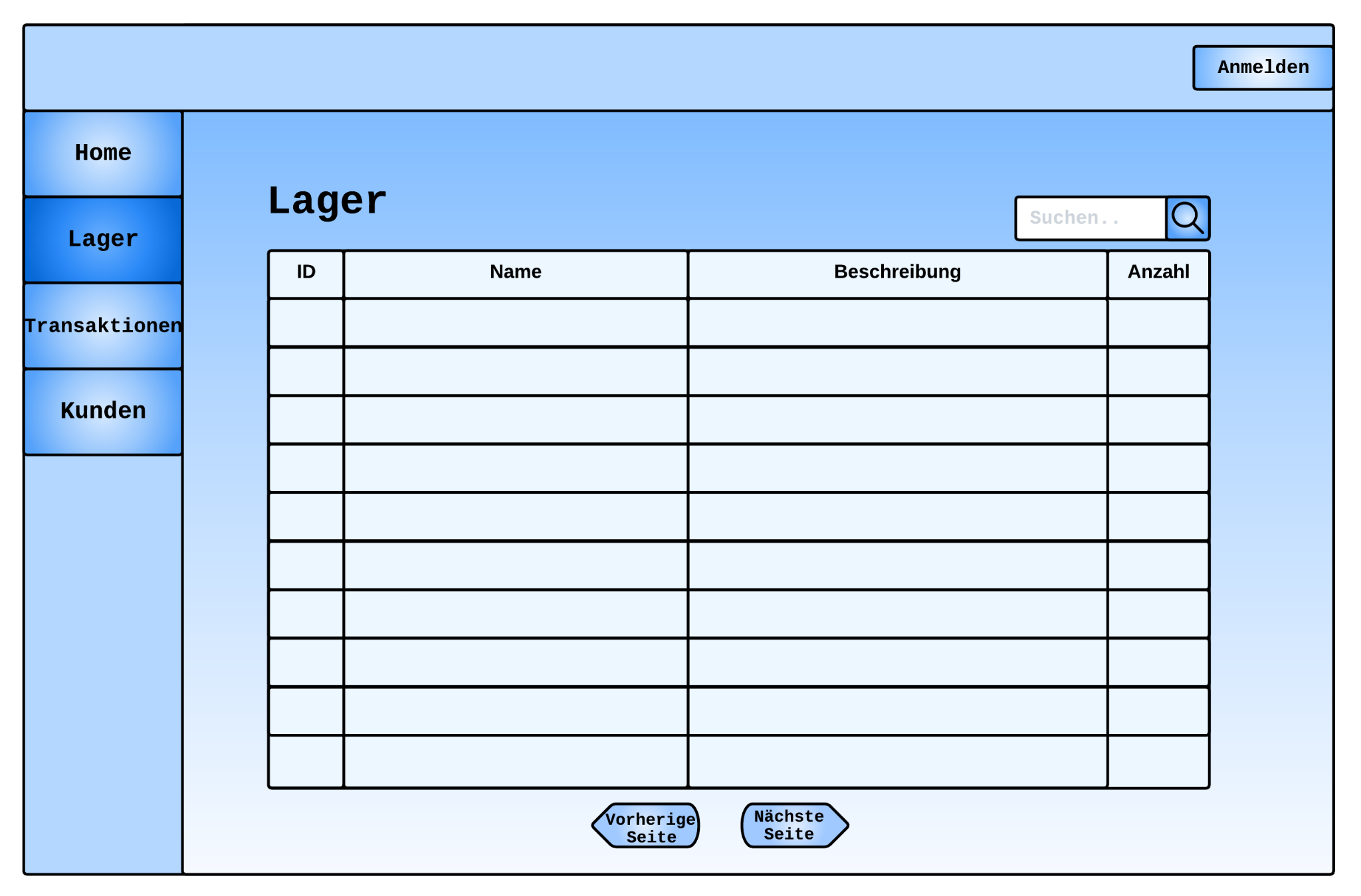
# **Funktionale Anforderungen**

**Einleitung**

Dieses Kapitel enthält wichtige Informationen zur Funktionalität des Systems. Beziehungsweise wird das System von verschiedenen Aspekten betrachtet. Welche Use Cases die Benutzeroberfläche bietet, oder welche Informationen durch API zu erreichen sind.

**UI Use Cases**

Die grafische Benutzeroberfläche wird ähnlich wie im unten angegebenen Bild aussehen. Auf der linken Seite befinden sich einige Buttons, durch die man in verschiedene Teile des Systems hingehen kann. Der Home-Button leitet den Benutzer zum Beispiel zur Home-Seite hin. Durch Klicken auf Lager kann man das Lager tabellarisiert ansehen und die Werte darin verändern. Wenn der Benutzer auf den Button “Transaktionen” klickt, kann er ansehen, wie viele von welchen Produkten gekauft/verkauft wurden.



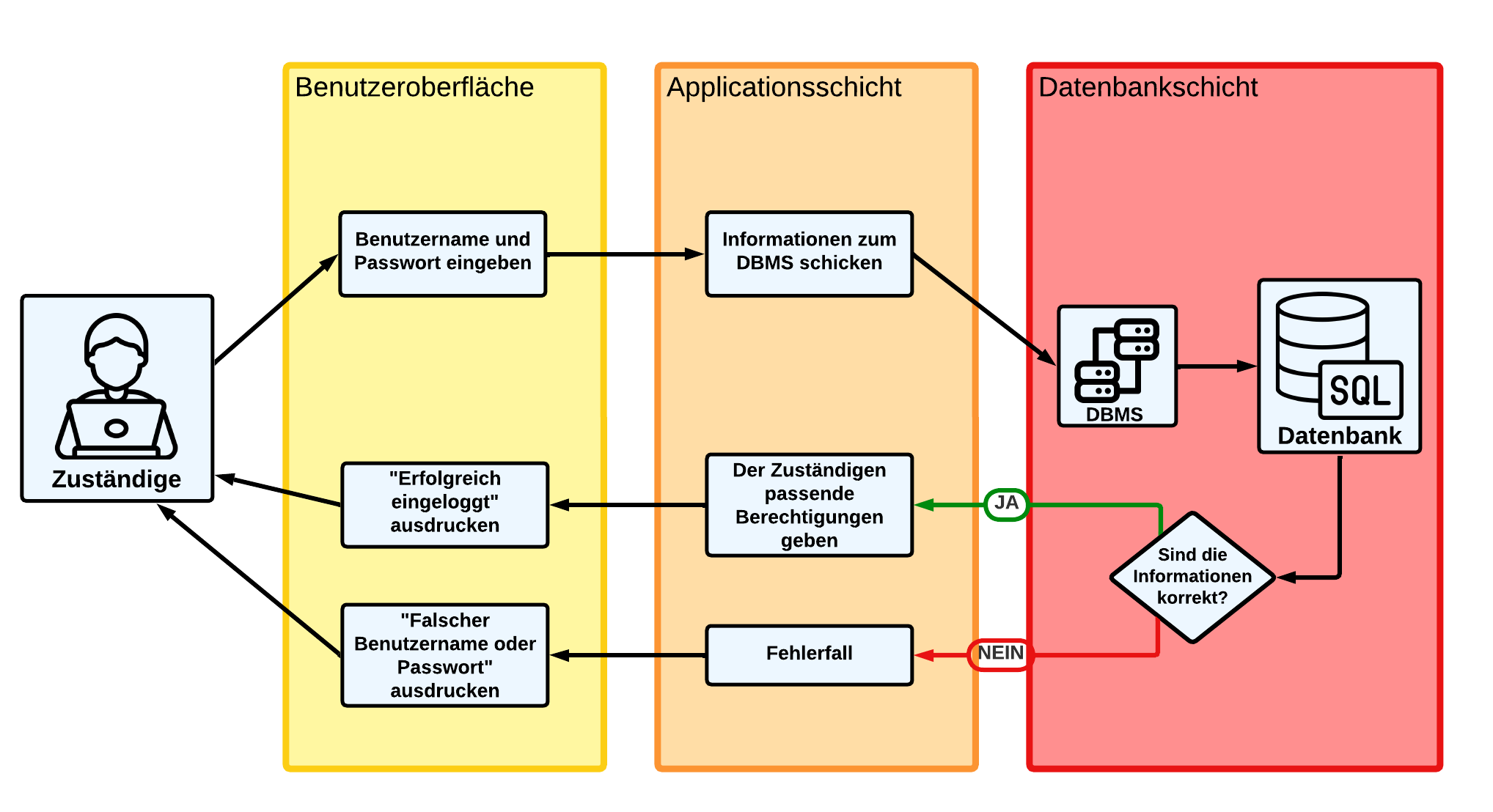
Ein visualisiertes Use-Case-Beispiel ist unten angegeben. In diesem Beispiel klickt die Zuständige

auf einen Button, auf dem “Das Lager anzeigen” steht. Danach wird eine Anfrage erstellt und die Datenbank wird nach der Liste der Bestände im Lager angefragt. Nachdem die Datenbank vorhandene Informationen zur Bestandsliste ausgeliefert wurde, wird die Liste visualisiert und der Zuständigen angezeigt.

# 

**API Use Cases**

Unten wurde ein Beispiel für das Use-Case “Einloggen” angegeben. Das System funktioniert ähnlich in anderen Use-Cases wie dieses Beispiel.



**Technische und fachliche Anforderungen**

* **Lagerverwaltung**

Nach dem Einloggen ins System kann der Benutzer das Lager tabellarisch ansehen. In dieser Tabelle befinden sich alle Produkte, die im Lager vorhanden sind. Wenn der Anwender die benötigten Berechtigungen hat, kann er auch die Daten im Lager bzw. in der Datenbank modifizieren.

* **Einkäufe und Verkäufe**

Der Benutzer kann Ihre Einkäufe und Verkäufe unter dem Titel “Transaktionen” ansehen und diese Aufzeichnungen nach Datum, Einstandspreisen und Verkaufspreisen sortieren. Wenn der Benutzer eine passende Zuständige ist, kann er auch neue Einkäufe und Verkäufe aufzeichnen. Alle diese Daten befinden sich natürlich in der Datenbank.

* **Finanzmanagement**

Der Anwender kann Profit- und Verluststatus der Firma überprüfen und sie mit vergangenen Tagen, Wochen, Monaten, Jahren vergleichen, wenn er verantwortlich für Finanz ist. Diese Daten werden grafisch dargestellt, um die Durchschaubarkeit zu verbessern.

**Datenmodell**

Ein DBMS ist für die Persistenz der Daten notwendig. Deswegen wird ein externes DBMS zusammen mit dem System verwendet. Zudem wird ein UML-Diagramm erstellt, damit die Kommunikation zwischen den Klassen klarer und besser verstanden werden kann. Außerdem kann auch ein Entity-Relationship Diagramm erstellt werden, um die Klassifizierung von Daten zu vereinfachen.

# **Nichtfunktionale Anforderungen**

**Einleitung**

In diesem Kapitel geht es um Anforderungen, die nicht-fachlicher Natur sind. Diese werden auch als nicht-funktionale Anforderungen genannt.

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

Die Kommunikation zwischen den Hauptkomponenten muss gut gestaltet werden, damit die Schichten zusammen effizient laufen können. JVM (Java Virtual Machine) muss auf dem System bereitgestellt werden.

**Nicht-funktionale Anforderungen an die Entwicklungsumgebung**

Die Entwicklungsumgebung kann frei gewählt werden.

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

Die Backend-Applikationen sollen mit Java und Spring Framework implementiert werden. Als IDE sollte Intellij-IDE verwendet werden, weil es sehr kompatibel mit Java ist.

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

Alle Abhängigkeiten (Dependencies) müssen gut geklärt werden, um die Inkompatibilität so viel wie möglich zu verringern.

# **Abnahmekriterien**

*Die Abnahmekriterien sind durch den Stakeholder definiert und sie dürfen nur mit Zustimmung des Stakeholders neu definiert, geändert oder erweitert werden.*

Das Projekt wird mit den folgenden Artefakten abgegeben:

* Dokumentation:
  + Pflichtenheft: *stockwell.docx*
* Software
  + Link zu GitHub Projekt: *https://github.com/Gruppe1-Fulya/stockwell*
* Evidenz:
  + System/Software-Demo via Videoclip: *https://youtu.be/\_CDxmhszCd0*

Anm.: Die Abgabetermine der Projektartefakts werden durch den Stakeholder festgelegt!

# **Projekt Meilensteine**

Meilenstein #1: Es wurde festgestellt, welche Technologien im Projekt verwendet werden und daneben wurden Use Cases definiert.

Meilenstein #2: Die Architektur des Systems entschied sich und eine Demoversion des Programms wurde bereitgestellt.

# **Referenzen**

1. <https://docs.oracle.com/en/java/javase/16/docs/api/index.html>
2. <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>