

# **Inventarisierungsmodule der Bestellsoftware in Kooperation mit KEIM für die Hochschule Esslingen**

## **Dokumentation**

im Studiengang  
Softwaretechnik und Medieninformatik  
der Fakultät Informationstechnik

vorgelegt von  
**Dominik Ehmann, Luka Weipert, Sandro Lipinski,  
Akim Kausch, Valentin Mitrev**

am 20. Mai 2025  
an der Hochschule Esslingen

---

**Kunde:** Dipl.-Phys. Emanuel Reichsöllner  
**Betreuer:** M. Sc. Andreas Heinrich  
**Prüfer:** Prof. Dr. rer. nat. Jörg Nitzsche  
**Zeitraum:** 10.03.2025 - 27.06.2025

## **Ehrenwörtliche Erklärung**

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Esslingen, den 20. Mai 2025

\_\_\_\_\_  
Unterschrift

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>I</b>
<b>Tabellenverzeichnis</b>	<b>II</b>
<b>Quellcodeverzeichnis</b>	<b>III</b>
<b>Abkürzungsverzeichnis</b>	<b>IV</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Problemstellung . . . . .	1
1.2 Problemlösungsansatz . . . . .	1
1.3 Zielgruppe . . . . .	1
1.3.1 Angestellte . . . . .	1
1.3.2 Administrator . . . . .	1
1.4 Personas . . . . .	1
1.4.1 Angestellte . . . . .	1
1.4.2 Administrator . . . . .	3
1.5 User Stories . . . . .	4
1.5.1 Angestellter . . . . .	4
1.5.2 Administrator . . . . .	5
<b>2 Gefordeter Funktionsumfang</b>	<b>6</b>
2.1 Funktionale Anforderungen . . . . .	6
2.1.1 Benutzerverwaltung und Authentifizierung . . . . .	6
2.1.2 Geführter Inventarisierungsprozess . . . . .	6
2.1.3 Anbindung an Bestellprozess . . . . .	6
2.1.4 Such- und Filterfunktionen . . . . .	6
2.1.5 Datenimport und –export . . . . .	7
2.1.6 Statistiken . . . . .	7
2.1.7 Nice to have . . . . .	7
2.2 Nicht-Funktionale Anforderungen . . . . .	7
2.2.1 Wartbarkeit . . . . .	7
2.2.2 Hohe Benutzerfreundlichkeit . . . . .	7
2.2.3 Integration in bestehende IT-Landschaft . . . . .	8
2.2.4 Code-Qualität . . . . .	8
<b>3 UI Entwürfe</b>	<b>9</b>
3.1 Inventarliste . . . . .	9
3.1.1 Filtermenü . . . . .	10
3.2 Detailansicht eines Gegenstandes . . . . .	10

3.3	Inventarisierung . . . . .	11
3.3.1	Bestellübersicht . . . . .	12
3.3.2	Artikelübersicht . . . . .	12
3.3.3	Inventarisierungswizard . . . . .	13
3.3.4	Direkte Inventarisierung . . . . .	14
3.4	Statistiken . . . . .	15
<b>4</b>	<b>Architektur</b>	<b>18</b>
4.1	Technologische & konzeptionelle Architektur . . . . .	18
4.1.1	Strukturschicht . . . . .	19
4.2	Deployment Prozess . . . . .	20
4.2.1	Einleitung . . . . .	20
4.2.2	Github Workflows . . . . .	20
4.2.3	Aufbau eines Workflows . . . . .	20
4.2.4	Docker . . . . .	20
4.2.5	Sonarqube . . . . .	21
4.3	Frontend . . . . .	22
4.3.1	Technologien . . . . .	23
4.4	Backend . . . . .	24
4.5	Datenbank . . . . .	25
4.6	Keycloak . . . . .	27
4.6.1	Keycloak . . . . .	27
<b>5</b>	<b>Projektmanagement</b>	<b>30</b>
5.1	Entwicklungsprozess und Methodik . . . . .	30
5.2	Versionsverwaltung . . . . .	30
5.3	Meetings . . . . .	30
5.4	Definition of Done . . . . .	30
5.5	Kommunikation und Dokumente . . . . .	31
5.6	Lizenz . . . . .	31
<b>6</b>	<b>Schnittstellen</b>	<b>32</b>
<b>7</b>	<b>Aufwandsschätzung</b>	<b>34</b>
7.1	Geschätzter Aufwand . . . . .	34

# Abbildungsverzeichnis

1.1	Persona Angestellter Michael . . . . .	2
1.2	Persona Administrator Stefan . . . . .	3
3.1	Prototyp Inventarliste . . . . .	9
3.2	Prototyp Filtermenü . . . . .	10
3.3	Prototyp Detailansicht . . . . .	11
3.4	Prototyp Neue Inventarisierung . . . . .	12
3.5	Prototyp Bestellübersicht . . . . .	13
3.6	Prototyp Artikelübersicht . . . . .	14
3.7	Prototyp Inventarisierungswizard . . . . .	15
3.8	Prototyp Direkte Inventarisierung . . . . .	16
3.9	Prototyp Erweiterungen . . . . .	16
3.10	Prototyp Statistiken . . . . .	17
4.1	Three-Tier Architecture . . . . .	18
4.2	Strukturschicht . . . . .	19
4.3	CI/CD Pipeline . . . . .	22
4.4	Datenbankmodell . . . . .	26
6.1	Dokumentationsbeispiel zu einem POST-Request an /inventories . . . . .	33

## **Tabellenverzeichnis**

7.1 Ursprüngliche Aufwandsschätzung . . . . .	41
-----------------------------------------------	----

## **Quellcodeverzeichnis**

## **Abkürzungsverzeichnis**

# 1 Einleitung

## 1.1 Problemstellung

Die Hochschule Esslingen besitzt eine Excel-Liste, die sämtliche Geräte und Gegenstände erfasst, die sich im Besitz der Hochschule befinden. Diese Liste ist auf einem internen Netzwerklaufwerk abgelegt, weist jedoch aufgrund ihrer begrenzten Funktionalitäten Integritätsprobleme auf. Daher besteht die Notwendigkeit, eine zuverlässige und benutzerfreundliche Lösung zu entwickeln, die eine effiziente Erfassung, Verwaltung und Aktualisierung der Geräte und Artikel gemäß den Inventarisierungsrichtlinien ermöglicht.

## 1.2 Problemlösungsansatz

Der Auftraggeber wünscht sich eine webbasierte Lösung zur Verwaltung von Inventarisierungslisten. Die entscheidenden Funktionen umfassen das Inventarisieren und De-Inventarisieren von Gegenständen, den Import von Excel-Tabellen in das Inventarisierungsmodul, den Export aus dem Inventarisierungsmodul in Excel-Tabellen sowie die Integration mit der derzeit in Entwicklung befindlichen Bestellsoftware. Die Software soll insbesondere die Pflege und das Hinzufügen neuer Gegenstände erleichtern und aussagekräftige Statistiken bereitstellen, um einen umfassenden Überblick über die Inventarliste zu ermöglichen. Detaillierte Anforderungen werden im folgenden Kapitel erläutert.

## 1.3 Zielgruppe

### 1.3.1 Angestellte

Wir haben die Zielgruppen unserer App umfassend analysiert, um deren spezifische Bedürfnisse und Anforderungen bestmöglich zu berücksichtigen. Dabei lag der Fokus auf Funktionalität und Benutzertauglichkeit, um eine optimale Nutzung für alle Anwender zu gewährleisten.

### 1.3.2 Administrator

Der Administrator ist für die Verwaltung der Benutzerzugriffsrechte, sowie die Überwachung und Wartung der App verantwortlich. Diese Zielgruppe ist in der Regel ab 30 Jahre alt, arbeitet in der IT-Branche und ist mit der Hochschule Esslingen verbunden.

## 1.4 Personas

### 1.4.1 Angestellte

Michael, wie auf Abbildung 1.1 zu sehen, ist ein verantwortungsbewusster, strukturierter 42-jähriger Angestellter, der in der Verwaltung einer Hochschule arbeitet. Mit seiner ruhigen und organisierten Art sorgt er dafür, dass Professoren und Assistenten stets mit den benötigten



Abbildung 1.1: Persona Angestellter Michael *Quelle: [<empty citation>]*

Materialien versorgt sind. Michael legt Wert auf Effizienz und klare Abläufe. Er ist es gewohnt, mit begrenzten Budgets zu arbeiten und muss Bestellungen strategisch planen, um Kosten zu optimieren.

### Ziele

- Benötigte Materialien einfach und effizient beschaffen.
- Den Überblick über Bestellungen und Budgets behalten.
- Prozesse optimieren, um Professoren und Assistenten bestmöglich zu unterstützen.

### Bedürfnisse

- Eine benutzerfreundliche Bestellplattform mit klaren Übersichten.
- Funktionen zur Budgetkontrolle und Nachverfolgung von Bestellungen.
- Transparente Preis- und Lieferinformationen.

### Probleme

- Hoher administrativer Aufwand bei der Materialbeschaffung.
- Begrenztes Budget, das effizient genutzt werden muss.
- Bedarf an einer einfachen, übersichtlichen Lösung zur Bestellung.

### Wie könnte unsere Bestell-App Michael helfen?

Die Bestell-App könnte Michael dabei unterstützen, Materialien schnell und unkompliziert zu bestellen. Mit Funktionen wie Bestellhistorien und Budgetübersichten behält er stets den Überblick. So kann er effizient arbeiten, ohne Zeit mit komplizierten Bestellprozessen zu verlieren – ideal für seine strukturierte und verantwortungsbewusste Arbeitsweise.



Abbildung 1.2: Persona Administrator Stefan *Quelle: [<empty citation>]*

## 1.4.2 Administrator

Stefan, wie auf Abbildung 1.2 zu sehen, ist ein 35-jähriger Administrator, der für die Verwaltung und Wartung der IT-Systeme an der Hochschule Esslingen verantwortlich ist. Mit seiner ruhigen und analytischen Art sorgt er dafür, dass die technischen Systeme reibungslos funktionieren und die Sicherheitsprotokolle eingehalten werden. Stefan hat ein tiefes Verständnis für Technologie und IT-Infrastruktur. Er ist es gewohnt, sowohl im Hintergrund als auch in direkter Zusammenarbeit mit verschiedenen Abteilungen zu arbeiten, um den reibungslosen Betrieb der Systeme zu gewährleisten.

### Ziele

- Die Sicherheit und Stabilität der App gewährleisten.
- Benutzerzugriffsrechte effizient verwalten.
- Regelmäßige Wartungsarbeiten und Systemupdates durchführen, um die App immer auf dem neuesten Stand zu halten.

### Bedürfnisse Bedürfnisse:

- Eine benutzeroberflächenfreundliche Verwaltungsoberfläche zur Verwaltung von Benutzerrechten.
- Detaillierte Berichte und Analysen zur Systemleistung und Sicherheit.
- Effiziente Tools zur Fehlerdiagnose und Problemlösung.

### Probleme Probleme:

- Hoher Druck, die Systeme rund um die Uhr verfügbar zu halten.
- Bedarf an schneller Fehlerbehebung, um Ausfallzeiten zu minimieren.
- Hohe Verantwortung für die Systemsicherheit und Performance.

### Wie könnte unsere App Stefan helfen?

Die App könnte Stefan helfen, Benutzerzugriffsrechte schnell und übersichtlich zu verwalten und gleichzeitig die Systemleistung in Echtzeit zu überwachen. Mit integrierten Sicherheitsfunktionen, automatischen Updates und einem benutzerfreundlichen Dashboard könnte Stefan effizient arbeiten und sicherstellen, dass die App jederzeit reibungslos funktioniert – ideal für seine technische Expertise und seine verantwortungsvolle Position.

### 1.5 User Stories

#### 1.5.1 Angestellter

- Als Angestellter möchte ich benötigte Materialien schnell und unkompliziert über eine klare Bestellplattform bestellen, um meine Arbeit effizient erledigen und Professoren sowie Assistenten zuverlässig unterstützen zu können.
- Als Angestellter möchte ich jederzeit mein verfügbares Budget und den Status meiner Bestellungen einsehen können, damit ich strategisch planen und das Budget optimal einsetzen kann.
- Als Angestellter möchte ich bestehende Excel-Listen importieren und aktuelle Inventardaten exportieren können, damit wir nahtlos mit bestehenden Systemen kompatibel bleiben.
- Als Angestellter möchte ich die Historie und Änderungen eines Inventargegenstandes einsehen können, um nachvollziehen zu können, wer wann welche Änderungen gemacht hat.
- Als Angestellter möchte ich bestehende Inventareinträge bearbeiten oder löschen können, um die Richtigkeit der Bestandsdaten sicherzustellen.
- Als Angestellter möchte ich die Bestellungen mit frei wählbaren Tags versehen können, um Bestellungen besser kategorisieren, schneller wiederfinden und die Übersicht behalten zu können.
- Als Angestellter möchte ich die Möglichkeit haben, Sachen zu deinvantarisieren, um Gegenstände, die nicht mehr ein Bestandteil des Inventars sind, löschen zu können.
- Als Angestellter möchte ich die Möglichkeit haben, Kostenstellen zu archivieren, um relevante und aktive Kostenstellen angezeigt zu bekommen. Gleichzeitig möchte ich die wichtigsten Kostenstellen einmalig manuell erstellen können, damit ich sie langfristig verwalten und effizient nutzen kann.

### 1.5.2 Administrator

- Als Administrator möchte ich Zugriffsrechte verwalten, um unberechtigte Zugriffe zu vermeiden.
- Als Administrator möchte ich regelmäßig Updates und Wartungsarbeiten durchführen können, um die Sicherheit und Stabilität der Anwendung zu gewährleisten.
- Als Administrator möchte ich Berichte und Analysen zur Systemleistung und Sicherheit abrufen können, damit ich schnell auf Probleme reagieren und die Stabilität der App sicherstellen kann.
- Als Administrator möchte ich Benutzerverwaltung un Authentifizierung der App über Keycloak integrieren und steuern, um Benutzerzugriffe sicher, zentralisiert und effizient verwalten zu können.

## 2 Gefordeter Funktionsumfang

### 2.1 Funktionale Anforderungen

#### 2.1.1 Benutzerverwaltung und Authentifizierung

Das System muss ein Zugriffsmanagement-Tool zur Authentifizierung der Benutzer bereitstellen, um sicherzustellen, dass ausschließlich berechtigte Nutzer Zugriff auf das Inventarisierungsmodul erhalten.

#### 2.1.2 Geführter Inventarisierungsprozess

Um die Benutzer bei der Inventarisierung zu unterstützen, soll das System sie strukturiert durch den Bestell- und Inventarisierungsprozess führen. Dazu gehört die geführte Erstellung neuer Gegenstände, bei der alle relevanten Daten erfasst werden. Zudem erfolgt eine automatische Überprüfung des Eintrags, wobei keine Abfrage für Nutzungsdauer, Verbrauchs- oder Gebrauchsgüter erforderlich ist. Falls der Wert eines Gegenstands unter 250 € liegt, muss eine Abfrage erfolgen, ob dieser tatsächlich inventarisiert werden soll.

Die De-Inventarisierung erfolgt manuell alle zehn Jahre, wobei Einträge als „Nicht mehr Bestandteil des Inventars“ markiert werden können, ohne dass sie aus dem System gelöscht werden.

Erweiterungen bestehender Gegenstände werden nicht als separate Einträge erfasst, sondern dem ursprünglichen Eintrag hinzugefügt und mit einer Notiz versehen, beispielsweise „RAM-Riegel hinzugefügt am dd.mm.yyyy“.

#### 2.1.3 Anbindung an Bestellprozess

Das Inventarisierungsmodul muss zudem eine Anbindung an die Bestellsoftware der Hochschule Esslingen bereitstellen. Nach einer Bestellung soll der Benutzer entscheiden können, ob der bestellte Gegenstand in das Inventarisierungsmodul aufgenommen wird.

#### 2.1.4 Such- und Filterfunktionen

Um eine effiziente Verwaltung der Inventargegenstände zu gewährleisten, muss das System umfassende Such- und Filterfunktionen bieten. Nutzer sollen Einträge anhand verschiedener Kriterien suchen und filtern können.

Darüber hinaus muss eine Historienverwaltung bereitgestellt werden, mit der nachvollzogen werden kann, welche Person welchen Gegenstand bestellt oder inventarisiert hat. Zusätzlich

## **2 GEFORDETER FUNKTIONSUMFANG**

---

sollen Inventargegenstände mit Tags wie „Laborbestand“ oder „Haushaltsausgaben“ versehen werden können, die ebenfalls durchsuchbar sind.

### **2.1.5 Datenimport und –export**

Zur Sicherstellung der Abwärtskompatibilität mit dem bestehenden System soll das Inventarisierungsmodul den Import bisheriger Excel-Listen ermöglichen und den Export von Daten in das bestehende Excel-System unterstützen.

### **2.1.6 Statistiken**

Das System muss visuelle Darstellungen von Inventarisierungsstatistiken in Form von Diagrammen und Grafiken ermöglichen. Dabei sollen beispielsweise Auswertungen möglich sein, die zeigen, welche Person im Zeitraum X bis Y die meisten Gegenstände bestellt hat.

### **2.1.7 Nice to have**

Die folgenden Features sind optional und können bei ausreichend Zeit implementiert werden:

- QR-Code/Barcode Generator + Scanner
- Fotos der Gegenstände
- Raumplan der Hochschule, Zuordnung zu nicht beweglichen Gegenständen, Suchen nach Raum
- Erinnerungen für Wartungen (z.B. für elektronische Geräte)
- Protokollierung aller Änderungen eines Gegenstandes

## **2.2 Nicht-Funktionale Anforderungen**

### **2.2.1 Wartbarkeit**

Die Anwendung sollte zusammen mit ihren Abhängigkeiten in einer isolierten Umgebung, einem sogenannten „Container“, bereitgestellt werden. Durch die Containerisierung wird eine hohe Portabilität gewährleistet, die das Deployment der Anwendung für nachfolgende Administratoren vereinfacht. Zudem lässt sich die Anwendung problemlos in eine CI/CD-Pipeline integrieren.

### **2.2.2 Hohe Benutzerfreundlichkeit**

Die Softwareanwendung muss einen intuitiven, schnellen und einfachen Zugriff ermöglichen, der dem Benutzer eine klare und übersichtliche Bedienoberfläche bietet. Zudem soll die Benutzeroberfläche (UI) gestalterisch anspruchsvoll sein.

### 2.2.3 Integration in bestehende IT-Landschaft

Das Inventarisierungsmodul muss über eine REST-API mit dem Bestellsystem „BeSy“ kommunizieren, welches als Single-Component-Angular-Anwendung entwickelt wird. Nach Abschluss einer Bestellung soll das System automatisch einen POST-Request an das Inventarisierungsmodul senden, um die Bestelldaten nahtlos zu übernehmen.

### 2.2.4 Code-Qualität

Die Code-Qualität und die Wartbarkeit der Software haben einen hohen Stellenwert. Zur automatisierten Code-Analyse wird SonarQube in die CI/CD-Pipeline integriert. Ein festgelegter Mindestwert kann als Kriterium für das Pushen von Änderungen auf GitHub dienen. Für das Backend werden Unit-Tests implementiert, während für das Frontend manuelle Testprotokolle ausreichen. Die Code Coverage wird als Bestandteil der „Definition of Done“ berücksichtigt.

### 3 UI Entwürfe

Das Design soll sich am Corporate Design der Hochschule Esslingen orientieren, um eine einheitliche Gestaltung zu gewährleisten. Zudem sollte es an die Bestellsoftware angelehnt sein, um den Nutzern einen vertrauten und intuitiven Zugang zu ermöglichen.

### 3.1 Inventarliste

Abbildung 3.1: Prototyp Inventarliste/Homepage Quelle: Erstellt in Figma /<empty citation>/

Als Homepage soll zunächst die in Abbildung 3.1 abgebildete Inventarliste verwendet werden. Eine dedizierte Homepage kann im späteren Entwicklungsprozess noch hinzugefügt werden, ist aber für eine minimale, funktionierende Software nicht nötig.

Die Inventarliste orientiert sich stark an der bereits bestehenden Excel Tabelle, um eine schnelle Umgewöhnung auf das Inventarisierungsmodul zu unterstützen. Die Liste kann nach jeder Spalte sortiert werden und soll über eine Volltextsuche verfügen, die oben links unter dem Hochschullogo zu sehen ist. Mit dieser kann die Tabelle nach jedem Eintrag durchsucht werden, was aufgrund der geringen formalen Überschneidung der Spalteneinträge zu wenigen falsch positiven Suchergebnissen führen sollte.

Mit den oben rechts positionierten, entsprechend gekennzeichneten Buttons kann zwischen mehreren Seiten geblättert werden, zudem soll die Anzahl an sichtbaren Einträgen pro Seite wählbar sein. Der aktuell unter dem Mauszeiger liegende Eintrag soll zeilenweise farblich hervorgehoben werden, um eine bessere Lesbarkeit zu unterstützen.

Wenn nötig bietet die Inventarliste ein umfangreiches Filtermenü, das über den Button rechts neben der Suchleiste aufgerufen werden kann. So kann eine umfangreiche Sammlung an Filtern zur Verfügung gestellt werden, die trotzdem keine Bildschirmfläche belegt, wenn sie nicht benötigt wird.

#### 3.1.1 Filtermenü

Abbildung 3.2: Prototyp Filtermenü der Inventarliste *Quelle: Erstellt in Figma /<empty citation>/*

Wie in Abbildung 3.2 zu sehen, soll das Filtermenü für jede Kategorie passende Filter zur Auswahl bereitstellen. Für fließende Werte soll in Zukunft, statt der aktuell dargestellten Unter und Obergrenzen Buttons, interaktive Slider verwendet werden. Die eingestellten Grenzwerte sollen eindeutig ablesbar sein und zusätzlich die Möglichkeit einer direkten Eingabe bieten.

Sämtliche Filter sollen frei und interaktiv kombinierbar sein und zudem mit der Suche, als auch der Sortierung nach Tabellenspalten, eindeutige Trefferschnittmengen bilden. Falls während des Entwicklungsprozesses der Bedarf nach weiteren Filtern entsteht, können diese, dank des vertikalen, listenartigen Aufbaus, Problemlos in das Filtermenü integriert werden. Gegenüber einer horizontalen Anordnung der einzelnen Filter, bietet das in Abbildung 3.2 dargestellte System wesentlich mehr Platz und ist flexibler im Umgang mit einer dynamisch wachsenden Anzahl an einzelnen Filtertags.

## 3.2 Detailansicht eines Gegenstandes

In der Inventarliste soll die Möglichkeit gegeben sein einen Eintrag anzuklicken, und eine Detailansicht des Gegenstands anzuzeigen. Diese soll auf einer eigenen Seite angezeigt werden und über eine URL, abhängig von der Inventarnummer, erreichbar sein. So kann auch die User Experience verbessert werden, da direktes aufrufen, beziehungsweise teilen eines Links, über die eindeutige Inventarnummern gewährleistet werden kann. Wie in Abbildung

### 3 UI ENTWÜRFE

---

Kostenstelle: KG1281  
Inventarnummer: 471938  
Firma: Computerteile GmbH  
Preis: 192,20 €  
Inventarisiert am: 12.01.2001  
Seriennummer: SFK490LF  
Standort / Nutzerin: F1.405  
Bestellt von: Prof. Huber

**Erweiterungen**

Erweiterungstyp	Firma	Preis / €	Kostenstelle	Seriennummer	Hinzugefügt von	Hinzugefügt am
Erweiterung A	Firma B	32,12	Kostenstelle 2	123456	Anna Schlosser	27.05.2001
Erweiterung B	Firma B	48,10	Kostenstelle	49295032	Prof. Huber	10.02.2013

**Notizen**  
01.01.2025 - Max Grün: USB Buchse defekt

**Tags**  
Netzteil, Mobil, Zubehör, Laptop Bedarf, ASUS

**Historie**

Änderung	Alter Wert	Neuer Wert	Geändert von	Geändert am
Preis	84,18 €	192,20 €	Max Grün	20.08.2020
Erweiterung B: Seriennummer	-	49295032	Prof. Schiener	26.11.2017

Abbildung 3.3: Prototyp Detailansicht *Quelle: Erstellt in Figma [<empty citation>]*

3.3 zu sehen, sollen alle Informationen der Inventarliste, wie auch zusätzliche Daten angezeigt werden, für die in der Haupttabelle der Bildschirmplatz nicht ausreicht.

Über die Detailansicht soll ein Einblick in die Erweiterungen, die Notizen, die Tags und die Historie, also die Änderungen, die ein Gegenstand bereits durchlaufen hat, möglich sein. Wie in Abbildung 3.3 zu sehen, sind diese Informationen Tabellenweise dargestellt und können bei Bedarf auch ausgeblendet werden. Wenn genügend Zeit zur Verfügung steht, besteht die Möglichkeit noch zusätzliche Informationen, wie beispielsweise Brutto/Nettopreis, hinzuzufügen.

Die Detailansicht kann außerdem im Inventarisierungsprozess als Kontrollansicht wiederverwendet werden, um beispielsweise einen neu inventarisierten Gegenstand zu überprüfen, oder im Falle einer Erweiterung, um den richtigen Artikel aus den Suchergebnissen herauszusuchen.

### 3.3 Inventarisierung

Es sind mehrere Möglichkeiten geplant, Gegenstände zu inventarisieren. Zunächst ist auf der Inventarisierungsstartseite der Inventarisierungs-Wizard zu sehen, auf den in Punkt 3.3.3 noch genauer eingegangen wird. Auf der linken Seite befindet sich ein Untermenü, das Platz für verschiedene kontextabhängige und kontextübergreifende Buttons bietet. In Abbildung 3.4 sind nur die offenen Bestellungen in der Seitenleiste anwählbar.

Der Inventarisierungs-Wizard kann von dieser Seite aus verwendet werden, um einen einzelnen Artikel zu inventarisieren, wenn die Bestellung nicht aus dem Bestellsystem BeSy an das Inventarisierungssystem InSy übertragen wurde. Alternativ kann auch das Menü zur direkten

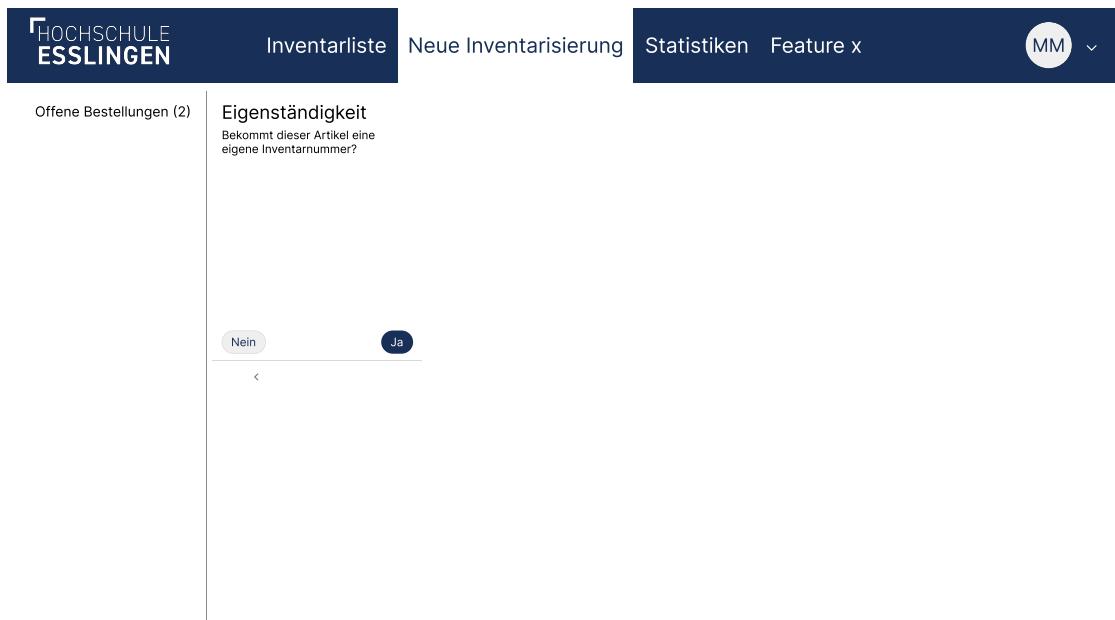


Abbildung 3.4: Prototyp Neue Inventarisierung *Quelle: Erstellt in Figma [<empty citation>]*

Inventarisierung aus Punkt 3.3.4 zum manuellen inventarisieren verwendet werden.

Da aufgrund der Teamumstrukturierung durch die Teamverkleinerung die zur Verfügung stehende Arbeitszeit geringer ist als initial angenommen, ist das Menü zur direkten Inventarisierung der einzige geplante Weg zur Inventarisierung. Der Wizard kann aus Zeitgründen nicht mehr umgesetzt werden. Die API sollte dennoch alle nötigen Endpunkte bereitstellen, falls ein zukünftiges Team den Wizard im Frontend implementieren will.

#### 3.3.1 Bestellübersicht

Über den Button "Offene Bestellungen" wird die in Abbildung 3.5 dargestellte Bestellübersicht geöffnet. Hier werden alle noch nicht vollständig inventarisierten Bestellungen angezeigt, die durch BeSy an InSy übertragen wurden. Diese Seite bietet eine Übersicht über die Bestellungen und deren Artikel. Bestellungen, die keine einzelnen Komponenten enthalten, die als ein Gesamtgegenstand inventarisiert werden sollen, können ausgewählt werden und dann artikelweise mit dem Inventarisierungs-Wizard oder der Direktansicht abgearbeitet werden. Bestellungen, die Einzelteile enthalten, können für das Zusammenfassen in einen oder mehreren Gesamtgegenständen ausgewählt werden und in der Artikelübersicht aggregiert werden.

#### 3.3.2 Artikelübersicht

In der in Abbildung 3.6 dargestellten Artikelübersicht werden zuvor ausgewählte Bestellungen und deren Artikel angezeigt. Hier besteht die Möglichkeit einzelne Artikel auszuwählen und zu einem oder mehreren Gesamtgegenständen zusammenzufassen. Wenn möglich sollen diesen

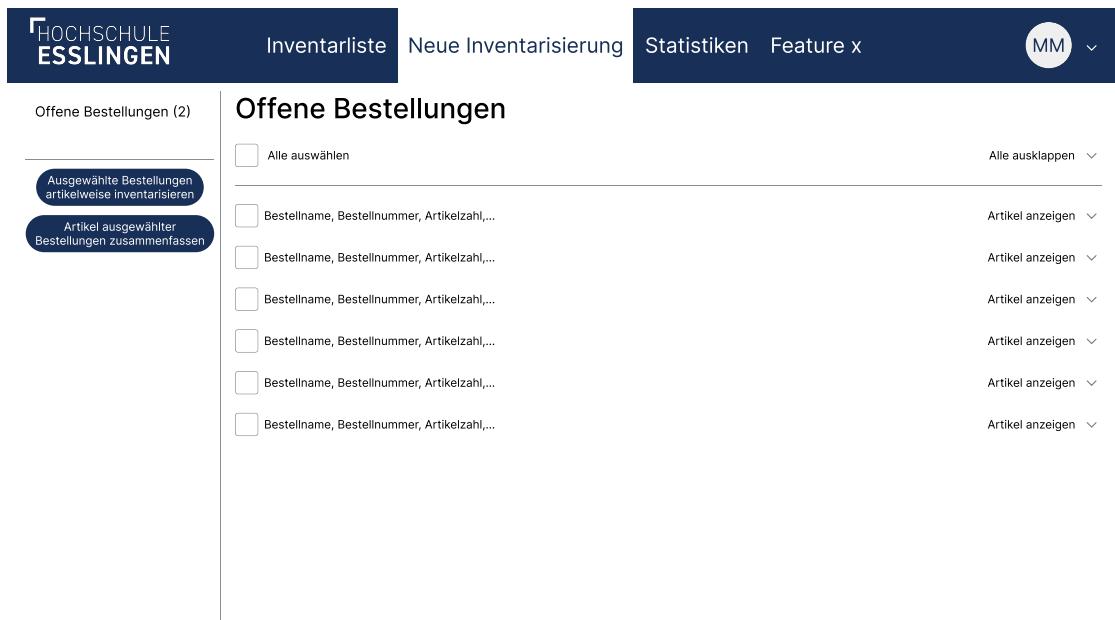


Abbildung 3.5: Prototyp Bestellübersicht *Quelle: Erstellt in Figma [<empty citation>]*

auch individuelle Namen gegeben werden können. Anschließend können die ausgewählten, einzelnen oder aggregierten Artikel über den entsprechenden Button in der Seitenleiste inventarisiert werden, indem für jeden Artikel der Wizard durchlaufen wird. Alternativ kann statt des Wizards auch die direkte Inventarisierungsübersicht verwendet werden. Bei aggregierten Gegenständen werden automatisch Notizen generiert, in denen die einzelnen Komponenten und deren Bestellungsinformationen aufgeführt werden.

#### 3.3.3 Inventarisierungswizard

In Abbildung 3.7 sind die einzelnen Schritte des Inventarisierungs-Wizards dargestellt. In jedem Schritt des Wizards werden ein bis zwei zusammenhängende Daten über den zu inventarisierten Artikel abgefragt, das dient dem Zweck, den Workflow zu strukturieren und Unübersichtlichkeit zu Vermeiden.

Der Wizard soll alle Felder automatisch vorausfüllen, sofern eine gültige Bestellnummer hinterlegt, beziehungsweise eingegeben wurde und diese im Backend oder über BeSy erfasst ist. Optionale Einträge sollen immer klar durch den entsprechenden “Überspringen” Button gekennzeichnet sein. Außerdem soll es die Möglichkeit geben, den Wizard jederzeit zu beenden und in die Gesamtübersicht zu wechseln, die in jedem Fall am Ende auch als Kontrollansicht dient.

Abhängig davon, über welchen Weg der Wizard aufgerufen wird, sollen einige Schritte übersprungen werden oder dazukommen. Die Bestellnummer wird nur abgefragt, wenn die zugehörige Bestellung nicht von BeSy hinterlegt wurde. Ebenso entfällt die Abfrage der Eigenständigkeit, wenn es sich bei dem zu inventarisierten Gegenstand um eine bereits auf der Artikelübersicht zusammengestellte Aggregation handelt. Wird diese Frage verneint, wird direkt, die auf der Abbildung ganz unten rechts dargestellte Abfrage nach der Inventarnummer

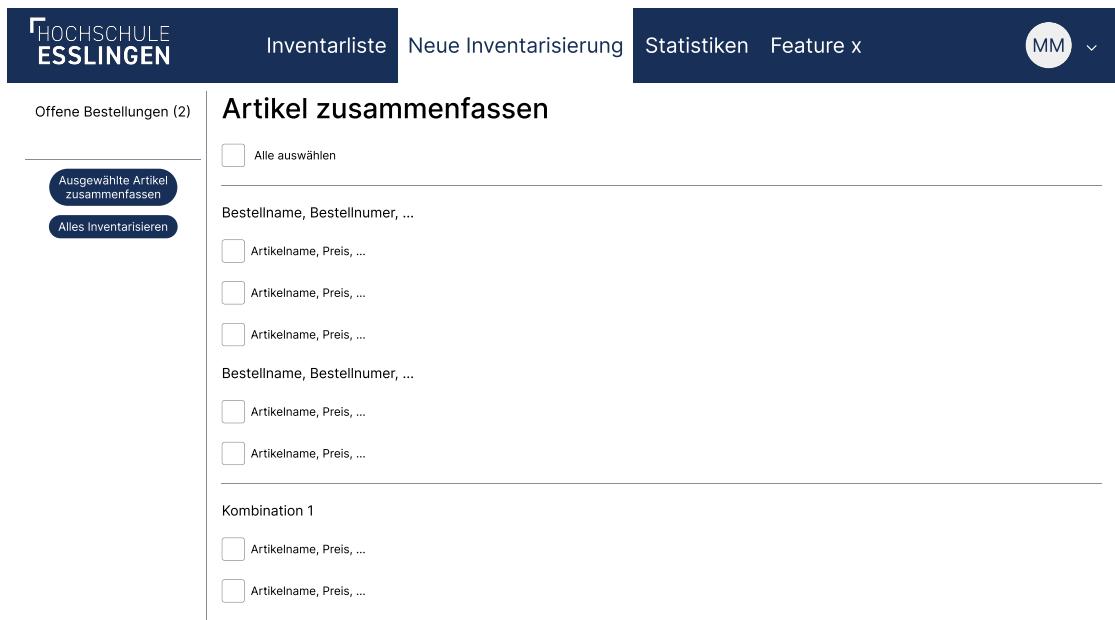


Abbildung 3.6: Prototyp Artikelübersicht *Quelle: Erstellt in Figma [<empty citation>]*

des zugehörigen Gegenstands angezeigt. Anschließend soll rechts neben dem Wizard, der bestehende Artikel angezeigt, und der Wizard normal durchlaufen werden. Die in diesem Fall abgefragten Informationen sollen verwendet werden, um automatisch eine Änderungsnotiz für den bestehenden Artikel zu verfassen.

#### 3.3.4 Direkte Inventarisierung

Die direkte Inventarisierung soll über eine klassische Inputform erfolgen, wie in Abbildung 3.8 dargestellt. Die Seitenleiste aus vorherigen Entwürfen ist durch eine horizontale Leiste ersetzt worden um Bildschirmplatz zu sparen. Für entsprechende Daten, wie beispielsweise die Kostenstelle oder die liefernde Firma, sollen Drop Down Menüs vorhanden sein. Wenn der gewünschte Eintrag noch nicht vorhanden ist, können diese aber auch als reguläres Textfeld verwendet werden.

Die freiählbaren Tags können ebenfalls über ein Drop Down Menü ausgewählt, oder durch Texteingabe neu erstellt werden. Die ausgewählten Tags werden unter dem Menü angezeigt und können durch anklicken wieder entfernt werden. Notizen können über ein Textfeld hinzugefügt werden. Author und Datum werden anhand des/der angemeldeten Nutzer:in automatisch gespeichert.

Wenn die direkte Inventarisierung für einen von BeSy importierten Artikel ausgeführt wird, sind alle Felder mit den jeweils vorhandenen Daten vorausgefüllt.

Wenn es sich bei dem direkt zu Inventarisierenden Gegenstand um eine Erweiterung eines bereits inventarisierten Artikels handelt, kann wie in Abbildung 3.9 zu sehen, die Detailansicht des bereits existierenden Artikels neben der Inputform angezeigt werden. Diese Oberfläche kann eventuell auch als Bearbeitungsansicht mit einer Livepreview Funktion des fertigen Artikels wiederverwendet werden.

<p><b>Bestellnummer</b> Die Bestellnummer wird zum automatisierten Vorausfüllen verwendet.</p> <input type="text" value="Bestellnummer"/> <p><a href="#">Überspringen</a> <a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Eigenständigkeit</b> Bekommt dieser Artikel eine eigene Inventarnummer?</p> <p><a href="#">Nein</a> <a href="#">Ja</a></p> <p>&lt;</p>	<p><b>Gerätetyp / Software</b> Um welchen Gerätetyp oder welche Software handelt es sich?</p> <input type="text" value="Artikeltyp"/> <p><a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Preis</b> der Bruttopreis des Artikels.</p> <input type="text" value="0,00"/> €
<p><b>Datum</b> das Rechnungsdatum der Bestellung.</p> <input type="text" value="TT.MM.JJJJ"/> <p><a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Tags</b> optionale Tags hinzufügen um die Suche zu vereinfachen.</p> <p><a href="#">Tag</a> <a href="#">Hinzufügen</a></p> <p>Tag 1 Beispiel Tag</p> <p><a href="#">Überspringen</a> <a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Seriennummer</b> bitte trage die Seriennummer des Artikels ein.</p> <input type="text" value="Seriennummer"/> <p><a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Foto</b> füge ein optionales Foto des Geräts hinzu. "Scanne QR-Code oder sonstiges..."</p> <p><a href="#">Überspringen</a> <a href="#">Weiter</a></p> <p>&lt;</p>
<p><b>Inventarnummer</b> Die Inventarnummer, unter der dieser Artikel hinzugefügt werden soll.</p> <input type="text" value="Inventarnummer"/> <p><a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Standort / Nutzer:in</b> der geplante Standort beziehungsweise die verwendende Person.</p> <p><a href="#">Name oder Raum</a></p> <p><b>Bestellt von</b></p> <p><a href="#">Name</a></p> <p><a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Notizen</b> allgemeine Notizen, unter anderem, um Erweiterungen festzuhalten.</p> <p><a href="#">Notiz</a></p> <p>01.01.2025 Änderung 1</p> <p><a href="#">Überspringen</a> <a href="#">Weiter</a></p> <p>&lt;</p>	<p><b>Inventarnummer</b> bitte klebe die Inventarnummer auf das Gerät und überprüfe sie auf Korrektheit.</p> <input type="text" value="Inventarnummer"/> <p><a href="#">Weiter</a></p> <p>&lt;</p>

Abbildung 3.7: Prototyp Inventarisierungswizard Quelle: Erstellt in Figma [[<empty citation>](#)]

#### 3.4 Statistiken

InSy soll einfache Statistiken über aktuelle Bestellungen, beziehungsweise Inventarisierungen bereitstellen. In Abbildung 3.10 sind zum Beispiel die inventarisierten Artikel der letzten 30 Tage und deren Preis und Besteller:in zu sehen.

Auch dieses Feature muss, aufgrund der Teamverkleinerung, stark eingegrenzt oder eventuell ganz weggelassen werden. Die Statistiken statisch und nicht interaktiv zu berechnen und darzustellen, stellt eine Möglichkeit dar diese zu vereinfachen. Insgesamt ist dieses Feature außerdem ein Nice-To-Have und nicht zwingend für die Funktionalität der Software ausschlaggebend.

### 3 UI ENTWÜRFE

---

Abbildung 3.8: Prototyp Direkte Inventarisierung Quelle: Erstellt in Figma []

Abbildung 3.9: Prototyp Erweiterung einem Gegenstand hinzufügen Quelle: Erstellt in Figma []



Abbildung 3.10: Prototyp Statistiken über aktuelle Inventarisierungen *Quelle: Erstellt in Figma [<empty citation>]*

## 4 Architektur

### 4.1 Technologische & konzeptionelle Architektur

Die Webapplikation basiert auf einer klassischen Three-Tier Architektur, die die Anwendung in die Schichten Präsentation, Applikation und Daten unterteilt. Diese Architektur bietet zahlreiche Vorteile, wie etwa eine schnellere und unabhängiger Entwicklung, da verschiedene Teammitglieder an unterschiedlichen Teilen der Software arbeiten können. Zudem lässt sich die Anwendung dadurch wesentlich besser skalieren, da jede Schicht unabhängig voneinander agiert., entwickelt und gewartet werden kann.

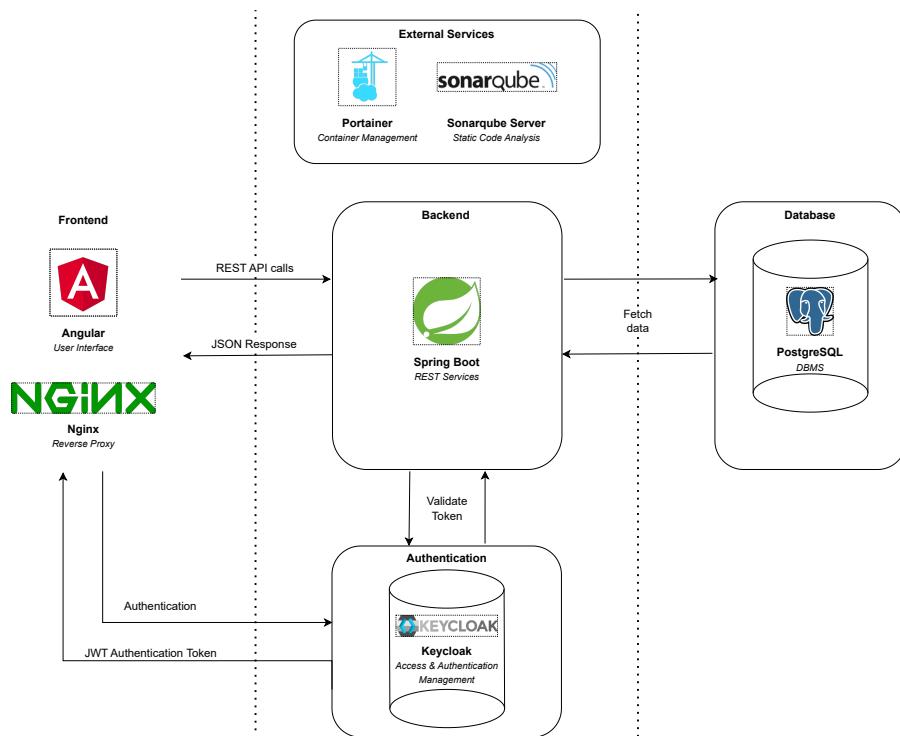


Abbildung 4.1: Three-Tier Architecture *Quelle: in Anlehnung [<empty citation>]*

### 4.1.1 Strukturschicht

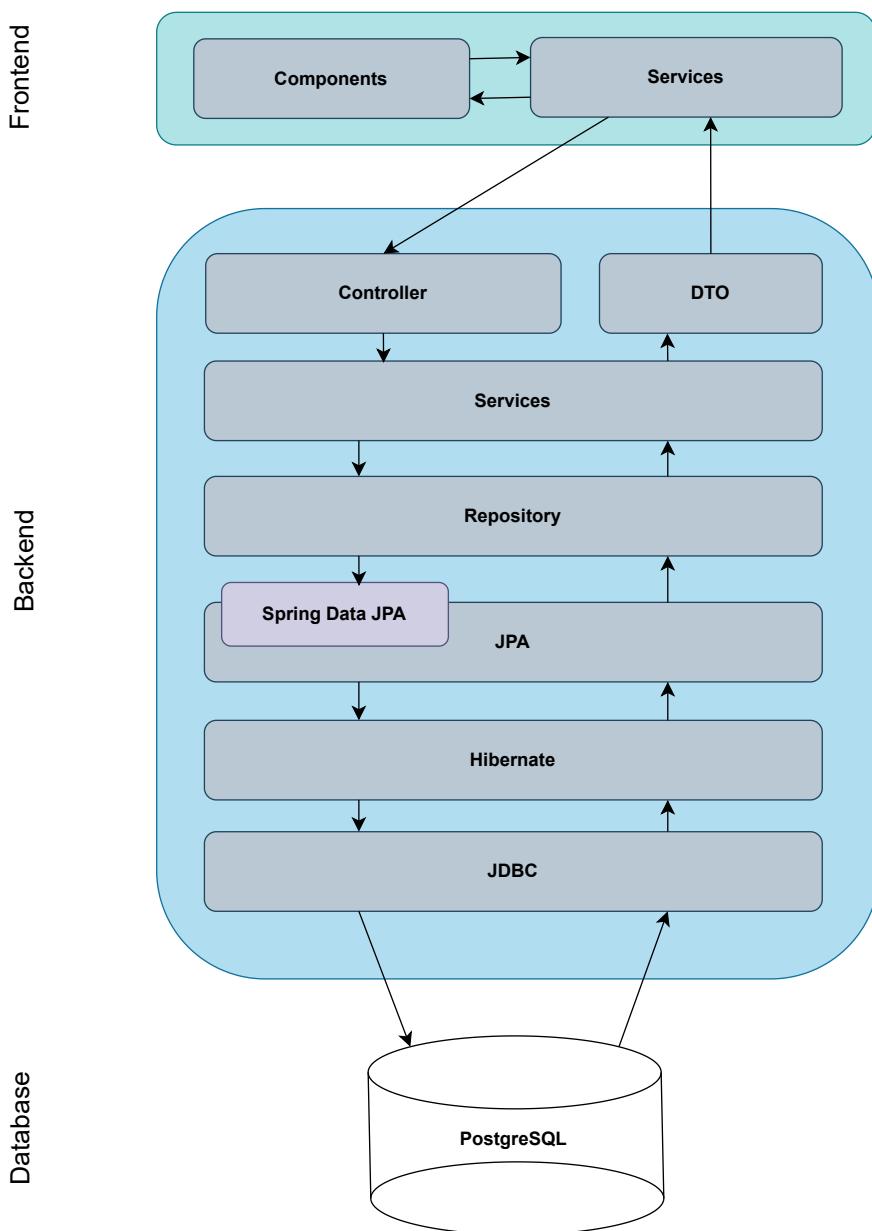


Abbildung 4.2: Strukturschicht *Quelle: in Anlehnung [<empty citation>]*

## 4.2 Deployment Prozess

### 4.2.1 Einleitung

Zur Vereinfachung der Softwareintegration sowie zur Automatisierung von Testprozessen kommt eine CI/CD-Pipeline zum Einsatz. Diese ermöglicht es, Änderungen an der Codebasis in das bestehende System zu integrieren und frühzeitig auf Fehler zu prüfen.

Sobald Entwickler ihre Codeänderungen in das zentrale GitHub-Repository pushen, wird ein GitHub-Workflow ausgelöst. Dieser automatisiert wiederkehrende Aufgaben wie das Erstellen von Docker-Containern, das Ausführen von Backend-Tests sowie die Durchführung statischer Code-Analysen.

### 4.2.2 Github Workflows

Bei der Wahl der Workflow-Technologie fiel die Entscheidung auf GitHub Workflows, da sich diese nahtlos in das von GitHub bereitgestellte Repository integrieren lassen. Ein weiterer Vorteil ist der GitHub Marketplace for Workflows, welcher eine umfangreiche Sammlung bereits vordefinierter Workflows bietet. Diese stammen sowohl von anderen Entwicklern als auch von Unternehmen und können problemlos in das eigene Projekt eingebunden werden. Dadurch lassen sich zusätzliche Aufgaben automatisieren und Entwicklungsaufwand sowie Zeit weiter reduzieren.

### 4.2.3 Aufbau eines Workflows

Ein Workflow definiert zunächst, auf welchen Branches und innerhalb welcher Verzeichnispfade er auf Änderungen reagieren soll. So kann beispielsweise konfiguriert werden, dass ein Workflow ausschließlich auf dem Branch ‘backend’ ausgelöst wird, wenn Änderungen im Pfad ‘backend/repository’ vorgenommen werden.

Diese Konfiguration ermöglicht eine granulare Steuerung der Ausführung von Workflows. Dadurch wird vermieden, dass bei jeder geringfügigen Änderung ein vollständiger Workflow ausgeführt wird, was zu unnötigem Overhead führen würde. Stattdessen werden Workflows nur dann gestartet, wenn relevante, intern geprüfte Änderungen vorliegen, die in zentrale Branches wie ‘main’ überführt werden sollen.

### 4.2.4 Docker

Nach der initialen Trigger-Definition eines Workflows folgen die sogenannten Jobs, welche die eigentlichen Prozessschritte abbilden. In nahezu jedem Workflow ist der Schritt ‘actions/check-out@v4‘ erforderlich, um dem Workflow den Zugriff auf den Code im GitHub-Repository zu ermöglichen [1].

Im nachfolgenden Beispiel wird über das Schlüsselwort ‘uses‘ ein bereits vorgefertigter Docker-Workflow eingebunden. Hierbei kommen Zugangsdaten (Credentials) zum Einsatz,

die zuvor als Secrets im Repository hinterlegt wurden. Diese Secrets sind standardmäßig nur für den Repository-Owner einsehbar und dienen dazu, Umgebungsvariablen bereitzustellen. Auf diese Weise kann sich die VM bzw. der Container, in dem der Workflow ausgeführt wird, bei Docker Hub authentifizieren.

Ein weiterer Code-Snippet zeigt, wie das erstellte Docker-Image in eine eigene Docker Registry gepusht wird. Dabei müssen der entsprechende Build-Kontext – also der Pfad zur Dockerfile – sowie spezifische Tags angegeben werden. Tags dienen unter anderem der klaren Unterscheidung zwischen verschiedenen Komponenten wie Frontend und Backend, indem sie entsprechend benannt oder mit einem Suffix versehen werden.

[1] GitHub Actions: [actions/checkout](<https://github.com/actions/checkout>)

### 4.2.5 Sonarqube

Im Rahmen der kontinuierlichen Qualitätskontrolle kommt auch SonarQube zum Einsatz. Dabei handelt es sich um ein Tool zur statischen Codeanalyse, das den Quellcode auf potenzielle Fehler, Sicherheitsrisiken und Verstöße gegen Programmierstandards überprüft.

Der Einsatz von SonarQube soll insbesondere unserem noch vergleichsweise unerfahrenen Team dabei helfen, die Codequalität nachhaltig zu verbessern, versteckte Fehler frühzeitig zu erkennen und bewährte Best Practices in der Softwareentwicklung zu fördern.

In der obigen Grafik ist ein Beispiel für einen sogenannten Issue dargestellt. Während der Analyse des Codes identifiziert SonarQube verschiedene Issues, die auf potenzielle Probleme im Code hinweisen. Diese Issues werden in mehrere Kategorien unterteilt, darunter die Softwarequalität (z.B. Security, Reliability, Maintainability), der Schweregrad (z.B. Blocker, High, Medium, Low, Info), sowie die Programmiersprache – in unserem Fall Java – und viele weitere Kriterien.

Diese Kategorisierung ermöglicht es den Entwicklern, gezielt nach den schwerwiegendsten Problemen zu suchen und Prioritäten bei der Behebung zu setzen. Darüber hinaus bietet SonarQube einen speziellen Reiter für Security Hotspots, der schwerwiegende Sicherheitsrisiken hervorhebt. Ein Beispiel aus unserer Entwicklungsumgebung: Das Tool hat erkannt, dass die Cross-Site Request Forgery (CSRF)-Schutzmaßnahme deaktiviert wurde, was ein potenzielles Sicherheitsrisiko darstellt.

**Ablauf der Workflow Analyse** Der angestoßene Workflow spezifiziert ein Verzeichnis im GitHub-Repository, das vom SonarScanner aufgerufen und analysiert wird. Der SonarScanner übermittelt die Ergebnisse der Analyse an unseren eigens gehosteten SonarQube-Server zur weiteren Auswertung.

Es ist wichtig zu beachten, dass SonarQube selbst keine Unit-Tests für JavaScript oder TypeScript durchführen kann. Daher werden diese Tests im Frontend mit den Frameworks Karma und Jasmine ausgeführt. Der dabei generierte Coverage Report wird anschließend von SonarQube gelesen und in die Analyse integriert. **\*\*WEITERE INFORMATIONEN HIERZU UNTER KAPITEL XYZ.\*\***

**Sonarqube Server** Die SonarQube UI wird auf der Development VM gehostet und ist unter folgender Domäne erreichbar: [<http://sonar.test.insy.hs-esslingen.com/>](<http://sonar.test.insy.hs-esslingen.com/>). Alle fünf Teammitglieder verfügen über einen regulären Nutzer-Account, der ihnen den Zugang zu den oben gezeigten Analyse-Ergebnissen ermöglicht.

Zusätzlich existiert ein Administrator-Account, auf den ausschließlich Sandro Zugriff hat. Dieser Account dient der Konfiguration des SonarQube-Servers, wie etwa der Verknüpfung mit dem GitHub-Account. Der Administrator-Account wurde bewusst nicht mit dem gesamten Team geteilt, da der selbst gehostete SonarQube-Server mit Sandros privatem GitHub-Nutzerkonto verknüpft ist.

Referenzen: Sonarqube: <https://www.sonarsource.com/products/sonarqube> Sonarqube Server Docker Image: [https://hub.docker.com/\\*sonarqube](https://hub.docker.com/*sonarqube)

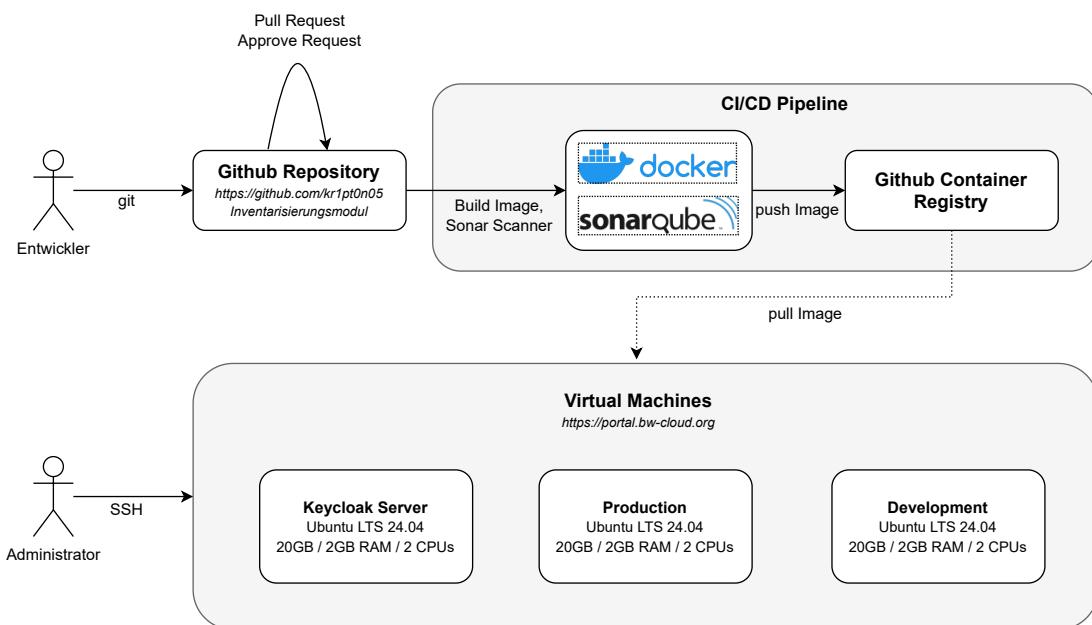


Abbildung 4.3: CI/CD Pipeline *Quelle: in Anlehnung [<empty citation>]*

### 4.3 Frontend

Aufgezählt werden die hier wesentlichen Technologien, welche bewusst von uns eingesetzt und genutzt wurden. Andere Technologien, die bspw. als Produkt einer Bibliothek mit einge-

bunden wurden, werden nicht weiter erläutert.

Darüber hinaus erfolgt eine exemplarische Vorstellung ausgewählter Angular-Komponente und -Services der Benutzeroberfläche. Im Fokus steht dabei die **\*\*Inventarliste\*\***, da sie den funktionalen Kern der Anwendung bildet. Weitere Funktionalitäten werden nur am Rande behandelt. Der vollständige Quellcode ist jedoch über das GitHub-Repository öffentlich zugänglich.

Zusätzlich wird die grundlegende Konfiguration des Angular-Projekts vorgestellt.

Abschließend gibt eine Versionsmatrix einen Überblick über die im Projekt verwendeten Framework- und Bibliotheksversionen.

### 4.3.1 Technologien

**Angular** Das Frontend der Anwendung wurde mit dem weit verbreiteten TypeScript-Framework Angular entwickelt. Die Entscheidung für Angular basiert unter anderem auf dem Wunsch unseres Projektbetreuers Andreas Heinrich, da auch das angebundene Bestellsystem BESY, mit dem unsere Anwendung interagieren soll, in Angular realisiert wurde.

Angular bietet eine umfangreiche Sammlung integrierter Funktionalitäten wie Routing, Formularverarbeitung, HTTP-Client, sowie State Management. Diese ermöglichen eine strukturierte und effiziente Entwicklung komplexer Anwendungen. Die klare Architektur und die strikte Trennung von Komponenten fördern sowohl die Wartbarkeit als auch die Skalierbarkeit der Anwendung.

Ein weiterer Vorteil liegt in der hohen Modularität des Frameworks: Die Verwendung wiederverwendbarer Komponenten vereinfacht die Entwicklung dynamischer Benutzeroberflächen erheblich. Zudem lässt sich Angular problemlos mit zusätzlichen Abhängigkeiten und externen Bibliotheken erweitern.

Angular wird aktiv von Google entwickelt und ist als Open-Source-Projekt verfügbar, was eine kontinuierliche Weiterentwicklung und eine große Entwickler-Community sicherstellt. Angular basiert auf TypeScript, einer von Microsoft entwickelten Erweiterung von JavaScript, die statische Typisierung sowie moderne Sprachfeatures bietet. Diese Eigenschaften führen zu einer höheren Codequalität und erleichtern das Debugging, da viele potenzielle Fehler bereits zur Entwicklungszeit erkannt werden.

**Referenz:** <https://angular.dev>

**Tailwind CSS** Zur Gestaltung der Benutzeroberfläche kommt das Open-Source-CSS-Framework Tailwind CSS zum Einsatz. Es ermöglicht die schnelle und konsistente Umsetzung moderner und ansprechender UI-Komponenten, ohne das klassische, manuelle Pflegen von unüberschaubaren CSS Dateien.

Tailwind CSS definiert eigene CSS-Klassen, welche direkt im HTML-Markup verwendet werden können. Die Klassen beschreiben einzelne Stil-Eigenschaften wie Farben, Abstände, Größen oder Layout und werden über das Attribut class eingebunden. Dadurch lässt sich das Design direkt im Kontext der HTML-Struktur entwickeln und anpassen, was den Entwicklungsprozess erheblich vereinfacht.

Beispielsweise setzt der folgende Code-Snippet mithilfe von Tailwind Klassen die Hintergrundfarbe auf Weiß ('bg-white'), die Textfarbe auf ein Grauton ('text-gray-700') und definiert eine Mindesthöhe, die der Höhe des Viewports entspricht ('min-h-screen'):

Die folgendene vordefinierte Klasse veranschaulicht, wie Tailwind CSS einzelne Stilmerkmale abbildet: **Referenzen:** <https://tailwindcss.com>

**Angular Material** Zur weiteren Unterstützung der UI-Entwicklung kommt Angular Material zum Einsatz, eine Komponentenbibliothek, die nahtlos in Angular integriert ist. Sie bietet eine Vielzahl vorgefertigter und responsiver Komponenten für die Benutzeroberfläche.

Zu den enthaltenen Komponenten zählen unter anderem Autocomplete-Felder, Datepicker, Formularelemente, Eingabefelder sowie sortierbare Tabellen. Der Einsatz dieser Komponenten ermöglicht eine erhebliche Zeitersparnis im Entwicklungsprozess, da viele häufig benötigte Funktionalitäten nicht von Grund auf neu entwickelt werden müssen.

Trotz der Vorteile ist bei der Integration ein gewisser Konfigurationsaufwand erforderlich. Insbesondere kam es im Projekt zu Kompatibilitätsproblemen mit der eingesetzten Angular-Version, sodass einzelne Komponenten manuell angepasst oder durch Alternativen ersetzt werden mussten. Siehe <https://github.com/kr1pt0n05/Inventarisierungsmodul/issues/24> (Ausformulieren lösen!)

**Referenz:** <https://material.angular.dev>

### 4.4 Backend

Das Backend wird mit dem ebenfalls sehr populären Java-Framework Spring Boot entwickelt, das Teil des umfassenden Spring Frameworks ist und von dessen riesigem Ökosystem profitiert. Durch die einfache Konfiguration und die breite Unterstützung innerhalb der Community ist Spring Boot eine besonders gute Wahl. Trotz der Empfehlung für Node.js, das im Bestellsystem BeSy verwendet wird, haben wir uns für Spring Boot entschieden, da unser Team bereits erste Erfahrungen mit diesem Framework gesammelt hat, was uns einen kleinen Zeitvorteil verschafft.

Spring Boot ermöglicht es uns, schnell und effizient REST-Applikationen zu entwickeln, mit denen das Frontend interagieren kann. Die Architektur gliedert sich dabei in drei Schichten: Controller, Service und Repository. Der Controller ist dafür zuständig, alle API-Anfragen entgegenzunehmen und an das Service Layer weiterzuleiten. Auch die Verarbeitung von Cookies

erfolgt hier. Im Service Layer erfolgt die zentrale Business Logik. Unter anderem erfolgt hier die Datenmanipulation, beispielsweise das Erstellen oder Aktualisieren von Benutzerkonten, das Hinzufügen von Inventargegenständen oder das Aktualisieren des Lagerbestands nach einem Kauf. Zudem interagiert das Service Layer mit dem Repository Layer, welches für die Kommunikation mit der Datenbank und die Ausführung von CRUD-Operationen verantwortlich ist.

Für das Repository Layer setzen wir in Spring Boot auf Spring Data JPA, eine Abstraktion von JPA (Jakarta Persistence API). JPA ist eine Spezifikation für das Persistieren von Java-Objekten in einer relationalen Datenbank. JPA bietet die eigene Abfragesprache Java Persistence Query Language (JPQL) an, die mit Java-Klassen arbeitet, statt direkt auf Datenbanktabellen zuzugreifen. So würde man beispielsweise statt "SELECT \* FROM users" die Abfrage "SELECT u FROM User u" verwenden. JPA schließt den Einsatz von SQL jedoch nicht aus, sondern bietet vielmehr eine zusätzliche Möglichkeit der Abfrage. Außerdem gibt es die Möglichkeit mit JPA Paging und Sorting über Query-Parameter in der URL zum implementieren, wodurch viele nötige Filtermethoden für unsere Applikation direkt über die URL an das Backend mitgegeben werden und dort direkt auf Datenbankabfragen angewendet werden können.

Eine Spezifikation von JPA ist Hibernate, ein Framework, das die Arbeit mit relationalen Datenbanken in Java-Anwendungen deutlich vereinfacht. Hibernate übernimmt die automatische Abbildung von Java-Klassen auf Datenbanktabellen und reduziert damit den manuellen Aufwand für SQL-Code. Entwickler können so effizient und objektorientiert auf Daten zugreifen. Ein wesentlicher Vorteil von Hibernate ist seine Datenbankunabhängigkeit – es unterstützt verschiedene SQL-Dialekte und ermöglicht damit eine flexible Nutzung unterschiedlicher Datenbanksysteme. Zusätzlich bietet Hibernate ein integriertes Caching, das die Performance bei wiederholten Datenbankabfragen verbessert. Funktionen wie Lazy Loading, automatische Schema-Generierung und Transaktionsmanagement erleichtern die Entwicklung und sorgen für eine saubere Trennung von Geschäftslogik und Datenzugriff.

Durch die vollständige Unterstützung der JPA-Spezifikation lässt sich Hibernate problemlos in moderne Java-Frameworks wie Spring integrieren und bietet gleichzeitig alle Vorteile einer standardisierten Schnittstelle.

### 4.5 Datenbank

PostgreSQL hat sich als führender Open-Source-Konkurrent zu einem weit verbreiteten relationalen Datenbankmanagementsystem (DBMS) etabliert. Es bietet umfassende Funktionen wie die Unterstützung von JSON-Daten, Volltextsuche und benutzerdefinierten Datentypen. Zudem zeichnet es sich durch enorme Skalierbarkeit aus und ist sowohl für kleine als auch für sehr große Datenbanken geeignet. Durch die hohe ACID-Konformität gewährleistet es

eine ausgezeichnete Datenintegrität und Transaktionssicherheit, während die engagierte Community kontinuierlich neue Funktionen und Updates bereitstellt.

Ergänzen: Trigger

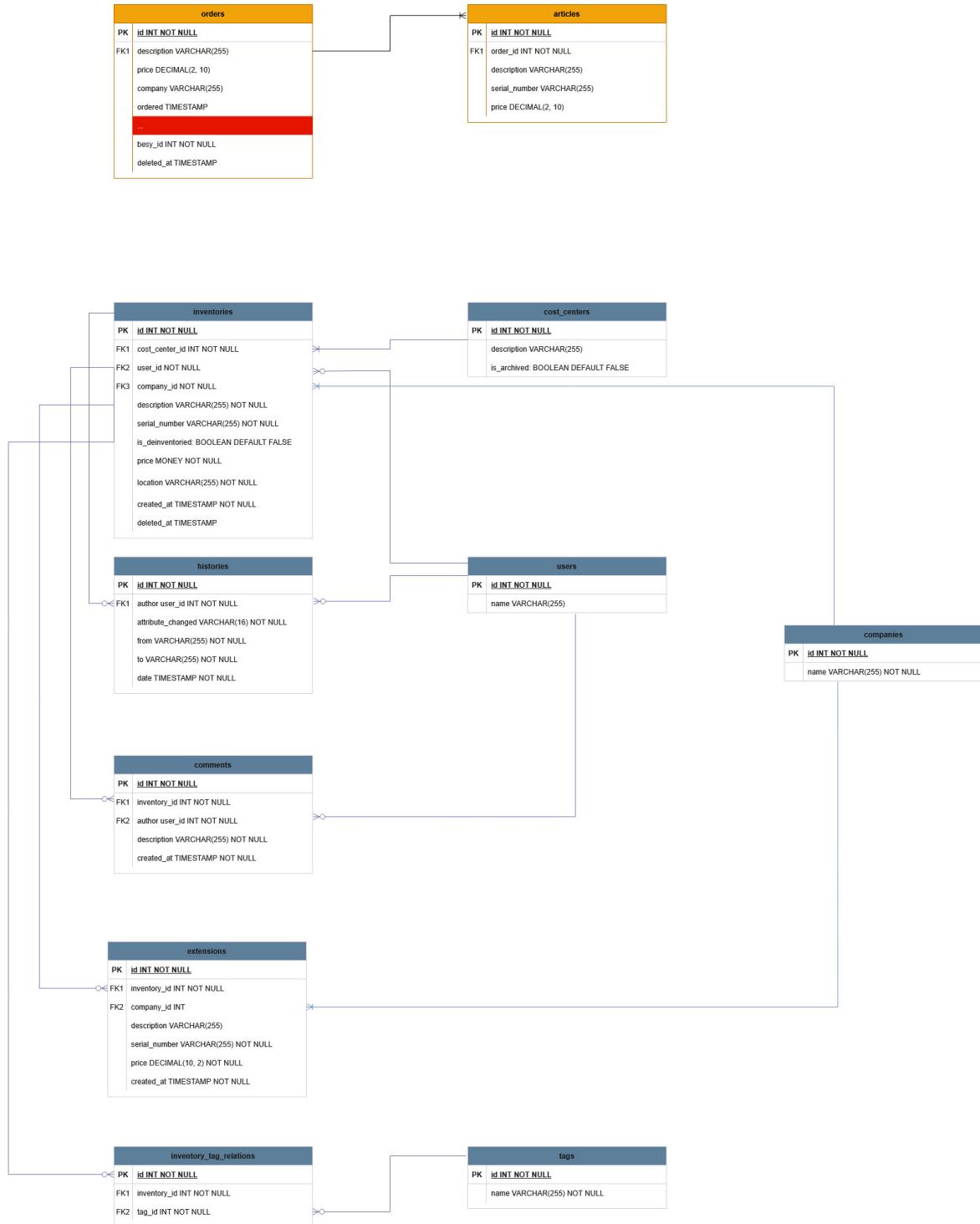


Abbildung 4.4: Datenbankmodell *Quelle: in Anlehnung [<empty citation>]*

## 4.6 Keycloak

Zur Absicherung unserer Schnittstellen verwenden wir Keycloak, ein weit verbreitetes Open-Source-System für Authentifizierungs- und Identitätsmanagement. Dieser Abschnitt bietet einen Überblick über die grundlegenden Konzepte von OpenID Connect (OIDC) und OAuth 2.0, beschreibt die Einrichtung von Keycloak sowie die Integration von Frontend und Backend in die Authentifizierungsinfrastruktur.

### 4.6.1 Keycloak

Keycloak bietet eine Vielzahl an Funktionen zur Absicherung von Ressourcen und stellt eine zentrale Komponente im Bereich Authentifizierungs- und Identitätsmanagement dar. Der Keycloak Server Administration Guide dokumentiert umfassend die Konfigurations- und Sicherheitsoptionen des Systems.

Da die Anwendung im vorliegenden Projekt durch das hochschuleigene Netzwerk bereits in hohem Maße geschützt ist und nur ein begrenzter Personenkreis potenziell Zugriff auf das Inventarisierungsmodul hat, wurde im Rahmen dieses Projekts auf eine minimalistische Keycloak-Konfiguration mit grundlegenden Sicherheitsmaßnahmen gesetzt. Eine vollumfängliche Integration würde den Umfang des Projekts überschreiten. Im Folgenden werden ausgewählte Funktionen beschrieben, die bei der Konzeption berücksichtigt wurden:

**SSL-Verschlüsselung** Keycloak wird in einem isolierten Docker-Netzwerk betrieben und ist über einen vorgesetzten Reverse Proxy angebunden, der die Kommunikation mit dem Nutzer vollständig verschlüsselt. Unter diesen Voraussetzungen kann auf eine zusätzliche Verschlüsselung der Verbindung zwischen Reverse Proxy und Keycloak verzichtet werden. Für sicherheitskritische Anwendungen ist es jedoch empfehlenswert, auch diese interne Verbindung per SSL zu schützen.

**E-Mail Integration** Zum aktuellen Zeitpunkt ist unklar, ob und wie der hochschuleigene Mailserver an Keycloak angebunden werden kann. Daher werden E-Mail-Adressen innerhalb dieses Projekts lediglich simuliert.

**Realm-Einstellungen** Zu den konfigurierten Grundeinstellungen gehören:

- Passwort-Zurücksetzen über „Passwort vergessen“-Funktion
- Möglichkeit zur Änderung der E-Mail-Adresse
- Deaktivierung der Selbstlöschung von Benutzerkonten – diese Funktion bleibt ausschließlich dem Realm-Administrator vorbehalten

- Konfiguration von Token-Gültigkeitsdauern und Session-Timeouts zur Erhöhung der Sicherheit

**Password-Richtlinien:** Basierend auf Empfehlungen des Bundesamts für Sicherheit in der Informationstechnik (BSI) wurden restriktive Passwortvorgaben implementiert. Es wurde die Strategie „kurz und komplex“ gewählt, da solche Passwörter besser merkbar und gleichzeitig sicher sind. Die Richtlinien umfassen:

- Mindestlänge: 8–12 Zeichen
- Verwendung von mindestens vier Zeichentypen (Großbuchstaben, Kleinbuchstaben, Zahlen, Sonderzeichen)

Quelle: BSI Faktenblatt: Sichere Passwörter

**Sitzungsbegrenzung (Session Limits):** Zur Eindämmung potenzieller Missbrauchsszenarien kann die Anzahl gleichzeitig aktiver Sitzungen pro Nutzer auf eine beschränkt werden.

**Schutz vor Brute-Force-Angriffen:** Bei mehrfach fehlerhafter Passworteingabe kann der betroffene Account temporär oder dauerhaft gesperrt werden. Dies erschwert unautorisierten Zugriff durch automatisierte Angriffe.

**Nicht realisierte Funktionen (aus Projektumfang ausgeschlossen):** Obwohl Keycloak zahlreiche weitere Integrationsmöglichkeiten bietet, wurden diese im Rahmen dieses Projekts nicht berücksichtigt. Dazu zählen:

- Anbindung an Active Directory
- Verwendung von Google reCAPTCHA
- Einsatz von Wiederherstellungscodes
- Unterstützung für SAML oder externe Identity Provider wie Google oder Facebook

Keycloak Offizielle Website

Keycloak Server Administration Guide – Security Hardening

**Reverse Proxy** To Do: Ergänzen

-SSL Zertifikate, ...

Authentication OIDC To Do: Ergänzen Stichwörter: Resource Owner, Client, Identity Provider, Resource Server (Siehe Nitzsche Folien)

Authorization OAuth 2.0 To Do: Ergänzen

Authentication Code Flow (Grant) To Do: Ergänzen

JSON Web Token To Do: Ergänzen

PKCE To Do: Ergänzen

Setup To Do: Ergänzen

Configure Spring Boot To Do: Ergänzen

Configure Angular To Do: Ergänzen

Configure Keycloak To Do: Ergänzen

## 5 Projektmanagement

### 5.1 Entwicklungsprozess und Methodik

Für die Softwareentwicklung wird die Scrum-Methodik eingesetzt. Durch den Fokus auf ein Minimum Viable Product (MVP) mit inkrementellen Verbesserungen wird eine effiziente und iterative Entwicklung angestrebt.

Die wöchentlichen Meetings werden vom Scrum Master geleitet, der aus den Reihen des Teams gestellt wird. Die Rollen des Scrum Masters und des Protokollanten rotieren im wöchentlichen Wechsel.

Aufgaben, die ausschließlich die Programmierung betreffen, werden zusätzlich auf einem Kanban-Board in GitHub verwaltet.

### 5.2 Versionsverwaltung

Für die Versionsverwaltung unseres Projektes benutzen wir ein öffentliches Repository auf GitHub.

### 5.3 Meetings

Die regelmäßigen Projektmeetings finden wöchentlich mittwochs um 14:00 Uhr über WebEx statt. Teilnehmer sind das Projektteam, der Betreuer und (nach Möglichkeit) der Kunde.

In den Meetings werden die Fortschritte der vergangenen Woche besprochen sowie die Aufgaben für die kommende Woche an die Teammitglieder verteilt. Zudem bieten die Sitzungen die Möglichkeit, offene Fragen zu klären und Unklarheiten im Projektverlauf zu beseitigen.

Ein Protokollant dokumentiert die Besprechungsergebnisse in einem Protokoll, das nach Abschluss an alle Teammitglieder, den Kunden und den Betreuer weitergeleitet wird.

### 5.4 Definition of Done

Eine Aufgabe gilt als abgeschlossen, bzw. ein Feature gilt als implementiert, wenn das zu implementierende Feature die folgenden Kriterien erfüllt:

1. Der hinzugefügte Code hat alle lokalen Tests erfolgreich durchlaufen.
2. Codequalität mit Hilfe von SonarQube geprüft

3. Der Pull-Request wurde von einem Teammitglied durch ein Peer-Review überprüft und freigegeben
4. Die Dokumentation wurde um das implementierte Feature ergänzt

### 5.5 Kommunikation und Dokumente

Die interne Kommunikation und Abstimmung erfolgen primär über den aufgesetzten Discord-Server. Im Channel „links-und-dokumente“ werden alle relevanten Dokumente, Protokolle und weiterführenden Links zum Projekt zentral bereitgestellt.

Meetings finden über Webex statt und werden mithilfe des integrierten Einladungstools geplant. Zur Nachverfolgbarkeit werden alle Besprechungen in einem Protokoll dokumentiert. Die Erstellung der Protokolle erfolgt durch den Protokollführer in HedgeDoc.

Die Projektdokumentation wird in Microsoft Word bzw. Word Online verfasst.

### 5.6 Lizenz

Unser Produkt möchten wir als Open Source bereitstellen, da wir der Hochschule Esslingen und ihren Anwendern die Freiheit geben wollen, das Produkt nach ihren Bedürfnissen zu nutzen und anzupassen. Eine Closed-Source Lizenz könnte den Einsatz des Produkts beim Kunden einschränken. Zudem möchten wir sicherstellen, dass zukünftige Werkstudenten und Administratoren in der Lage sind, das Produkt weiterzuentwickeln.

## 6 Schnittstellen

Im Rahmen der Entwicklung unserer Schnittstellen haben wir uns bewusst für ein REST-basiertes API-Design entschieden. Ziel war es, eine konsistente, leicht verständliche und wartbare Schnittstellenstruktur zu schaffen, die sowohl interne als auch externe Entwickler effizient nutzen können.

Zur Sicherstellung eines hohen Qualitätsstandards und zur Förderung bewährter Best Practices haben wir uns an den Zalando RESTful API Guidelines orientiert. Diese bieten eine umfassende Sammlung von Empfehlungen zur Strukturierung, Benennung, Versionierung und Dokumentation von RESTful APIs.

Wichtige Prinzipien, die wir umgesetzt haben, umfassen unter anderem:

- Konsistente Ressourcennamen gemäß REST-Konventionen (Nomen im Plural, z.B. /users, /orders)
- Klare Trennung von Ressourcen und Aktionen mittels HTTP-Methoden (GET, POST, PUT, DELETE)
- Explizite Versionierung der API über den URL-Pfad (z.B. /v1/users)
- Verwendung standardisierter HTTP-Statuscodes zur eindeutigen Kommunikation von Erfolgs- und Fehlermeldungen
- Einheitliche Fehlerstruktur (Problem+JSON Format) zur besseren Nachvollziehbarkeit und Debugging

Durch die Orientierung an diesen Richtlinien erhöhen wir nicht nur die technische Qualität unserer Schnittstellen, sondern stärken auch die langfristige Wartbarkeit und Erweiterbarkeit unseres Systems.

Zur Dokumentation nutzen wir Swagger. Swagger ist ein Framework für die Spezifikation, Visualisierung und Interaktion mit RESTful APIs. Es basiert auf der OpenAPI-Spezifikation, die eine standardisierte, maschinen- und menschenlesbare Beschreibung von HTTP-Schnittstellen ermöglicht.

Swagger erleichtert uns die Erstellung, Pflege und Verständlichkeit der API-Dokumentation, indem es sowohl Entwicklern als auch externen Nutzern einen klaren Überblick über verfügbare Endpunkte, Parameter, Rückgabewerte und mögliche Fehler liefert.

Zudem bietet es mit Swagger UI eine interaktive, grafische Benutzeroberfläche, über die die API direkt im Browser getestet werden kann.

Unsere aktuelle, kontinuierlich aktualisierte API-Dokumentation ist unter der Domain [http:](http://)

## 6 SCHNITTSTELLEN

//swagger.test.insy.hs-esslingen.com/ verfügbar. Als Beispiel ist hier in Abbildung 6.1 ein POST-Request zu sehen, das an insy-hs-esslingen.de/inventories geschickt wird.

The screenshot shows the Swagger UI interface for a POST request to the '/inventories' endpoint. The endpoint is described as 'Erstellt einen Gegenstand in der Inventarisierungsliste'. The description is 'Neues Gerät/Software inventarisiern.' and it specifies 'No parameters'. The 'Request body' is marked as 'required' and has a 'application/json' dropdown. The example value for the JSON body is:

```
{
  "cost_center": "7234583202",
  "inventories_id": 1,
  "inventories_description": "Asus Gaming Laptop",
  "company": "Gedankenfabrik AG",
  "inventories_price": "1299,99",
  "inventories_serial_number": "834232T32H",
  "inventories_location": "F99.234",
  "orderer": "Peter",
  "tags": [
    {
      "1,
      "3
    }
  ]
}
```

The 'Responses' section includes three entries:

- 201**: 'Gerät wurde inventarisiert' with a 'application/json' dropdown and a note 'No links'. Example value and schema are shown.
- 400**: 'Error 400: Bad Request' with a 'application/json' dropdown and a note 'No links'. Example value and schema are shown.
- 500**: 'Innterer Fehler beim Inventarisiern' with a 'application/json' dropdown and a note 'No links'. Example value and schema are shown.

Abbildung 6.1: Dokumentationsbeispiel zu einem POST-Request an /inventories *Quelle: erstellt mit SwaggerUI[<empty citation>]*

## 7 Aufwandsschätzung

Für die verfügbare Zeit für unser Projekt orientieren wir uns an dem Arbeitsaufwand für einen ECTS laut der SPO, welcher näherungsweise 30 Stunden entspricht. Bei 10 Credits für das Projekt ergibt das 300 Stunden, da wir zu sechs sind sind also insgesamt 1800 Stunden.

### 7.1 Geschätzter Aufwand

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
Erstes Meeting mit Betreuer und Kunde + Protokoll	3.5h*6P
Regelmäßige Meetings mit Betreuer (11 Meetings)	172h
<ul style="list-style-type: none"> <li>• bereits Vergangen</li> <li>• geplant</li> </ul>	<ul style="list-style-type: none"> <li>• 2,5h*6P*11W</li> <li>• 1,5*6P</li> </ul>
Seminar 1: Teambildung und Konfliktlösung	42h <ul style="list-style-type: none"> <li>• 7h*6P</li> </ul>
UI-Mockups	41h
Bereits vergangen:	
<ul style="list-style-type: none"> <li>• Basiskomponenten, Inventarliste, Filtermenü, Kontrollansicht</li> <li>• Inventarisierungs-Wizard</li> <li>• Dokumentation</li> <li>• Statistiken</li> </ul>	<ul style="list-style-type: none"> <li>• 8,5h</li> <li>• 4h</li> <li>• 3h</li> <li>• 2h</li> </ul>
Geplant:	
<ul style="list-style-type: none"> <li>• Detailansicht</li> <li>• Wizard: Artikelerweiterung</li> <li>• Administrationsmenü</li> <li>• Überarbeitung Kontrollansicht</li> <li>• Dashboard/Homepage</li> <li>• Zusätzliche Features</li> <li>• Finale Überarbeitung</li> <li>• Dokumentation</li> </ul>	<ul style="list-style-type: none"> <li>• 2h</li> <li>• 2h</li> <li>• 1.5h</li> <li>• 2h</li> <li>• 3h</li> <li>• ca. 4h</li> <li>• 4h</li> <li>• 5h</li> </ul>
VM's aufsetzen	16h <ul style="list-style-type: none"> <li>• 4h*4P</li> </ul>
Team interne Meetings	210h
<ul style="list-style-type: none"> <li>• Bisher vergangen</li> <li>• Veranschlagt</li> </ul>	<ul style="list-style-type: none"> <li>• 2h*6P</li> <li>• 3h*6P*11 Wochen</li> </ul>
Seminar 2: Präsentation und Disputation	48h <ul style="list-style-type: none"> <li>• 8h*6P</li> </ul>
Dokumentation Meilenstein 1	ca. 52.5h
<ul style="list-style-type: none"> <li>• Einleitung</li> <li>• Personas</li> <li>• Funktionsumfang</li> <li>• Architektur</li> <li>• Projektmanagement</li> <li>• Aufwandsschätzung</li> <li>• Aufwandsschätzung überarbeiten</li> </ul>	<ul style="list-style-type: none"> <li>• 2h</li> <li>• 2h*2P?</li> <li>• 3h</li> <li>• 6h</li> <li>• 1.5h?</li> <li>• 3h*6P Teil 1</li> <li>• 18h</li> </ul>
Vorbereitung Präsentationen	50h

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
<ul style="list-style-type: none"> <li>• Arbeitszeit einzeln</li> <li>• Gemeinsam</li> <li>• Finales Überarbeiten</li> </ul>	<ul style="list-style-type: none"> <li>• 2*2h*6P</li> <li>• 2*2h*6P</li> <li>• 2h</li> </ul>
Tag der Präsentationen	48h
	<ul style="list-style-type: none"> <li>• 2*4h?*6P</li> </ul>
Dokumentation Meilenstein 2	24h
<ul style="list-style-type: none"> <li>• User Stories pro Feature</li> <li>• Technisches Konzept</li> </ul>	<ul style="list-style-type: none"> <li>• 2h</li> </ul>
<ul style="list-style-type: none"> <li>• Logische Schichten           <ul style="list-style-type: none"> <li>• Struktursicht</li> <li>• Verteilungssicht</li> <li>• Verhaltenssicht</li> </ul> </li> <li>• Verwendete Technologie/Frameworks</li> <li>• Schnittstellentechnologie</li> </ul>	<ul style="list-style-type: none"> <li>22h</li> <li>3h</li> <li> <ul style="list-style-type: none"> <li>• 1h</li> <li>• 1h</li> <li>• 1h</li> </ul> </li> <li>• 2h</li> <li>• 2h</li> </ul>
<ul style="list-style-type: none"> <li>• Datenmodell           <ul style="list-style-type: none"> <li>• Logisch</li> <li>• Physisch</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>15h</li> <li> <ul style="list-style-type: none"> <li>• 5h</li> <li>• 10h</li> </ul> </li> </ul>
Meilenstein 3	28h
<ul style="list-style-type: none"> <li>• Code Reviews</li> <li>• Technischen Prototyp vorbereiten</li> </ul>	<ul style="list-style-type: none"> <li>• 3h*6P</li> <li>• 10h</li> </ul>
Meilenstein X	69h
<ul style="list-style-type: none"> <li>• Installations- und Administrationshandbuch</li> <li>• Aufteilung des Teams: wer hat was gemacht (mit Namen der Teilnehmer)?</li> <li>• Reflektion Projektmanagement</li> <li>• Reflektion Lernfortschritt</li> <li>• Ausblick: Was könnte an Ihrem Projekt ergänzt werden?</li> </ul>	<ul style="list-style-type: none"> <li>• 18h</li> <li>• 3h*6P</li> <li>• 16h</li> <li>• 12h</li> <li>• 5h</li> </ul>
Initialisieren Repositories	7h
Benutzerverwaltung & Authentifizierung / Keycloak aufsetzen	109h
<ul style="list-style-type: none"> <li>• Frontend Design</li> <li>• Backend Einbindung</li> <li>• Kommunikation mit Keycloak-Server</li> <li>• Login/Registrierung mit keycloak</li> <li>• Sichere Speicherung von Benutzerdaten</li> <li>• Inter-App-Authentifizierung (BeSy-InSy)</li> <li>• Troubleshooting</li> </ul>	<ul style="list-style-type: none"> <li>• 5h</li> <li>• 18h</li> <li>• 12h</li> <li>• 10h</li> <li>• 8h</li> <li>• 12h</li> <li>• 18h</li> </ul>

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
<ul style="list-style-type: none"> <li>• Code Review</li> <li>• Testing (Frontend und Backend)</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 6h</li> <li>• 16h</li> <li>• 4h</li> </ul>
(Neue) Technologien: Einarbeiten & Einlesen	108h
Angular <ul style="list-style-type: none"> <li>• Einzeln</li> <li>• Besprechung im Team</li> </ul>	24h <ul style="list-style-type: none"> <li>• 6h*ca.2P</li> <li>• 2h*6P</li> </ul>
Spring Boot <ul style="list-style-type: none"> <li>• Einzeln</li> <li>• Besprechung im Team</li> </ul>	30h <ul style="list-style-type: none"> <li>• 6h*ca.3P</li> <li>• 2h*6P</li> </ul>
Nginx <ul style="list-style-type: none"> <li>• Einzeln</li> <li>• Besprechung im Team</li> </ul>	10h <ul style="list-style-type: none"> <li>• 4h*ca.1P</li> <li>• 1h*6P</li> </ul>
Keycloak <ul style="list-style-type: none"> <li>• Einzeln</li> <li>• Besprechung im Team</li> </ul>	28h <ul style="list-style-type: none"> <li>• 8h*ca.2P</li> <li>• 2h*6P</li> </ul>
Postgres <ul style="list-style-type: none"> <li>• Einzeln</li> <li>• Besprechung im Team</li> </ul>	16h <ul style="list-style-type: none"> <li>• 5h*ca.2P</li> <li>• 1h*6P</li> </ul>
CI/CD Pipeline	51h
<ul style="list-style-type: none"> <li>• Einlesen</li> <li>• Einrichten</li> <li>• Docker Setup &amp; Troubleshooting</li> <li>• Sonarqube</li> <li>• Github Docker Registry</li> <li>• Github Actions</li> </ul>	<ul style="list-style-type: none"> <li>• 14h</li> <li>• 8h</li> <li>• 16h</li> <li>• 6h</li> <li>• 4h</li> <li>• 3h</li> </ul>
Inventarliste	52h
<ul style="list-style-type: none"> <li>• Frontend Design &amp; Logik</li> <li>• Backend Einbindung</li> <li>• Testing (Front + Back End)</li> <li>• Code Review</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 18h</li> <li>• 16h</li> <li>• 10h</li> <li>• 4h</li> <li>• 4h</li> </ul>
Inventarisierungs Wizard	59h
<ul style="list-style-type: none"> <li>• Frontend Design &amp; Logik</li> </ul>	<ul style="list-style-type: none"> <li>• 12h</li> </ul>

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
<ul style="list-style-type: none"> <li>• In Webserverstruktur einpflegen</li> <li>• Evtl. Kommunikation mit BeSy</li> <li>• Backend Einbindung</li> <li>• Frontend Tests</li> <li>• Backend Testing</li> <li>• Code Review</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 4h</li> <li>• 6h</li> <li>• 16h</li> <li>• 6h</li> <li>• 8h</li> <li>• 3h</li> <li>• 4h</li> </ul>
Inventarisieren	196h
Bestellübersicht	52h
<ul style="list-style-type: none"> <li>• API für BeSy</li> <li>• Frontend Design</li> <li>• Backend Logik</li> <li>• Testing</li> <li>• Code Review</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 6h</li> <li>• 12h</li> <li>• 16h</li> <li>• 12h</li> <li>• 4h</li> <li>• 2h</li> </ul>
Artikelübersicht	47h
<ul style="list-style-type: none"> <li>• Frontend Design</li> <li>• Backend Logik</li> <li>• Testing</li> <li>• Code Review</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 10h</li> <li>• 16h</li> <li>• 12h</li> <li>• 5h</li> <li>• 4h</li> </ul>
Artikel kombinieren	30h
<ul style="list-style-type: none"> <li>• Frontend Design &amp; Logik</li> <li>• Testing</li> <li>• Code Review</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 16h</li> <li>• 6h</li> <li>• 4h</li> <li>• 4h</li> </ul>
Manuelles Inventarisieren	29h
<ul style="list-style-type: none"> <li>• Frontend Design</li> <li>• Backend Logik</li> <li>• Testing</li> <li>• Code Review</li> <li>• Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 6h</li> <li>• 8h</li> <li>• 8h</li> <li>• 4h</li> <li>• 3h</li> </ul>
Direkt/Kontrollansicht	38h
<ul style="list-style-type: none"> <li>• Frontend Design</li> <li>• Backend Logik</li> <li>• Testing</li> <li>• Code Review</li> </ul>	<ul style="list-style-type: none"> <li>• 10h</li> <li>• 12h</li> <li>• 9h</li> <li>• 4h</li> </ul>

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
• Doku	• 3h
De-Inventarisieren	27h
• Frontend Design & Logik • Backend Einbindung • Testing (Front + Back End) • Code Review • Doku	• 8h • 6h • 10h • 1h • 2h
Erweiterung bestehender Gegenstände	53h
• Frontend Design & Logik • Backend Einbindung • Testing (Front + Back End) • Code Review • Doku	• 16h • 18h • 12h • 4h • 3h
Anbindung an Bestellprozess BeSy	41h
• Frontend Design & Logik • Backend Einbindung • Testing (Front + Back End) • Code Review • Doku	• 1h • 19h • 14h • 4h • 3h
Such- und Filterfunktionen	55h
• Frontend Design & Logik • Backend Einbindung • Testing (Front + Back End) • Code Review • Doku	• 16h • 18h • 14h • 4h • 3h
Historienverwaltung	48h
• Frontend Design & Logik • Backend Einbindung • Testing (Front + Back End) • Code Review • Doku	• 10h • 18h • 10h • 3h • 3h
Datenimport und –export	37h

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
<ul style="list-style-type: none"> <li>Frontend Design &amp; Logik</li> <li>Backend Einbindung</li> <li>Testing (Front + Back End)</li> <li>Code Review</li> <li>Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 8h</li> <li>• 16h</li> <li>• 8h</li> <li>• 3h</li> <li>• 2h</li> </ul>
Statistiken	36h
<ul style="list-style-type: none"> <li>Frontend Design &amp; Logik</li> <li>Backend Einbindung</li> <li>Testing (Front + Back End)</li> <li>Code Review</li> <li>Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 14h</li> <li>• 8h</li> <li>• 8h</li> <li>• 3h</li> <li>• 3h</li> </ul>
Repository pflegen	18h
API-Design	59h
<ul style="list-style-type: none"> <li>Festlegung der API-Ziele</li> <li>Endpunkt Definition</li> <li>Sicherheitskonzept</li> <li>Fehlerbehandlung</li> <li>Testing</li> <li>Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 6h</li> <li>• 8h*ca.4P</li> <li>• 5h</li> <li>• 6h</li> <li>• 6h</li> <li>• 4h</li> </ul>
Testprotokoll für das Frontend	9h
<ul style="list-style-type: none"> <li>Erste Entwürfe &amp; Planung</li> <li>Im Team besprechen</li> <li>Finalisieren</li> <li>Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 4h</li> <li>• 1h*ca.3P</li> <li>• 1h</li> <li>• 1h</li> </ul>
Datenbank aufsetzen	10h
<ul style="list-style-type: none"> <li>Einarbeiten</li> <li>Installieren</li> <li>Datenmodel implementieren</li> </ul>	<ul style="list-style-type: none"> <li>• 3h</li> <li>• 3h</li> <li>• 4h</li> </ul>
Clean Code Definition	14h
<ul style="list-style-type: none"> <li>Erste Entwürfe &amp; Planung</li> <li>Im Team besprechen</li> <li>Finalisieren</li> <li>Doku</li> </ul>	<ul style="list-style-type: none"> <li>• 1h*6P</li> <li>• 1h*6P</li> <li>• 1h</li> <li>• 1h</li> </ul>

## 7 AUFWANDSSCHÄTZUNG

---

Aufgabe	Zeit
Datenmodellierung	19h
<ul style="list-style-type: none"><li>• Erste Entwürfe</li><li>• Besprechungen</li><li>• Überarbeitung</li><li>• Finalisieren</li><li>• Doku</li></ul>	<ul style="list-style-type: none"><li>• 2h*2P</li><li>• 1h*4P</li><li>• 8h</li><li>• 1h</li><li>• 2h</li></ul>
Verwendete Lizenzen	3h
Summe	1.832,5h

Tabelle 7.1: Ursprüngliche Aufwandsschätzung