

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,
Danny Holzinger, Akim Kausch, Valentin Mitrev**

am 29. April 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 29. April 2025

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis	I
Tabellenverzeichnis	II
Quellcodeverzeichnis	III
Abkürzungsverzeichnis	IV
1 Einleitung	1
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
2 Gefordeter Funktionsumfang	6
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
3 UI Entwürfe	9
3.1 Inventarliste	9
3.1.1 Filtermenü	10
3.2 Detailansicht eines Gegenstandes	10

3.3	Inventarisierung	11
3.3.1	Bestellübersicht	11
3.3.2	Artikelübersicht	11
3.3.3	Inventarisierungswizard	12
4	Architektur	14
4.1	Technologische & konzeptionelle Architektur	14
4.1.1	Strukturschicht	15
4.2	Deployment Prozess	16
4.3	Frontend	16
4.4	Backend	17
4.5	Datenbank	18
5	Projektmanagement	20
5.1	Entwicklungsprozess und Methodik	20
5.2	Versionsverwaltung	20
5.3	Meetings	20
5.4	Definition of Done	20
5.5	Kommunikation und Dokumente	21
5.6	Lizenz	21
6	Schnittstellen	22
7	Aufwandsschätzung	24
7.1	Geschätzter Aufwand	24

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 Neue Inventarisierung	11
3.4	Prototyp Bestellübersicht	12
3.5	Prototyp Artikelübersicht	13
4.1	Three-Tier Architecture	14
4.2	Strukturschicht	15
4.3	CI/CD Pipeline	16
4.4	Datenbankmodell	19
6.1	Dokumentationsbeispiel zu einem POST-Request an /inventories	23

Tabellenverzeichnis

7.1 Aufwandsschätzung	31
---------------------------------	----

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 Benutzerzugänglichkeit, 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, 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

Filter	
Kostenstellen	000000 000001 000002 000003 000004
Inventarnummer	ab: 003817 bis: 004800 ab: 010040
ab: [Startwert] <input type="button" value="Apply"/>	
bis: [Höchstwert] <input type="button" value="Apply"/>	
Zeitraum	bis: 03.01.2024 ab: 11.07.2024 bis: 24.11.2024
ab: [Startdatum] <input type="button" value="Apply"/>	
bis: [Enddatum] <input type="button" value="Apply"/>	
Firma	<input type="button" value="Firma A"/> <input type="button" value="Firma B"/> <input type="button" value="Firma C"/>
Preis	bis: 300€
ab: [Startwert] <input type="button" value="Apply"/>	
bis: [Höchstwert] <input type="button" value="Apply"/>	
Seriennummer	ab: XX30DE8
ab: [Startwert] <input type="button" value="Apply"/>	
bis: [Höchstwert] <input type="button" value="Apply"/>	
Standort / Nutzer	<input type="button" value="Prof. Test"/> <input type="button" value="Prof. Tmp"/> <input type="button" value="Prof. X F1.304"/> <input type="button" value="Prof. Z"/>
Bestellt von	<input type="button" value="Test"/> <input type="button" value="Tmp"/> <input checked="" type="button" value="X"/>
Tags	000000 000001 000002 000003 000004

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 wie eine ausgeklappte Version des Eintrags aussehen und die darunterliegenden Zeilen dynamisch nach unten verschieben. Hier sollen zusätzliche Information angezeigt werden, für die in der Haupttabelle kein Platz ist. Über die Detailansicht soll ein Einblick in die Historie, die Änderungen, die ein Gegenstand bereits durchlaufen hat, möglich sein. Außerdem zusätzliche Informationen, Brutto/Nettopreis, die benutzerdefinierten Tags und die Notizen.

3.3 Inventarisierung

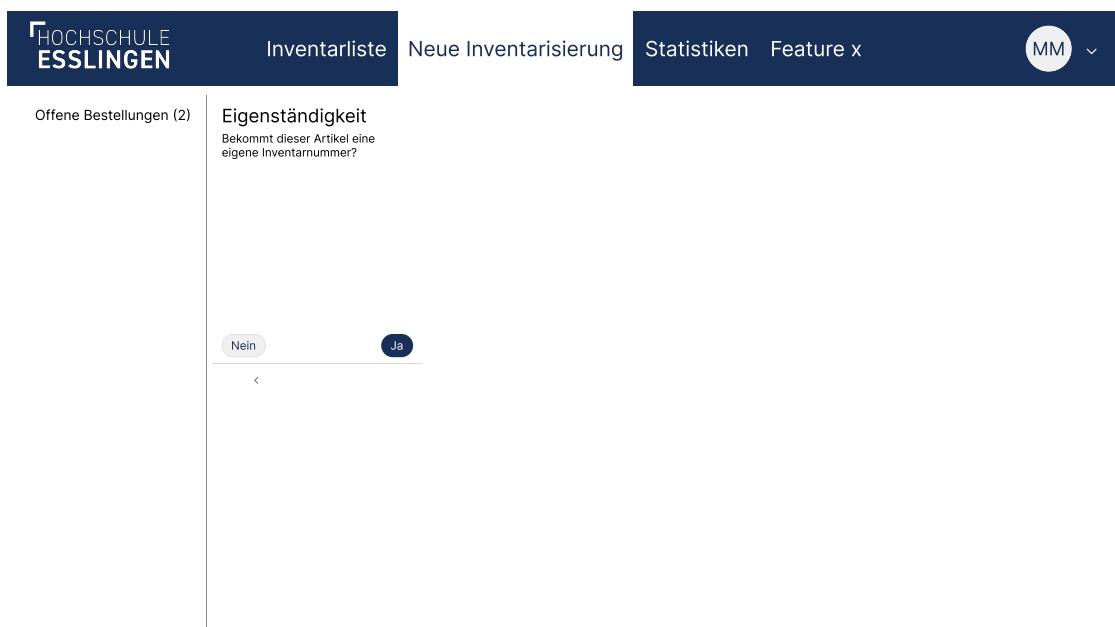


Abbildung 3.3: Prototyp Neue Inventarisierung *Quelle: Erstellt in Figma [<empty citation>]*

Es sind mehrere Möglichkeiten geplant, Gegenstände zu inventarisieren. Zunächst ist auf der Inventarisierungsstartseite der Inventarisierungs-Wizard zu sehen, auf den in Punkt 4.3.3 noch genauer eingegangen wird. Auf der linken Seite befindet sich ein Untermenü, das Platz für verschiedene kontextabhängige und kontextübergreifende Buttons bietet. In Abbildung 3.3 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.

3.3.1 Bestellübersicht

Über den Button “Offene Bestellungen” wird die in Abbildung 3.4 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 abgearbeitet werden. Bestellungen, die Einzelteile enthalten, können für das Zusammenfassen in einen oder mehrere Gesamtgegenstände ausgewählt werden und in der Artikelübersicht aggregiert werden.

3.3.2 Artikelübersicht

In der in Abbildung 3.5 dargestellten Artikelübersicht werden zuvor ausgewählte Bestellungen und deren Artikel angezeigt. Hier besteht die Möglichkeit einzelne Artikel auszuwählen und zu

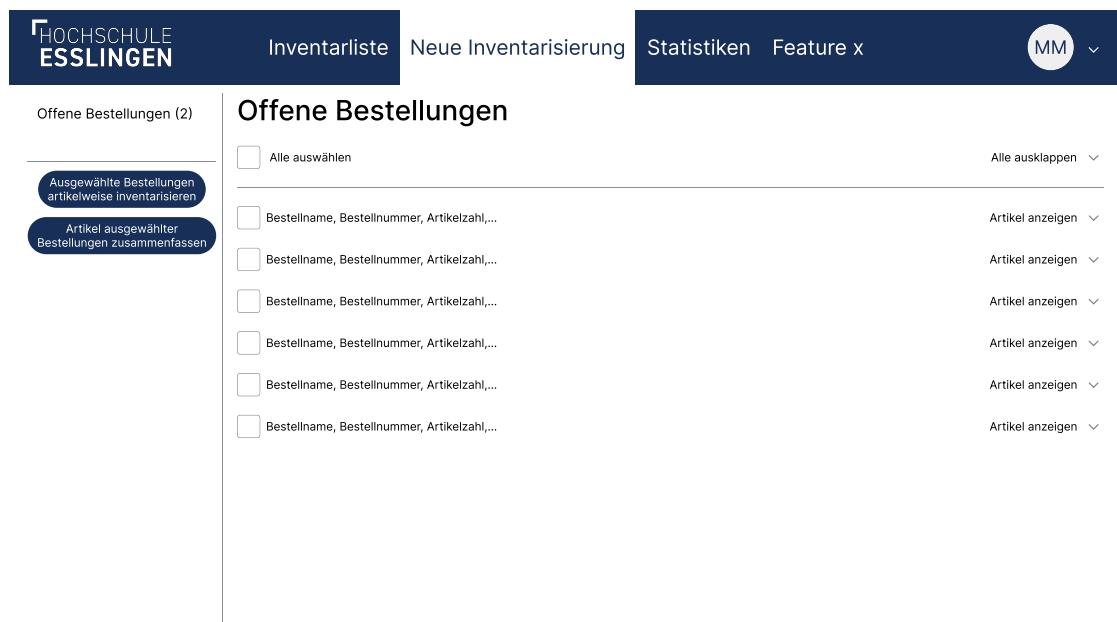


Abbildung 3.4: Prototyp Bestellübersicht *Quelle: Erstellt in Figma [<empty citation>]*

einem oder mehreren Gesamtgegenständen zusammenzufassen. Wenn möglich sollen diesen 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. Bei aggregierten Gegenständen werden automatisch Notizen generiert, in denen die einzelnen Komponenten und deren Bestellungsinformationen aufgeführt werden.

3.3.3 Inventarisierungswizard

In Abbildung ?? 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

3 UI ENTWÜRFE

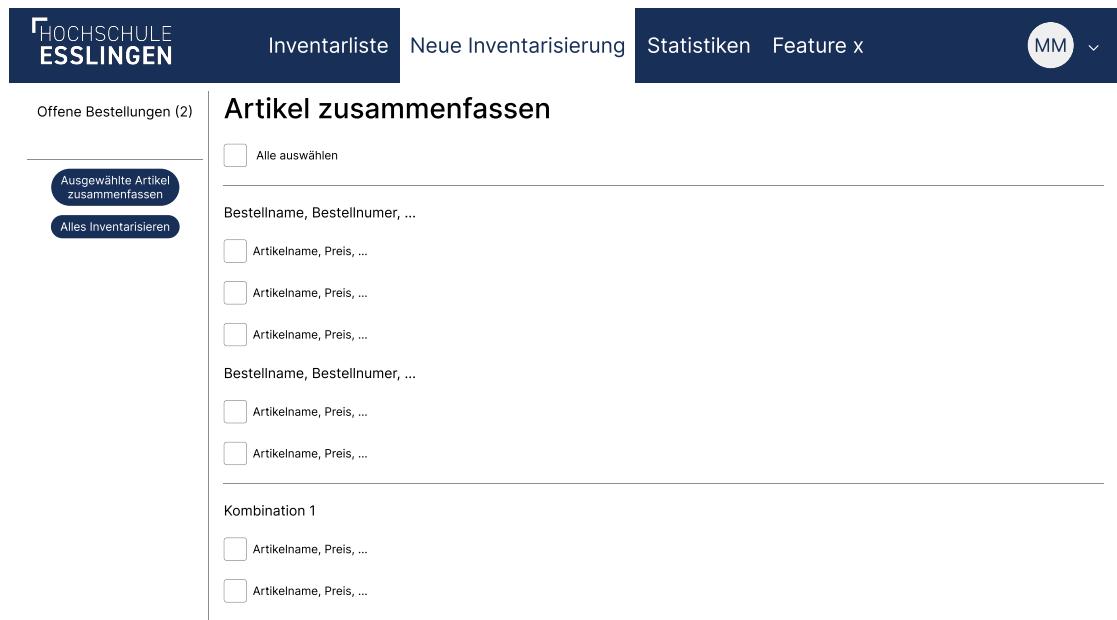


Abbildung 3.5: Prototyp Artikelübersicht *Quelle: Erstellt in Figma [<empty citation>]*

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.

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.

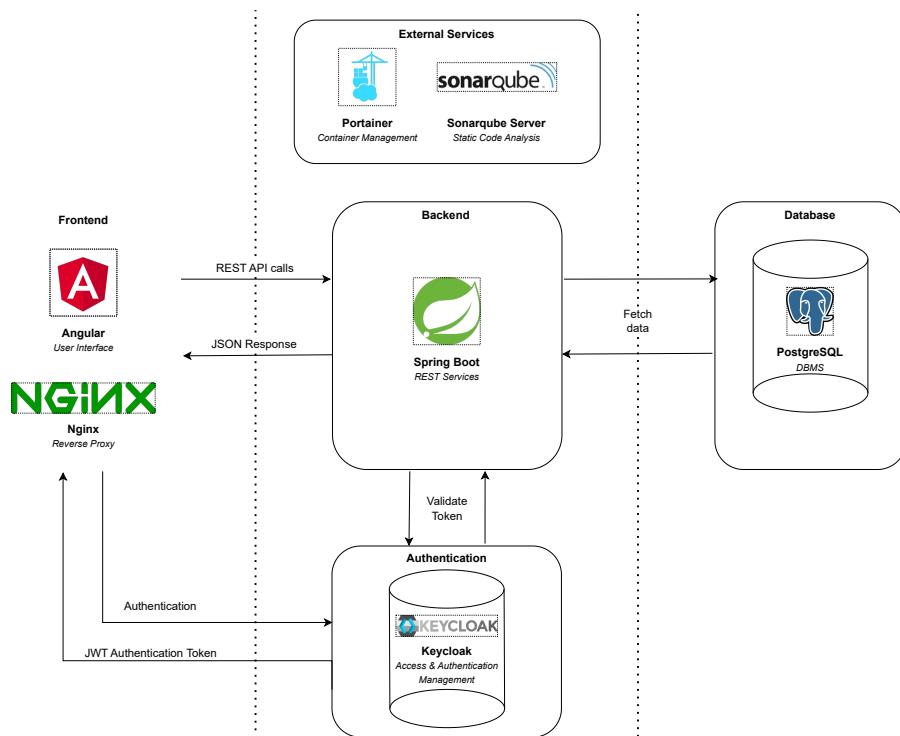


Abbildung 4.1: Three-Tier Architecture *Quelle: in Anlehnung [<empty citation>]*

4.1.1 Strukturschicht

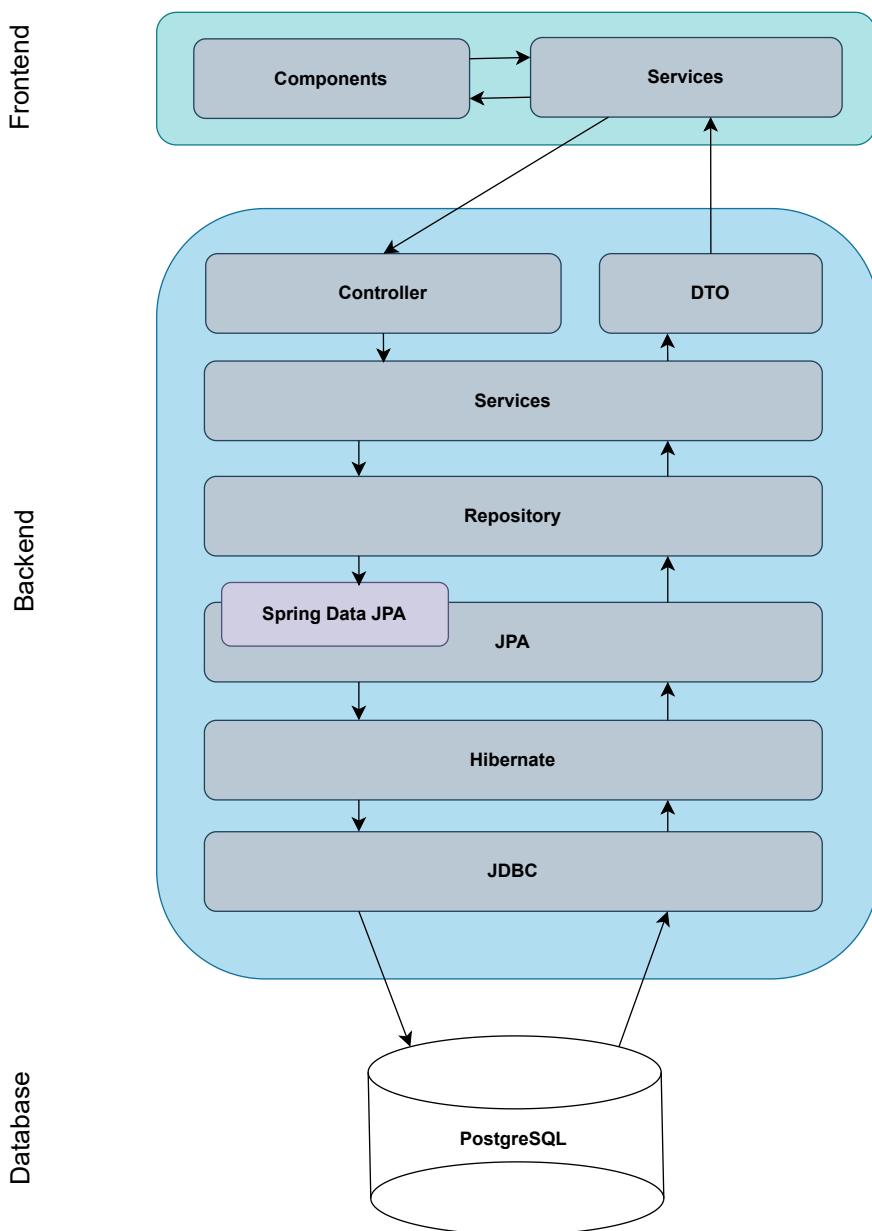


Abbildung 4.2: Strukturschicht *Quelle: in Anlehnung [<empty citation>]*

4.2 Deployment Prozess

Mit einer CI/CD-Pipeline soll die Integration und das Testen von Software im bestehenden System vereinfacht werden. Über das GitHub-Repository, in das die Entwickler ihre Codeänderungen pushen, wird ein GitHub-Workflow angestoßen. Dieser Workflow baut automatisch Docker-Container, führt Backend-Tests durch und führt statische Code-Analysen mithilfe von SonarQube aus. Danach werden die fertigen Docker-Images in die GitHub Container Registry gepusht, von wo sie entweder manuell oder automatisiert auf die virtuellen Maschinen (VMs) gezogen werden können, um dort ausgeführt zu werden. Die statischen Code-Analysen sollen anschließend über einen SonarQube-Server auf der VM zugänglich gemacht werden und eine hohe Code-Qualität gewährleisten.

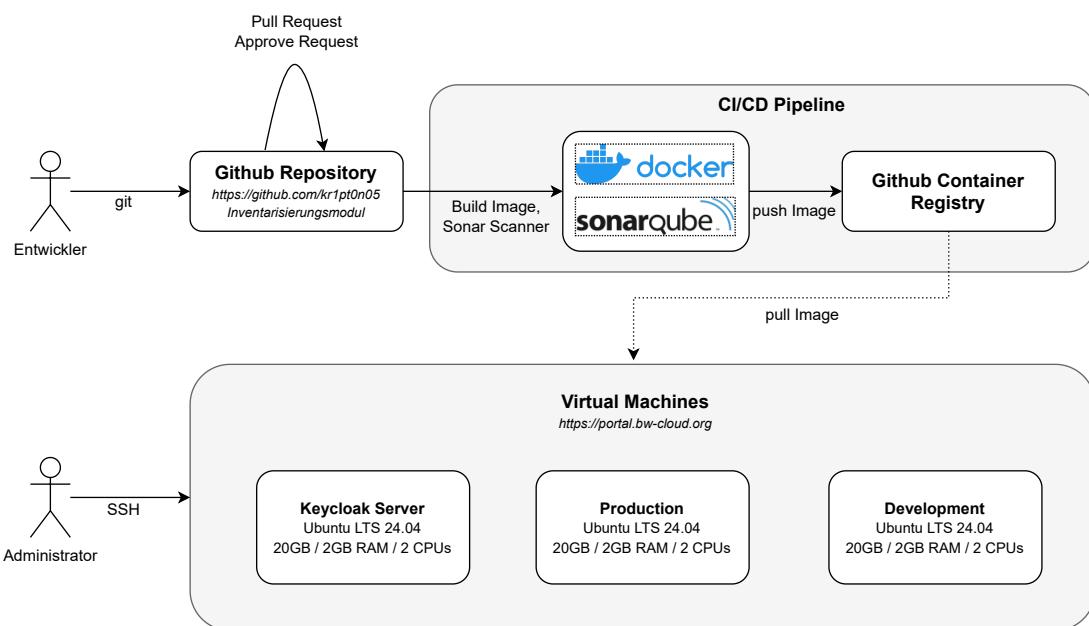


Abbildung 4.3: CI/CD Pipeline *Quelle: in Anlehnung [<empty citation>]*

4.3 Frontend

Das Frontend wird mit dem weit verbreiteten TypeScript-Framework Angular entwickelt. Es bietet eine umfassende Sammlung an Funktionen wie Routing, Formulare, HTTP-Client und State Management, die den Entwicklungsprozess durch seine vorgegebene und effektive Struktur erheblich vereinfachen. Der Einsatz von TypeScript zum Einsatz, das durch seine strikte Typisierung im Vergleich zu JavaScript Vorteile in Bezug auf Wartbarkeit und Lesbarkeit bietet. Angular ist besonders geschickt, da es bereits viele verschiedene Anwendungsfälle abdeckt und mit seiner vielseitigen Bibliothek zahlreiche Funktionen bereitstellt. Zudem lässt es sich problemlos mit weiteren Abhängigkeiten erweitern. Der Einsatz von Komponenten vereinfacht den Entwicklungsprozess und fördert die Wiederverwendbarkeit von Code durch seine hohe Modularität. Das Bauen von dynamischen Webseiten ist erheblich einfacher.

Da auch die Bestellsoftware BeSy in Angular entwickelt wird, fügt sich diese Technologie besonders gut in die bestehende Technologielandschaft ein.

Um Frontend und Backend voneinander zu trennen und eine bessere Skalierbarkeit zu gewährleisten, wird die Angular-Applikation über einen Reverse Proxy bereitgestellt. Die Wahl fiel hier auf Nginx, da es ebenfalls weit verbreitet und einfach zu konfigurieren ist. Nginx ist ein äußerst leistungsfähiger Webserver, der unter anderem als Load Balancer, zum Caching oder für Sicherheitskonfigurationen eingesetzt werden kann. In diesem Projekt konzentrieren wir uns jedoch auf die Hauptfunktionalität – das Bereitstellen von statischen Seiten –, da die weiteren Funktionen im aktuellen Rahmen des Projekts vernachlässigt werden können.

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 (Java Persistence API). JPA ist eine Spezifikation für das Persistieren von Java-Objekten in einer relationalen Datenbank. Eine weit verbreitete Implementierung von JPA ist Hibernate. Darüber hinaus bietet JPA die eigene Abfragesprache Java Persistence Query Language (JPQL) an, die mit Java-Klassen arbeitet, statt direkt auf Datenbanktabellen zuzugreifen. So würde man beispielsweise statt `SSELECT * FROM users` die Abfrage `SSELECT u FROM User u` verwenden. JPA schließt den Einsatz von SQL jedoch nicht aus,

sondern bietet vielmehr eine zusätzliche Möglichkeit der Abfrage.

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.

4 ARCHITEKTUR

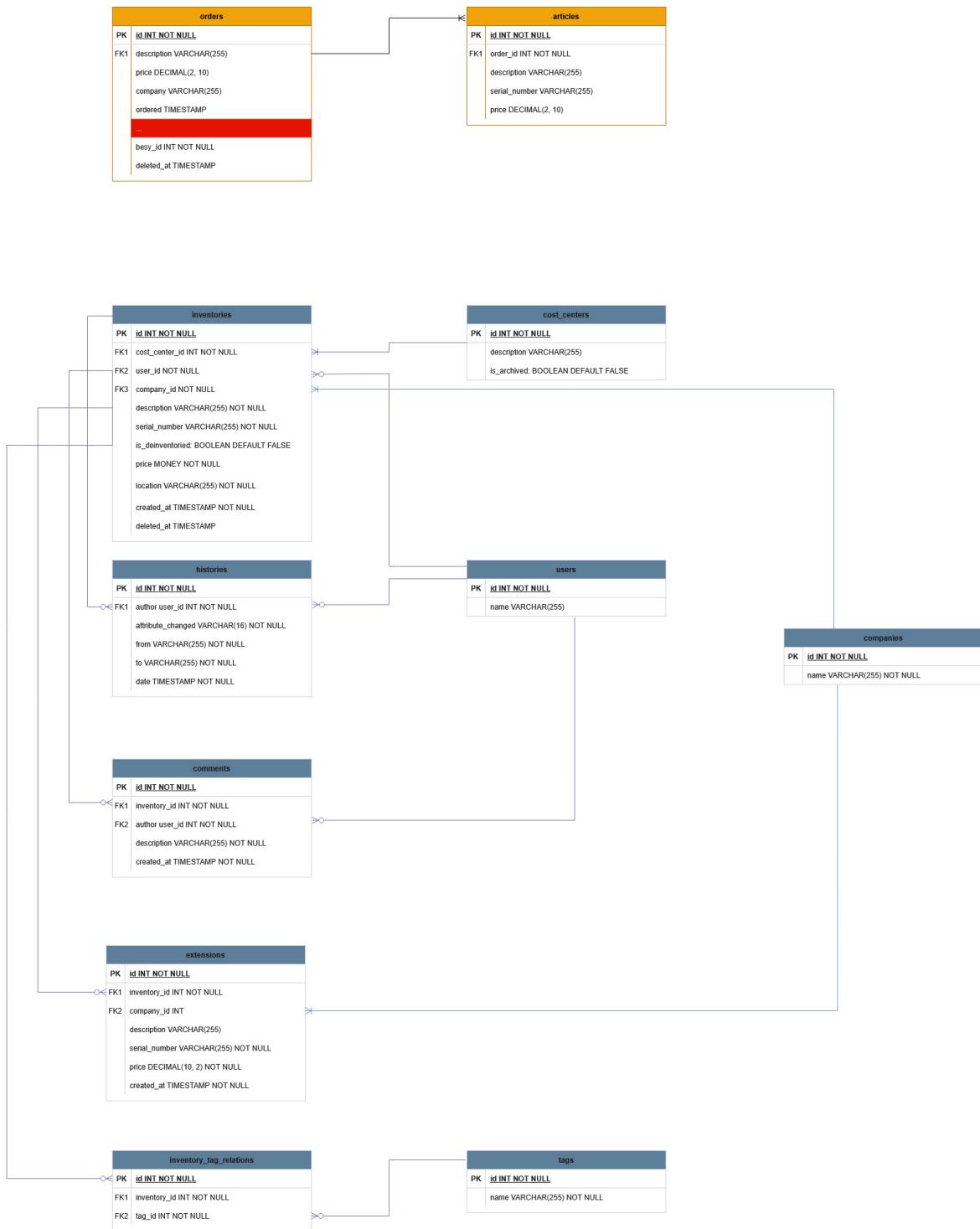


Abbildung 4.4: Datenbankmodell Quelle: in Anlehnung [] /

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 top bar indicates the method (POST) and the endpoint (/inventories). A brief description follows: 'Erstellt einen Gegenstand in der Inventarisierungsliste'. Below this, a note says 'Neues Gerät/Software inventarisiern.' Under the 'Parameters' section, it states 'No parameters'. The 'Request body' section is marked as 'required' and has a dropdown set to 'application/json'. A note below says 'Neues Gerät inventarisieren. Ausgewählte Tags werden beim Erstellen in einem JSON-Array übergeben. Falls keine Tags ausgewählt wurden wird ein leeres JSON-Array übergeben.' An example value is provided:

```
{
  "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 lists three entries:

- 201**: 'Gerät wurde inventarisiert' (Media type: application/json). Example value: same as the request body above.
- 400**: 'Error 400: Bad Request' (Media type: application/json). Example value: { "code": 400, "message": "Bad Request" }
- 500**: 'Innterer Fehler beim Inventarisiern' (Media type: application/json). Example value: { "code": 500, "message": "Internal Server Error" }

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
- bereits Vergangen	- 2,5h*6P*11W - 1,5*6P
Seminar 1: Teambildung und Konfliktlösung	42h - 7h*6P
UI-Mockups	39h+?h
Bereits vergangen:	
- Basiskomponenten, Inventarliste, Filtermenü, Kontrollansicht	- 8,5h
- Inventarisierungs-Wizard	
- Dokumentation	- 4h
- Statistiken	- 3h
Geplant:	- ?h
- Detailansicht	
- Wizard: Artikelerweiterung	- 2h
- Administrationsmenü	- 2h
- Überarbeitung Kontrollansicht	- 1.5h
- Dashboard/Homepage	- 2h
- Zusätzliche Features	- 3h
- Finale Überarbeitung	- ca. 4h
- Dokumentation	- 4h
	- 5h
VM's aufsetzen	16h - 4h*4P
Team interne Meetings	210h
- Bisher vergangen	- 2h*6P
- Veranschlagt	- 3h*6P*11 Wochen
Seminar 2: Präsentation und Disputation	48h - 8h*6P
Dokumentation Meilenstein 1	ca. 52.5h
· Einleitung	- 2h
· Personas	- 2h*2P?
· Funktionsumfang	- 3h
· Architektur	- 6h
· Projektmanagement	- 1.5h?
· Aufwandsschätzung	- 3h*6P Teil 1
· Aufwandstzg.: Überarbeiten	- 18h
Vorbereitung Präsentationen	50h

7 AUFWANDSSCHÄTZUNG

Aufgabe	Zeit
- Arbeitszeit einzeln	- 2*2h*6P
- Gemeinsam	- 2*2h*6P
- Finales Überarbeiten	- 2h
Tag der Präsentationen	48h - 2*4h?*6P
Dokumentation Meilenstein 2	52h
· User Stories pro Feature	- 2h
· Technisches Konzept	- 10h
o Logische Schichten	- 3h
§ Struktursicht	- 1h
§ Verteilungssicht	- 1h
§ Verhaltenssicht	- 1h
o Verwendete Technologie/Frameworks	- 2h
o Schnittstellentechnologie	
o Datenmodell	- 2h
§ Logisch	- 15h
§ Physisch	- 5h
	- 10h
Meilenstein 3	28h
· Code Reviews	- 3h*6P
· Technischen Prototyp vorbereiten	- 10h
Meilenstein X	69h
Installations- und Administrationshandbuch	- 18h
Aufteilung des Teams: wer hat was gemacht (mit Namen der Teilnehmer)?	- 3h*6P
Reflektion Projektmanagement	
Reflektion Lernfortschritt	- 16h
Ausblick: was könnte an Ihrem Projekt ergänzt werden?	- 12h
	- 5h
Initialisieren Repositories	7h
Benutzerverwaltung & Authentifizierung / Keycloak aufsetzen	
· Frontend Design	109h
· Backend Einbindung	· 5h
· Kommunikation mit Keycloak-Server	· 18h
· Login/Registrierung mit keycloak	· 12h
· Sichere Speicherung von Benutzerdaten	· 10h
· Inter-App-Authentifizierung (BeSy-InSy)	· 8h
· Troubleshooting	· 12h

7 AUFWANDSSCHÄTZUNG

Aufgabe	Zeit
· Code Review	· 18h
· Testing (Frontend und Backend)	· 6h
· Doku	· 16h
	· 4h
(Neue) Technologien: Einarbeiten & Einlesen	108h
Angular	
o Einzeln	· 6h*ca.2P
o Besprechung im Team	· 2h*6P
Spring Boot	
o Einzeln	· 6h*ca.3P
o Besprechung im Team	· 2h*6P
Nginx	
o Einzeln	· 4h*ca.1P
o Besprechung im Team	· 1h*6P
Keycloak	
o Einzeln	· 8h*ca.2P
o Besprechung im Team	· 2h*6P
Postgres	
o Einzeln	
o Besprechung im Team	· 5h*ca.2P
-	· 1h*6P
CI/CD Pipeline	51h
· Einlesen	- 14h
· Einrichten	- 8h
· Docker Setup & Troubleshooting	- 16h
· Sonarqube	- 6h
· Github Docker Registry	- 4h
· Github Actions	- 3h
Inventarliste	52h
- Frontend Design & Logik	- 18h
- Backend Einbindung	- 16h
- Testing (Front + Back End)	- 10h
- Code Review	- 4h
- Doku	- 4h
Inventarisierungs Wizard	59h
- Frontend Design & Logik	- 12h
- In Webserverstruktur einpflegen	- 4h

7 AUFWANDSSCHÄTZUNG

Aufgabe	Zeit
- Evtl. Kommunikation mit BeSy	- 6h
- Backend Einbindung	- 16h
- Frontend Tests	- 6h
- Backend Testing	- 8h
- Code Review	- 3h
- Doku	- 4h
Inventarisieren	196h
- Bestellübersicht	- 52h
o API für BeSy	- 6h
o Frontend Design	- 12h
o Backend Logik	- 16h
o Testing	- 12h
o Code Review	- 4h
o Doku	- 2h
- Artikelübersicht	- 47h
o Frontend Design	- 10h
o Backend Logik	- 16h
o Testing	- 12h
o Code Review	- 5h
o Doku	- 4h
- Artikel kombinieren	- 30h
o Frontend Design & Logik	- 16h
o Testing	- 6h
o Code Review	- 4h
o Doku	- 4h
- Manuelles Inventaris.	- 29h
o Frontend Design	- 6h
o Backend Logik	- 8h
o Testing	- 8h
o Code Review	- 4h
o Doku	- 3h
- Direkt/Kontrollansicht	- 38h
o Frontend Design	- 10h
o Backend Logik	- 12h
o Testing	- 9h
o Code Review	- 4h
o Doku	- 3h

7 AUFWANDSSCHÄTZUNG

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

7 AUFWANDSSCHÄTZUNG

Aufgabe	Zeit
Statistiken	36h
- Frontend Design & Logik	- 14h
- Backend Einbindung	- 8h
- Testing (Front + Back End)	- 8h
- Code Review	- 3h
- Doku	- 3h
Repository pflegen	18h
API-Design	59h
- Festlegung der API-Ziele	- 6h
- Endpunkt Definition	- 8h*ca.4P
- Sicherheitskonzept	- 5h
- Fehlerbehandlung	- 6h
- Testing	- 6h
- Doku	- 4h
Testprotokoll für das Frontend	9h
- Erste Entwürfe & Planung	- 4h
- Im Team besprechen	- 1h*ca.3P
- Finalisieren	- 1h
- Doku	- 1h
Datenbank aufsetzen	10h
- Einarbeiten	- 3h
- Installieren	- 3h
- Datenmodel implementieren	- 4h
Clean Code Definition	14h
- Erste Entwürfe & Planung	- 1h*6P
- Im Team besprechen	- 1h*6P
- Finalisieren	- 1h
- Doku	- 1h
Datenmodellierung	19h
- Erste Entwürfe	- 2h*2P
- Besprechungen	- 1h*4P
- Überarbeitung	- 8h
- Finalisieren	- 1h
- Doku	- 2h
Verwendete Lizenzen	3h
Summe	1.806h

7 AUFWANDSSCHÄTZUNG

Aufgabe

Zeit

Tabelle 7.1: Aufwandsschätzung