

LeihSy -

Das digitale Verleihsystem der Hochschule Esslingen

Dokumentation der Projektarbeit
im Studiengang
Softwaretechnik und Medieninformatik

vorgelegt von

Asinas Esber
Matr.-Nr.: 768274
Ceyda Yoldas
Matr.-Nr.: 768881
Dennis Mika
Matr.-Nr.: 777972
Yannick Jacobs
Matr.-Nr.: 771697
Malte Stein
Matr.-Nr.: 771640

Betreuer: Andreas Heinrich
Abgabedatum: 13. Januar 2026

Kurzfassung

Das Modul **Projekt Softwaretechnik und Medieninformatik** aus dem vierten Semester beinhaltet die Entwicklung einer **Fullstack-Applikation**. Im Rahmen des Projektes sollen die Studierenden die Kenntnisse, welche sie in den vorherigen Semestern erworben haben, in einem praxisnahen Umfeld einsetzen und vertiefen. Dabei soll das **Projekt- und Zeitmanagement** geübt werden sowie soziale Kompetenzen wie **Teamfähigkeit** und **Konfliktfähigkeit** weiterentwickelt werden. Die folgenden Seiten der Dokumentation befassen sich mit der **Planung**, **Umsetzung** und **Bewertung** des Projektes und beleuchten alle relevanten Aspekte der Projektarbeit.

Schlagwörter: Projektmanagement, Zeitmanagement, Fullstack, Backend, Frontend, UI, Datenbanken, UML

Inhaltsverzeichnis

1 Erklärung der Aufgabe	8
1.1 Ist-Zustand	8
1.2 Soll-Zustand	8
1.3 Zielgruppe	9
1.4 Stakeholder	9
2 Anforderungsanalyse	10
2.1 Must-Have Features	10
2.1.1 Authentifizierung & Benutzerverwaltung	10
2.1.2 Gegenstandsverwaltung	10
2.1.3 Ausleihprozess	11
2.1.4 Email-System	11
2.1.5 Administration & Übersicht	11
2.1.6 Protokollierung & Datenschutz	12
2.1.7 Technische Anforderungen	12
2.2 Nice-To-Have Features	12
2.2.1 Hohe Priorität (++)	12
2.2.2 Mittlere Priorität (+)	12
2.2.3 Niedrige Priorität (-)	13
2.3 Funktionale Anforderungen	13
2.4 Nicht-funktionale Anforderungen	14
2.4.1 Performance und Skalierbarkeit	14
2.4.2 Gebrauchstauglichkeit (Usability)	15
2.4.3 Sicherheit	15
2.4.4 Zuverlässigkeit und Verfügbarkeit	15
2.4.5 Wartbarkeit und Erweiterbarkeit	16
2.4.6 Datenschutz und Compliance	16
2.4.7 Kompatibilität	16
2.4.8 Dokumentation	17
2.5 Qualitätsziele	17
2.6 Constraints	17
2.7 User-Stories	18
2.7.1 Epic 1: Authentifizierung & Benutzerverwaltung	18
2.7.2 Epic 2: Gegenstandsverwaltung	19
2.7.3 Epic 3: Ausleihprozess – Warenkorb	20
2.7.4 Epic 4: Ausleihprozess – Terminverwaltung	21
2.7.5 Epic 5: Ausgabe/Rückgabe vor Ort	22
2.7.6 Epic 6: E-Mail-Benachrichtigungen	23
2.7.7 Epic 7: Protokollierung & Datenschutz	24
2.7.8 Epic 8: Administration & Reporting	25
2.7.9 Epic 9: Integration & Schnittstellen	25
2.7.10 Epic 10: Technische Anforderungen	26
2.7.11 Epic 11: Nice-to-Have Features	26
3 Zeitmanagement	28
3.1 Zeiterfassung	29

4 Projektmanagement	29
4.1 Methode	30
4.2 Organisation	30
4.3 Eingesetzte Tools	30
5 Entwicklungsumgebung	30
5.1 Eingesetzte IDE	31
5.2 Eingesetzte Frameworks	31
5.2.1 Backend Framework	31
5.2.2 Weitere Backend-Bibliotheken	31
5.2.3 Frontend Framework	31
5.2.4 Weitere Frontend-Bibliotheken	32
5.2.5 Lizenzen der eingesetzten Frameworks und Bibliotheken	32
5.2.6 Integration in bereits bestehende Infrastruktur	33
5.3 Server	33
5.4 Datenbank	33
5.4.1 Entwicklungsumgebung mit H2	33
5.4.2 Spring Profiles	33
5.4.3 Datenbankverbindung	34
5.4.4 Automatische Testdaten	34
5.5 Development-Server	35
5.6 Version Control management System	36
5.6.1 CI/CD Pipeline	36
5.7 API-Testing mit Postman	39
5.7.1 Motivation	39
5.7.2 Automatisches Token-Management	39
5.7.3 Collection Variables	39
5.7.4 Test-Requests	40
5.7.5 Vorteile	40
6 UI-Prototyp	40
6.1 Farbpalette	40
6.1.1 Version 1 (Erstkonzept)	40
6.1.2 Version 2 (Wireframe)	40
6.1.3 Finale, farbige Version	41
6.2 Erste Prototypen	41
6.2.1 Version 1 – Grundidee	41
6.2.2 Version 2 – Änderungen gegenüber Version 1	46
6.3 Visuelle Umsetzung der Funktionen	52
6.3.1 Login-Seite	52
6.3.2 Geräte-Details	52
6.3.3 Warenkorb	53
7 User Research	55
7.1 Proto-Personas	55
7.1.1 Persona A – Lena Schmid (Studierende)	55
7.1.2 Persona B – Max Schmidt (IT-Admin / Systemverantwortlicher)	55
7.1.3 Persona C – Prof. Tom Fischer (Lehrender/Verleiher)	56
7.2 Interview Auswertung	57

7.3	Auswertung (Online-Umfrage, n = 28)	58
7.3.1	Stichprobe & Nutzung	58
7.3.2	Zufriedenheit (Ist-Prozess)	58
7.3.3	Größte Pain Points (Top-Nennungen)	58
7.3.4	Adoptionsbereitschaft für ein digitales System	59
7.3.5	Top-Features (max. 3 Stimmen)	59
7.3.6	Unverzichtbare Filter	60
7.3.7	Verfügbarkeit je Campus	61
7.3.8	Storno-Fenster (Abhol-/Rückgabe-Slots)	61
8	Softwarearchitektur	62
8.1	Entity-Relationship-Diagramme	62
8.1.1	Erster Entwurf	62
8.1.2	Änderungen vom ersten Entwurf zur finalen Version	62
8.1.3	Finale Umsetzung	64
8.2	Logische Sichten	66
8.2.1	Verteilungssicht	66
8.2.2	Struktursicht	68
8.2.3	Verhaltenssicht	70
8.3	API-Dokumentation	73
8.3.1	API-Umstrukturierung nach REST Best Practices	74
9	Implementierung	83
9.1	Keycloak	83
9.2	Frontend	84
9.2.1	User Stories	84
9.2.2	Struktursicht	84
10	Features	85
10.1	Admin Dashboard	85
10.1.1	Admin-Menü	85
10.1.2	Admin-Menü: Alle Gegenstände (Übersicht)	86
10.1.3	Admin-Menü: Produktgruppen verwalten	87
10.1.4	Admin-Menü: Einzelne Gegenstände verwalten	89
10.1.5	Admin-Menü: Buchungen verwalten	91
10.1.6	Admin-Menü: Standorte Verwalten	93
10.1.7	Admin-Menü: Gruppen verwalten	94
10.1.8	Admin-Feature: Kategorienverwaltung	95
10.1.9	Admin-Feature: Verleiher-Zuordnung	96
10.2	Verleiher-Menü	98
10.2.1	Verleiher-Menü: Meine Gegenstände	99
10.2.2	Verleiher-Menü: Private Gegenstände verleihen	100
10.2.3	Verleiher-Dashboard: Backend-Implementierung	101
10.2.4	Offene Anfragen für Verleiher: Backend-Implementierung	102
10.3	Nutzer-Menü	103
10.3.1	Nutzer-Menü: Meine Buchungen	104
10.3.2	Mein Gruppen	105
10.4	System-Funktionen	106
10.4.1	Bestätigung der Abholung	106

10.4.2 Dokumentation der Rückgabe	106
10.4.3 Benachrichtigung bei Statusänderungen	107
10.4.4 Automatisierte Erinnerungen und Mahnwesen	107
10.5 Warenkorb	108
10.5.1 Benutzeroberfläche	111
10.5.2 Backend-Implementierung	112
10.6 Authentifizierung und Benutzerverwaltung	114
10.6.1 Überblick & Architektur	114
10.6.2 JWT-Token und Security-Konfiguration	115
10.6.3 Rollensystem	116
10.6.4 Frontend-Integration	117
10.6.5 Automatische Benutzersynchronisation	118
10.7 Bildverwaltung	120
10.7.1 Anforderungen	120
10.7.2 Architektur-Entscheidung: Filesystem vs. Datenbank	121
10.7.3 Backend-Implementierung	121
10.7.4 Frontend-Integration	123
10.7.5 Deployment-Überlegungen	124
10.7.6 Zusammenfassung	124
10.8 Automatische Buchungsstornierung	124
10.8.1 Anforderungen	124
10.8.2 Implementierung	124
10.8.3 Zusammenfassung	125
10.9 Studentengruppen für gemeinsame Ausleihen	125
10.9.1 Anforderungen	125
10.9.2 Architektur-Entscheidung: Gruppen vs. Sammel-Buchungen	126
10.9.3 Datenmodell	126
10.9.4 Backend-Implementierung	126
10.9.5 Zusammenfassung	128
10.10 InSy-Integration	128
10.10.1 Anforderungen	128
10.10.2 Architektur-Entscheidung: Staging-Tabelle vs. Direktimport	128
10.10.3 Backend-Implementierung	129
10.10.4 Entwicklungsgeschichte und Iterationen	131
10.10.5 Frontend-Integration	131
10.10.6 Mock-Daten für Entwicklung	132
10.10.7 Zusammenfassung	133
10.11 Sicheres Token-basiertes QR-Code-System	133
10.11.1 Motivation und Anforderungen	134
10.11.2 Architektur-Entscheidung: Token-Transaktionen	134
10.11.3 Backend-Implementierung	134
10.11.4 Frontend-Integration	135
10.11.5 Zusammenfassung	135
11 Ausblick	136
11.1 Datenmanagement	136
11.1.1 Performance und Wartbarkeit	136
11.2 Funktionale Erweiterungen	137

11.2.1	Erweiterung der Projektgruppen	137
11.2.2	Verleihergruppen	137
11.2.3	In-App Nachrichten	137
11.2.4	Verlängerung der Ausleihen	138
11.2.5	Herausforderungen bei Ausleihen anhand verschiedener Use-Cases .	138

Abbildungsverzeichnis

1	Netzwerkübersicht Development-Server	35
2	Alter Entwurf der CI/CD Pipeline Architektur in GitHub	37
3	Finale CI/CD Pipeline in GitHub	38
4	Login screen	42
5	Inventarkatalog	43
6	Geräte-Details Seite	44
7	Persönlicher Bereich	45
8	Aktualisierter Inventarkatalog	46
9	Kalender in Produktdetails	47
10	Abholungszeitfenster	48
11	Warenkorb	49
12	Standort des Geräts	50
13	Aktualisierter persönlicher Bereich	51
14	Aktualisierter Login screen	52
15	Geräte-Details Seite mit aktualisierten Design	53
16	Verbesserter Warenkorb	54
17	Persona Lena Schmid	55
18	Persona Max Schmidt	56
19	Persona Tom Fischer	57
20	Anteil Studenten mit kürzlichen Ausleihen	58
21	Zufriedenheit mit aktuellem Prozess	58
22	aktuelle Pain Points	59
23	Adoptionsbereitschaft für digitales System	59
24	Wichtigste Features	60
25	Unverzichtbare Filter	60
26	Wichtigkeit Verfügbarkeit nach Campus	61
27	Länge Stornierbarkeit	61
28	Erster Entwurf Entity-Relationship-Diagramm	62
29	Entity-Relationship-Diagramm	66
30	Diagramm zur Verteilungssicht	67
31	Komponentendiagramm	68
32	Sequenzdiagramm zum Ausleihprozess	71
33	Sequenzdiagramm zum Rückgabeprozess	72
34	Sequenzdiagramm zum Login	73
35	Kategorienverwaltung im Admin-Dashboard	95
36	Dialog zur Verleiher-Zuordnung in der Geräteverwaltung	96
37	Produkt-Detailseite	108
38	Zum Warenkorb Hinzugefügt	109
39	Warenkorb-Seite	110
40	Warenkorb-Checkout	110
41	Buchungsverlauf eines Users	111

1 Erklärung der Aufgabe

1.1 Ist-Zustand

Aktuell erfolgt die Geräteausleihe an der Hochschule in Papierform. Um ein Gerät auszuleihen, müssen sich die Studierenden einen analogen Leihchein ausdrucken und sich bei einer betreuenden Person melden. Eine digitale Übersicht über vorhandene oder ausgeliehene Geräte existiert derzeit nicht und die Rückgaben werden händisch notiert, wodurch häufig Unklarheiten entstehen.

Typische Probleme:

- Geräte werden verspätet zurückgebracht.
- Keine automatische Erinnerung an Rückgabefristen.
- Keine genaue Übersicht über verfügbare Geräte.
- Papierverbrauch durch gedruckte Leihscheine führt zu unnötiger Ressourcenverschwendungen.
- Der gesamte Prozess ist zeitaufwendig und fehleranfällig.

Durch die fehlende Digitalisierung ist der Ausleih-Prozess unübersichtlich, ineffizient und nicht nachhaltig. Und genau das wollen wir ändern.

1.2 Soll-Zustand

Unser Ziel ist es, den bisherigen papierbasierten Ablauf mit einer gebrauchstauglichenen und nachhaltigen Website zu ersetzen und den Ausleih-Prozess von Anfang bis Ende digital im Überblick zu behalten.

Studierende, Dozenten und Mitarbeitende können sich dabei mit ihrem Hochschul-Account anmelden, verfügbare Geräte einsehen sowie Reservierungen und Verlängerungen tätigen.

Unsere wichtigen Punkte dabei sind:

- Ein klarer, schneller, papierloser Ablauf und einfache Bedienung.
- Webbasierte Plattform.
- Anmeldung über Hochschul-Account.
- Unterschiedliche Rollen: Studierende/Dozenten und Administratoren.
- Einfache Verwaltung für Studierende, Dozenten und Mitarbeitende.
- Integration von Benachrichtigungsfunktionen.

- Eine Datenbank zur Speicherung von Geräten, Nutzern und Ausleihen.

Bei der Software-Implementierung achten wir darauf, dass der Code strukturiert, effizient und funktionsfähig ist.

1.3 Zielgruppe

Die Hauptzielgruppe unseres digitalen Verleihsystems sind Studierende, Dozenten und Mitarbeitende der Hochschule Esslingen.

Die Dozenten möchten Geräte für Lehrveranstaltungen bereitstellen, zum Beispiel für das Wahlfachmodul “Digitale Fotografie”. Auch für die Privatnutzung soll es möglich sein, dass einige Dozenten ihr eigenes Equipment hochladen und verleihen.

Studierende benötigen daher eine schnelle und digitale Möglichkeit, Geräte auszuleihen. Die Mitarbeitenden, die den Ausleih-Prozess betreuen, brauchen eine übersichtliche Verwaltung, um alle Ausleihen gründlich und vertraulich dokumentieren zu können.

Das digitale Leihsystem sorgt dafür, dass die Zielgruppen Zeit sparen, den Überblick behalten und einen reibungslosen Prozess haben.

1.4 Stakeholder

Die relevanten Stakeholder des Projekts umfassen mehrere Rollen mit unterschiedlichen Interessen und Verantwortlichkeiten.

- Projektbetreuer
- Kunde
- Potenzielle Nutzergruppen
- Entwicklerteam
- Leiter des Moduls

Der Projektbetreuer Andreas Heinrich begleitet das Projekt fachlich und methodisch und stellt die Einhaltung der Projektziele sicher und bewertet schlussendlich das Gesamtergebnis.

Der Kunde Christian Haas definiert die funktionalen Anforderungen und bringt die Ideen für die zukünftige Weiterentwicklung ein.

Die potenziellen Nutzergruppen wurden bereits in Abschnitt 1.3 näher betrachtet und stellen die Endnutzer dar.

Das Entwicklerteam ist für die technische Umsetzung, Implementierung, Qualitätssicherung und Visualisierung verantwortlich und arbeiten in Absprache mit dem Projektbetreuer.

Der Leiter des Moduls Herr Nitzsche weist die Entwicklerteams den Projektbetreuern zu und ist zuständig für einen formalen Ablauf der Endbewertung .

2 Anforderungsanalyse

Die Anforderungsanalyse für das Projekt LeihSy bildet die Grundlage für die Entwicklung des digitalen Verleihsystems an der Hochschule Esslingen. Sie wurde in Zusammenarbeit mit dem Betreuer Andreas Heinrich und dem Kunden Christian Haas erarbeitet und beschreibt sowohl funktionale als auch nicht-funktionale Anforderungen. Die Anforderungen sind priorisiert in Must-Have Features und Nice-to-Have Features (erweiterte Funktionalität) unterteilt, um eine schrittweise und agile Umsetzung innerhalb des Projektzeitraums zu ermöglichen.

2.1 Must-Have Features

2.1.1 Authentifizierung & Benutzerverwaltung

- **Single Sign-On (SSO) via Keycloak:** Integration von OpenID Connect zur Authentifizierung mit dem Hochschul-RZ-Account
- **Rollenbasierte Zugriffskontrolle:** Drei Benutzerrollen (Administrator, Verleiher, Student/Entleiher) mit spezifischen Berechtigungen
- **Rollenvergabe über Keycloak-Benutzergruppen:** Automatische Zuweisung von Rollen basierend auf Gruppenzugehörigkeit

2.1.2 Gegenstandsverwaltung

- **CRUD-Operationen für Verleihgegenstände:** Erstellen, Anzeigen, Bearbeiten und Löschen von Gegenständen durch Administratoren und Verleiher
- **Pflichtattribute:** Inventarnummer, Name, Beschreibung, Kategorie, Foto, Zubehör, Lagerort, Status
- **Sets von gleichartigen Gegenständen:** Möglichkeit, mehrere identische Gegenstände mit eindeutigen Nummern als Set anzulegen (z. B. 10x Meta Quest)
- **Kategorien und Filter:** Kategorisierung von Gegenständen und Filtermöglichkeiten nach Kategorie, Verfügbarkeit, Lagerort und Volltext-Suche
- **Verleiher-Zuordnung:** Verleiher sehen und verwalten nur ihre zugeordneten Gegenstände

- **Benötigte Zusatzgegenstände:** Automatische Anzeige von empfohlenem Zubehör zu Hauptgegenständen (z. B. Speicherkarte zu Kamera)

2.1.3 Ausleihprozess

- **Warenkorb-Funktionalität:** Studierende können mehrere Gegenstände in einen Warenkorb legen
- **Terminvorschlag über Kalender:** Auswahl des gewünschten Ausleihzeitraums mit Anzeige der Verfügbarkeit
- **Verfügbarkeitskalender auf Detailseite:** Anzeige der Verfügbarkeit eines Gegenstandes in einer Kalenderansicht
- **Bestätigung/Ablehnung durch Verleiher:** Verleiher können Ausleiheanfragen bestätigen oder ablehnen
- **Automatische Stornierung:** Nicht bestätigte Anfragen werden nach 24 Stunden automatisch storniert
- **Digitale Ausgabe und Rückgabe:** Email-Bestätigung bei Ausgabe und Rückgabe vor Ort mit Login-basierter Bestätigung durch Studierende
- **Ausleihverlauf:** Studierende können ihren persönlichen Ausleihverlauf einsehen (aktuelle, offene und vergangene Ausleihen)

2.1.4 Email-System

- **Automatische Benachrichtigungen:** Email-Versand bei verschiedenen Ereignissen (Anfrage erstellt, bestätigt, abgelehnt, Ausgabe, Rückgabe)
- **Erinnerungsmails:** Automatische Erinnerung 2 Tage vor Fälligkeit
- **Überfälligkeit-Benachrichtigungen:** Benachrichtigung an Student und Verleiher bei nicht fristgerechter Rückgabe

2.1.5 Administration & Übersicht

- **Admin-Dashboard:** Zentrale Übersicht über offene Anfragen, bestätigte aber nicht abgeholt Ausleihen, aktuelle und zukünftige Ausleihen sowie Statistiken
- **Systemkonfiguration:** Administratoren können Systemeinstellungen verwalten

2.1.6 Protokollierung & Datenschutz

- **Audit-Log:** Protokollierung aller relevanten Aktionen (Wer hat wann was ausgeliehen/zurückgegeben, Änderungen an Gegenständen, Login-Versuche)
- **DSGVO-Konformität:** Datensparsamkeit, Zweckbindung, definierte Aufbewahrungsfristen, Recht auf Datenexport und Löschung
- **Datenschutzkonforme Speicherung:** Verschlüsselte Session-Daten, automatische Löschung temporärer Daten

2.1.7 Technische Anforderungen

- **Containerisierung mit Docker:** Alle Komponenten (Frontend, Backend, Datenbank, Keycloak) sind containerisiert und über Docker-Compose startbar
- **PostgreSQL-Datenbank:** Persistente Datenspeicherung
- **Responsive Design:** Optimierung für Desktop und Mobile

2.2 Nice-To-Have Features

2.2.1 Hohe Priorität (++)

- **Budgetplanung mit Tagessätzen:** Verwaltung von Budgets mit definierten Tagessätzen für Gegenstände
- **Private Verleihgegenstände:** Möglichkeit für Nutzer, private Gegenstände im System zu verwalten und zu verleihen
- **Batch-Erstellung von Gegenständen:** Vereinfachte Massenerstellung von Gegenständen
- **CI/CD-Pipeline:** Automatisierte Builds, Tests und Deployments über GitHub Actions
- **QR/Barcode-Scanner:** Scannen von QR-Codes zur schnellen Identifikation von Gegenständen bei Ausgabe und Rückgabe
- **Code-Qualitätsprüfung:** Integration von SonarQube zur statischen Code-Analyse

2.2.2 Mittlere Priorität (+)

- **Verleihgruppen:** Mehrere Verleiher können als Gruppe zusammenarbeiten und gemeinsam Bestände verwalten

- **InSy-Integration:** POST-API-Endpoint für Datenimport aus dem Inventarisierungssystem InSy (Inventarnummer, Name, Beschreibung, Kategorie, Lagerort)
- **Verschiedene Email-Templates:** Individuelle Templates für unterschiedliche Anlässe und Zielgruppen
- **Verlängerungsfunktion:** Studierende können laufende Ausleihen verlängern (sofern keine Folgebuchungen existieren)

2.2.3 Niedrige Priorität (-)

- **Studentengruppen mit gemeinsamem Budget:** Studierende können sich zu Gruppen zusammenschließen und ein gemeinsames Budget nutzen
- **Dozenten-Freigabe-Workflow:** Dozenten können Ausleiheanfragen für Projektarbeiten freigeben
- **Detaillierte Budget-Reports:** Umfassende Auswertungen und Reports über Budgetauslastung und Kosten

2.3 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben das grundlegende Verhalten und die Kernfunktionalitäten des Systems LeihSy aus fachlicher Sicht. Sie definieren, was das System leisten muss, ohne dabei auf technische Implementierungsdetails einzugehen.

2.3.0.1 Benutzerverwaltung Das System muss eine sichere Authentifizierung über Keycloak (SSO) mit dem Hochschul-RZ-Account ermöglichen und drei Benutzerrollen (Administrator, Verleiher, Student/Entleiher) mit unterschiedlichen Berechtigungen unterstützen.

2.3.0.2 Gegenstandsverwaltung Das System muss die vollständige Verwaltung von Verleihgegenständen ermöglichen, einschließlich Anlegen, Bearbeiten, Löschen, Kategorisieren und Filtern. Gegenstände müssen eindeutig identifizierbar sein und können als Sets mit mehreren Einzelexemplaren angelegt werden.

2.3.0.3 Ausleihprozess Das System muss den gesamten Ausleihprozess digital abbilden: von der Anfrage über die Bestätigung/Ablehnung durch den Verleiher bis hin zur dokumentierten Ausgabe und Rückgabe vor Ort. Verfügbarkeiten müssen automatisch geprüft und nicht bestätigte Anfragen nach 24 Stunden automatisch gestrichen werden.

2.3.0.4 Benachrichtigungssystem Das System muss automatisch Email-Benachrichtigungen zu allen relevanten Ereignissen im Ausleihprozess versenden, einschließlich Erinnerungen vor Fälligkeit und Mahnungen bei Überfälligkeit.

2.3.0.5 Übersichten und Verlauf Das System muss allen Benutzergruppen rollenbezogene Übersichten bereitstellen: Studierende sehen ihren persönlichen Ausleihverlauf, Verleiher sehen Anfragen und Ausleihen ihrer zugeordneten Gegenstände, Administratoren erhalten ein zentrales Dashboard mit Statistiken und Gesamtübersicht.

2.3.0.6 Integration Das System muss eine REST-API-Schnittstelle zum Import von Gegenstandsdaten aus dem Inventarisierungssystem InSy bereitstellen.

2.3.0.7 Protokollierung Das System muss alle sicherheitsrelevanten und geschäftsrelevanten Ereignisse lückenlos und unveränderbar protokollieren (Audit-Log).

2.3.0.8 Datenschutz Das System muss DSGVO-konform sein und die Prinzipien der Datensparsamkeit, Zweckbindung und Speicherbegrenzung einhalten. Benutzer müssen ihre Daten exportieren und die Löschung ihres Accounts beantragen können.

2.4 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben qualitative Eigenschaften des Systems LeihSy, die über die reine Funktionalität hinausgehen. Sie definieren, wie gut das System seine Funktionen erfüllt und welche Rahmenbedingungen eingehalten werden müssen.

2.4.1 Performance und Skalierbarkeit

- **Antwortzeiten:** Das System muss eine hohe Reaktionsgeschwindigkeit gewährleisten. Seiten müssen innerhalb von maximal 2 Sekunden laden, API-Anfragen innerhalb von 500 Millisekunden beantwortet werden.
- **Gleichzeitige Benutzer:** Das System muss mindestens 20 gleichzeitige Benutzer ohne Performanceeinbußen unterstützen können. Bei höherer Last muss eine horizontale Skalierung möglich sein.
- **Datenbankperformance:** Datenbankabfragen müssen optimiert sein (Indizes, effiziente Queries). Die Datenbank muss mindestens 500 Gegenstände und 100 Ausleihvorgänge verwalten können.

2.4.2 Gebrauchstauglichkeit (Usability)

- **Intuitive Bedienung:** Die Benutzeroberfläche muss intuitiv und selbsterklärend sein. Neue Benutzer sollen ohne Schulung grundlegende Funktionen (Suchen, Anfragen, Warenkorb) nutzen können.
- **Responsive Design:** Das System muss auf verschiedenen Endgeräten (Desktop, Tablet, Smartphone) einwandfrei funktionieren und eine optimale Benutzererfahrung bieten.
- **Barrierefreiheit:** Die Anwendung soll grundlegende Accessibility-Standards einhalten (kontrastreiche Darstellung, semantisches HTML, Tastaturnavigation).
- **Mehrsprachigkeit (Optional):** Das System sollte mehrsprachig erweiterbar sein. Initial wird Deutsch unterstützt, eine spätere Erweiterung um Englisch soll möglich sein.

2.4.3 Sicherheit

- **Authentifizierung und Autorisierung:** Alle Zugriffe müssen über Keycloak authentifiziert sein. Nicht authentifizierte Anfragen werden abgelehnt. Autorisierung erfolgt rollenbasiert auf allen Endpunkten.
- **Datenverschlüsselung:** Die Kommunikation zwischen Client und Server muss über HTTPS/TLS verschlüsselt sein. Passwörter und sensitive Daten müssen verschlüsselt gespeichert werden.
- **API-Sicherheit:** API-Endpunkte (z. B. InSy-Import) müssen über API-Keys oder OAuth authentifiziert sein. Rate-Limiting verhindert Missbrauch.
- **Session-Management:** Sessions müssen nach Inaktivität automatisch ablaufen (Standard: 8 Stunden). Logout muss sowohl lokale als auch Keycloak-Session beenden.
- **Input-Validierung:** Alle Benutzereingaben müssen auf Client- und Server-Seite validiert werden, um Injection-Angriffe (SQL, XSS) zu verhindern.

2.4.4 Zuverlässigkeit und Verfügbarkeit

- **Verfügbarkeit:** Das System soll eine Verfügbarkeit von mindestens 95 % während der Hochschul-Öffnungszeiten (Mo–Fr, 8–18 Uhr) gewährleisten.
- **Fehlerbehandlung:** Das System muss Fehler graceful behandeln und dem Benutzer verständliche Fehlermeldungen anzeigen. Technische Details werden nur an Administratoren ausgegeben.

- **Datensicherung:** Datenbank-Backups müssen täglich automatisch erstellt und mindestens 30 Tage aufbewahrt werden. Recovery-Tests müssen regelmäßig durchgeführt werden.

2.4.5 Wartbarkeit und Erweiterbarkeit

- **Code-Qualität:** Der Code muss Clean-Code-Prinzipien folgen, gut dokumentiert und testbar sein. Komplexe Logik muss durch Kommentare erläutert werden.
- **Modularität:** Das System muss modular aufgebaut sein (Frontend/Backend-Trennung, Service-Architektur). Einzelne Module können unabhängig voneinander weiterentwickelt werden.
- **Versionierung:** Der Code muss über Git versioniert werden. Commit-Messages folgen einem einheitlichen Standard (Conventional Commits).
- **Testabdeckung:** Kritische Funktionen müssen durch Unit-Tests abgedeckt sein (angestrebte Testabdeckung: >60 %). Integration-Tests decken die wichtigsten User-Flows ab.
- **Deployment:** Das System muss über Docker-Container deployt werden können. Ein `docker-compose.yml` ermöglicht das Starten der gesamten Anwendung mit einem Befehl.

2.4.6 Datenschutz und Compliance

- **DSGVO-Konformität:** Das System muss alle Anforderungen der DSGVO erfüllen: Datensparsamkeit, Zweckbindung, Speicherbegrenzung, Auskunftsrecht, Recht auf Löschung.
- **Datenminimierung:** Es dürfen nur die für den Betrieb notwendigen personenbezogenen Daten gespeichert werden. Keine Sammlung von Daten „auf Vorrat“.
- **Audit-Trail:** Alle Zugriffe auf personenbezogene Daten und sicherheitsrelevante Aktionen müssen nachvollziehbar protokolliert werden.
- **Datenschutzerklärung:** Das System muss eine Datenschutzerklärung bereitstellen, die Benutzer über die Datenverarbeitung informiert.

2.4.7 Kompatibilität

- **Browser-Kompatibilität:** Das System muss in den gängigen modernen Browsern funktionieren (Chrome, Firefox, Safari, Edge – jeweils aktuelle Version und eine Version zurück).

- **Keycloak-Kompatibilität:** Das System muss mit der an der Hochschule eingesetzten Keycloak-Version kompatibel sein und OpenID Connect-Standards einhalten.
- **API-Standards:** REST-APIs müssen RESTful-Prinzipien folgen und Standard-HTTP-Methoden (GET, POST, PUT, DELETE) verwenden.

2.4.8 Dokumentation

Technische Dokumentation: Es muss eine technische Dokumentation existieren, die Architektur, Datenmodell, API-Schnittstellen und Deployment-Prozess beschreibt.

2.5 Qualitätsziele

Die Qualitätsziele der App orientieren sich am arc42-Modell und stellen sicher, dass das System sowohl technisch als auch aus Nutzersicht geeignet ist. Das Projekt orientiert sich an vier wesentlichen Qualitätszielen die in Tabelle 1 näher beschrieben werden

Qualitätsziel	Beschreibung
Transferierbarkeit	Die Web-Applikation soll auf unterschiedlichen Endgeräten wie PCs und mobilen Geräten lauffähig sein.
Funktionale Tauglichkeit	Das System stellt alle Funktionen bereit, die den definierten funktionalen Anforderungen entsprechen.
Kompatibilität	Die App ist mit bestehenden Systemen (InSy) innerhalb der vorhandenen Infrastruktur kompatibel.
Bedienbarkeit	Die App ermöglicht eine intuitive, verständliche und effiziente Nutzung für die Anwender.

Tabelle 1: Qualitätsziele der App gemäß arc42

2.6 Constraints

Die Randbedingungen (Constraints) des Projekts ergeben sich aus organisatorischen und technischen Vorgaben, die in Tabelle 2 näher beschrieben werden.

Kategorie	Constraint	Beschreibung
Organisatorisch	Zeit	Das Projekt ist auf die Dauer eines Semesters begrenzt und wird parallel zum regulären Studium durchgeführt.
Organisatorisch	Entwicklerteam	Das Entwicklerteam hat vor Projektbeginn noch nicht gemeinsam gearbeitet.
Technisch	Fachliche Erfahrung	Das Entwicklerteam verfügt über begrenzte fachliche Erfahrung im relevanten Themengebiet.
Technisch	Hardware	Entwicklung und Betrieb erfolgen größtenteils lokal oder auf Servern im Hochschulkontext.
Technisch	Software	Externe Software und Bibliotheken müssen Open Source sein.

Tabelle 2: Projekt-Constraints gemäß arc42

2.7 User-Stories

Die User Stories beschreiben die Anforderungen aus Sicht der verschiedenen Benutzergruppen und des Systems.

2.7.1 Epic 1: Authentifizierung & Benutzerverwaltung

US-001: Als Student möchte ich mich mit meinem RZ-Account anmelden können

- Akzeptanzkriterien:
 - Login über Keycloak
 - Weiterleitung nach erfolgreichem Login
 - Session-Management

US-002: Als Administrator möchte ich Rollen verwalten können

- Akzeptanzkriterien:
 - Rollen aus Keycloak-Gruppen übernehmen
 - Berechtigungen pro Rolle definiert

US-003: Als Verleiher möchte ich nur meine zugeordneten Gegenstände sehen

- Akzeptanzkriterien:
 - Zuordnung Verleiher ↔ Gegenstände

2.7.2 Epic 2: Gegenstandsverwaltung

US-004: Als Admin möchte ich einzelne Gegenstände anlegen können

- Akzeptanzkriterien:
 - Stammdaten (Name, Inventarnr, Beschreibung, Kategorie, Lagerort)
 - Formular mit allen Pflichtfeldern
 - Bildupload
 - Speicherung in DB

US-005: Als Admin möchte ich Sets von gleichen Gegenständen anlegen

- Akzeptanzkriterien:
 - Anzahl eingeben
 - Automatische Nummerierung
 - Jedes Item eindeutig identifizierbar

US-006: Als Admin möchte ich Gegenstände bearbeiten/löschen können

- Akzeptanzkriterien:
 - Edit-Formular
 - Soft-Delete (für Protokolle)
 - Nur löschen, wenn keine aktiven Ausleihen

US-007: Als Student möchte ich verfügbare Gegenstände durchsuchen können

- Akzeptanzkriterien:
 - Übersichtsseite mit allen Gegenständen
 - Filter nach Kategorie, Verfügbarkeit, Lagerort
 - Suchfunktion (Volltextsuche)
 - Anzeige Verfügbarkeit

US-008: Als Student möchte ich Detailinfos zu Gegenständen sehen

- Akzeptanzkriterien:
 - Detailseite mit Bild, Beschreibung
 - Anzeige benötigtes Zubehör
 - Verfügbarkeitskalender

US-009: Als System möchte ich Bilder speichern und ausliefern können

- Akzeptanzkriterien:
 - Bildupload (max. Größe 5 MB, Format JPG/PNG/WebP)
 - Speicherung im Filesystem
 - Optimierung/Thumbnails

US-026: Als Admin möchte ich Zusatzgegenstände zu Hauptgegenständen verknüpfen

- Akzeptanzkriterien:
 - Auswahl von Zusatzgegenständen bei Gegenstandsbearbeitung
 - Markierung als „empfohlen“ oder „erforderlich“
 - Anzeige beim Hinzufügen zum Warenkorb

US-027: Als Admin möchte ich Kategorien verwalten können

- Akzeptanzkriterien:
 - Kategorien hinzufügen, bearbeiten
 - Vordefinierte Kategorien (VR-Equipment, Foto-Equipment, IT-Geräte, Audio, Video, Sonstiges)
 - Kategorien können nicht gelöscht werden, wenn Gegenstände zugeordnet sind

2.7.3 Epic 3: Ausleihprozess – Warenkorb

US-010: Als Student möchte ich Gegenstände zum Warenkorb hinzufügen

- Akzeptanzkriterien:
 - Button „In den Warenkorb“
 - Warenkorb-Icon mit Counter
 - Warenkorbsicht

US-011: Als Student möchte ich im Warenkorb einen Zeitraum wählen

- Akzeptanzkriterien:
 - Kalender-Widget
 - Validierung Verfügbarkeit
 - Anzeige Konflikte

- Blockierung nicht verfügbarer Zeiträume

US-012: Als Student möchte ich eine Ausleihanfrage abschicken

- Akzeptanzkriterien:

- Alle Warenkorb-Items auf einmal
- Status „Wartend auf Bestätigung“
- Email an Verleiher
- Optional: Kommentarfeld

US-028: Als Student möchte ich empfohlene Zusatzgegenstände sehen

- Akzeptanzkriterien:

- Beim Hinzufügen zum Warenkorb werden Zusatzgegenstände angezeigt
- Zusatzgegenstände können mit einem Klick hinzugefügt werden
- Warnung bei nicht verfügbaren erforderlichen Zusatzgegenständen

2.7.4 Epic 4: Ausleihprozess – Terminverwaltung

US-013: Als Verleiher möchte ich offene Anfragen sehen

- Akzeptanzkriterien:

- Dashboard mit Pending-Anfragen
- Sortierung nach Datum
- Filter nach Gegenstand
- Hervorhebung von Anfragen nahe der 24h-Frist

US-014: Als Verleiher möchte ich Anfragen bestätigen/ablehnen

- Akzeptanzkriterien:

- Buttons Annehmen/Ablehnen
- Pflichtfeld für Begründung bei Ablehnung
- Optional: Kommentarfeld bei Bestätigung
- Status-Update
- Email an Student

US-015: Als System möchte ich unbestätigte Anfragen nach 24h stornieren

- Akzeptanzkriterien:
 - Cronjob/Scheduled Task
 - Automatische Stornierung
 - Email-Benachrichtigung an Student
 - Gegenstände werden wieder als verfügbar markiert

US-016: Als Student möchte ich eine Ausleihe verlängern können

- Akzeptanzkriterien:
 - Verlängerung anfragen
 - Nur wenn verfügbar (keine Folgebuchungen)
 - Verleiher muss bestätigen
 - Maximal 2 Verlängerungen pro Ausleihe

US-017: Als Student möchte ich meinen Ausleihverlauf sehen

- Akzeptanzkriterien:
 - Liste aller Ausleihen (vergangene + aktuelle + offene Anfragen)
 - Status angezeigt (Ausgeliehen, Ausstehend, Bestätigt, Zurückgegeben, Abgelehnt)
 - Details einsehbar
 - Chronologische Sortierung

US-029: Als Verleiher möchte ich aktuelle und zukünftige Ausleihen sehen

- Akzeptanzkriterien:
 - Übersicht über alle aktuell ausgeliehenen Gegenstände
 - Übersicht über bestätigte, aber noch nicht abgeholt Ausleihen
 - Sortierung nach Rückgabe-/Abholtermin
 - Hervorhebung überfälliger Rückgaben

2.7.5 Epic 5: Ausgabe/Rückgabe vor Ort

US-018: Als System möchte ich bei Abholung eine Bestätigungsmail senden

- Akzeptanzkriterien:
 - Verleiher triggert Email

- Link mit Token
- Gültig für 15 Minuten

US-019: Als Student möchte ich die Ausgabe per Email bestätigen

- Akzeptanzkriterien:

- Login via Keycloak beim Klick
- Bestätigung speichern
- Status → „Ausgeliehen“
- Zeitstempel erfassen

US-020: Als System möchte ich die Rückgabe dokumentieren

- Akzeptanzkriterien:

- Analog zu Ausgabe
- Status → „Verfügbar“
- Protokolleintrag
- Zeitstempel erfassen

US-030: Als Verleiher möchte ich Gegenstände per QR-Code scannen können

- Akzeptanzkriterien:

- QR-Code wird für jeden Gegenstand generiert
- QR-Code als PDF downloadbar
- Scanner-Funktion in der Webanwendung (Kamera-Zugriff)
- Nach Scan wird Gegenstand automatisch geladen
- Alternative: Manuelle Eingabe der Inventarnummer

2.7.6 Epic 6: E-Mail-Benachrichtigungen

US-021: Als System möchte ich Erinnerungsmails versenden

- Akzeptanzkriterien:

- 2 Tage vor Rückgabe
- Bei Überfälligkeit (1 Tag nach Fälligkeit, dann wöchentlich)
- Cronjob täglich
- Überfälligkeit-Mail geht an Student und Verleiher

US-022: Als System möchte ich Statusänderungs-Mails versenden

- Akzeptanzkriterien:
 - Bei Bestätigung/Ablehnung von Anfragen
 - Bei Stornierung
 - Bei Ausgabe/Rückgabe
 - Template-basiert mit Corporate Design
 - Personalisierung mit Benutzerdaten

2.7.7 Epic 7: Protokollierung & Datenschutz

US-023: Als System möchte ich alle Ausleihe-Ereignisse protokollieren

- Akzeptanzkriterien:
 - Ausgeliehen, Zurückgegeben, Verlängert, Erstellt, Geändert, Gelöscht
 - Timestamp, User, Aktion, betroffenes Objekt
 - Unveränderbar (Write-Once)
 - Login-Versuche protokollieren

US-024: Als System möchte ich Studentendaten sparsam speichern

- Akzeptanzkriterien:
 - Nur Name, Email, Matrikelnummer aus Keycloak
 - Keine Duplikation
 - Automatisches Löschen nach definierten Fristen (abgeschlossene Ausleihen nach 2 Jahren, Logs nach 1 Jahr)
 - Anonymisierung inaktiver Accounts nach 3 Jahren

US-031: Als Admin möchte ich Audit-Logs einsehen können

- Akzeptanzkriterien:
 - Chronologische Liste aller Ereignisse
 - Filtermöglichkeiten (Zeitraum, Benutzer, Aktionstyp)
 - Detailansicht mit allen Log-Informationen

2.7.8 Epic 8: Administration & Reporting

US-025: Als Admin möchte ich einen Überblick über alle Ausleihen haben

- Akzeptanzkriterien:
 - Dashboard mit Statistiken
 - Aktuelle Ausleihen
 - Überfällige Items
 - Offene Anfragen
 - Bestätigte, aber nicht abgeholt Ausleihen
 - Vorplanung (Vorschau zukünftiger Ausleihen, Filterung)
 - Top 10 meistgeliehene Gegenstände
 - Auslastung nach Kategorie

US-034: Als Admin möchte ich Verleiher zu Gegenständen zuordnen können

- Akzeptanzkriterien:
 - Bei Gegenstandsbearbeitung Verleiher auswählbar
 - Übersicht aller Zuordnungen
 - Filter nach Verleiher
 - Ein Gegenstand kann nur einem Verleiher zugeordnet sein

2.7.9 Epic 9: Integration & Schnittstellen

US-037: Als System möchte ich Daten aus InSy importieren können

- Akzeptanzkriterien:
 - REST-API Endpoint: POST /api/insy/import
 - Authentifizierung über API-Key
 - Import von Inventarnummer, Name, Beschreibung, Kategorie, Lagerort
 - Validierung der Daten
 - Bei bestehender Inventarnummer: Update statt Duplikat
 - Import-Vorgänge werden protokolliert

2.7.10 Epic 10: Technische Anforderungen

US-039: Als Entwickler möchte ich das System über Docker deployen können

- Akzeptanzkriterien:
 - Dockerfile für Frontend, Backend, Datenbank
 - `docker-compose.yml` für Orchestrierung
 - Environment-Variablen für Konfiguration
 - Persistent Volumes für Datenbank
 - Health-Checks für alle Services
 - Start mit `docker-compose up`

US-040: Als System möchte ich auf verschiedenen Geräten funktionieren

- Akzeptanzkriterien:
 - Responsive Design für Desktop, Tablet, Smartphone
 - Optimierte Layouts für verschiedene Bildschirmgrößen
 - Touch-Gesten auf Mobile unterstützt

2.7.11 Epic 11: Nice-to-Have Features

US-032: Als Student möchte ich meine Daten exportieren können

- Akzeptanzkriterien:
 - Export Button
 - Export PDF
 - Enthält Benutzerprofil und Ausleihe

US-033: Als Student möchte ich die Löschung meines Accounts beantragen können

- Akzeptanzkriterien:
 - Löschungsantrag über Formular
 - Keine aktiven Ausleihen dürfen bestehen
 - Anonymisierung statt vollständiger Löschung (Audit-Integrität)
 - Bestätigung per Email

US-035: Als Admin möchte ich Systemeinstellungen konfigurieren können

- Akzeptanzkriterien:
 - Session-Timeout konfigurierbar
 - Stornierungsfrist (Standard: 24h) konfigurierbar
 - Erinnerungs-Email Vorlaufzeit konfigurierbar
 - Min/Max Ausleihdauer konfigurierbar
 - SMTP-Server-Einstellungen
 - Änderungen werden im Audit-Log protokolliert

US-036: Als Admin möchte ich Statistiken exportieren können

- Akzeptanzkriterien:
 - Export als HTML oder PDF
 - Ausleihen pro Zeitraum
 - Top-Gegenstände nach Ausleihhäufigkeit
 - Frei wählbarer Zeitraum

US-038: Als Admin möchte ich Import-Logs einsehen können

- Akzeptanzkriterien:
 - Übersicht aller Import-Vorgänge
 - Zeitstempel, Anzahl importierter Gegenstände, Fehler
 - Detailansicht mit Fehlermeldungen

US-041: Als Admin möchte ich Budgets mit Tagessätzen verwalten können

- Akzeptanzkriterien:
 - Tagessätze pro Gegenstand definierbar
 - Gruppen haben ein Budget-Konto
 - Budget wird bei Ausleihe berechnet und abgezogen
 - Übersicht über Budget-Auslastung

US-042: Als Benutzer möchte ich private Gegenstände verwalten können

- Akzeptanzkriterien:
 - Eigenes Menü um private Gegenstände anzulegen

US-043: Als Verleiher möchte ich mit anderen Verleihern in Gruppen zusammenarbeiten

- Akzeptanzkriterien:
 - Verleihgruppen erstellen
 - Gegenstände der Gruppe zuordnen
 - Alle Gruppenmitglieder können Anfragen bearbeiten

US-044: Als Student möchte ich mit anderen Studenten eine Gruppe bilden

- Akzeptanzkriterien:
 - Studentengruppen erstellen
 - Gemeinsames Budget für die Gruppe
 - Alle Gruppenmitglieder sehen Gruppeneinschreibungen

US-045: Als Dozent möchte ich Ausleiheanfragen für Projekte freigeben

- Akzeptanzkriterien:
 - Studenten können Projekt angeben
 - Anfrage geht an Dozenten zur Freigabe
 - Nach Freigabe geht Anfrage an Verleiher
 - Workflow: Student → Dozent → Verleiher

US-046: Als Entwickler möchte ich eine CI/CD-Pipeline einrichten

- Akzeptanzkriterien:
 - Automatische Tests bei jedem Push
 - Automatisches Deployment bei Merge in Main-Branch
 - Code-Qualitätsprüfung mit SonarQube
 - GitHub Actions Integration

3 Zeitmanagement

In diesem Kapitel wird die zeitliche Umgebung beschrieben, in deren Rahmen sich das Projekt bewegt. Dazu zählen zuerst die Aufwandsschätzung in Stunden und außerdem die tatsächliche Zeiterfassung. Sowohl die Zeitplanung als auch die Zeiterfassung läuft über die Tabelle, die über diesen [Link](#) abrufbar ist. Das Vorgehen wird nachfolgend näher betrachtet.

Zur besseren Einordnung und Abwägung, welche Features implementiert werden sollen, wurde eine Aufwandsschätzung erstellt. Nachdem zuvor jedes Feature in Form User

Stories in seine Unteraufgaben aufgeteilt wurde, orientierte sich die Aufwandsschätzung an den User Stories. Es wurde trotz eingeschränkten Wissens über die zu verwendenden Technologien so gut wie möglich ein Zeitrahmen in Arbeitsstunden festgelegt, in dem mit der fertigen Implementierung des Features gerechnet wird. Darüber hinaus wurde eine pauschale Zeit festgelegt, um den Arbeitsaufwand, der zusätzlich zur Implementierung in Form beispielsweise der Dokumentation des Arbeitsschrittes anfällt, zu beschreiben. Die Aufwandsschätzung wurde außerdem von der reinen Textform in eine Tabelle überführt, um durch Verknüpfungen der Werte eine automatische Anpassung der Gesamtzeit bei der Änderung einzelner Werte zu ermöglichen. Die User Stories wurden nach der Sinnhaftigkeit ihrer Reihenfolge chronologisch sortiert und in einzelne Sprints unterteilt. Für jeden Sprint wurde in der Tabelle für eine bessere Übersichtlichkeit ein eigenes Tabellenblatt erstellt und die geschätzte Gesamtzeit für den Sprint anhand der in der Tabelle eingetragenen User Stories automatisch berechnet. Darüber hinaus wurde ein Tabellenblatt für einmalige Zeitaufwendungen angelegt. Dazu zählen vor allem die Vorbereitungen für Präsentationen und Meilenstein-Abgaben sowie Seminare. Im gleichen Stil wurde ein Tabellenblatt für wöchentlich wiederkehrende Zeitaufwendungen angelegt, in dem die entsprechende Zeit für beispielsweise Regelmeetings zur besseren Übersichtlichkeit einzeln geschätzt wird. Die Gesamtzeiten der einzelnen Tabellenblätter wurde dann in einem weiteren Tabellenblatt zu einer gesamten Zeitschätzung für das ganze Projekt zusammengerechnet.

3.1 Zeiterfassung

Für die Arbeitszeiterfassung wurde von Google Sheets auf Clockify umgestellt, da Clockify eine deutlich effizientere und integrierte Lösung bietet. Besonders wichtig war dabei die nahtlose Anbindung an Jira, über die Arbeitszeiten direkt auf Aufgaben und Tickets gebucht werden können, ohne manuelle Übertragungsfehler oder doppelten Aufwand. Darüber hinaus lässt sich Clockify nicht nur im Browser, sondern auch als Desktop- und Mobile-App nutzen, was die Erfassung unterwegs oder während Meetings erheblich vereinfacht und den gesamten Prozess flexibler und gebrauchstauglicher macht.

4 Projektmanagement

In diesem Kapitel wird beschrieben, wie das Management des Projekts aufgebaut ist. Ein besonderes Augenmerk liegt hierbei auf der Projektmanagement-Methode sowie auf der Organisation. Außerdem werden die in diesem Rahmen verwendeten Tools behandelt.

4.1 Methode

Es wird eine agile Projektmethode verwendet. Es wurde entschieden, ein Kanban Board zu verwenden, um die Aufgaben übersichtlich und variabel einzuteilen. Zusätzlich werden Elemente von Scrum übernommen. Hauptsächlich ist hierbei die Aufteilung in jeweils 2 Wochen langen Sprints zu nennen. Die Daily Meetings wurden durch ein wöchentliches Regelmeeting mit dem Betreuer und ein wöchentliches Meeting mit dem Team ersetzt. Um das Projekt schlank zu halten, wurde auf manche Scrum-Elemente wie die Sprint Reviews verzichtet.

4.2 Organisation

Die Kommunikation innerhalb des Teams läuft vor allem über den Teamchat und die wöchentlichen Team-Meetings. In den Team-Meetings werden die To-Dos festgelegt und verteilt. Die To-Dos werden auf dem Kanban-Board hinzugefügt. To-Dos aus dem letzten Meeting werden auf ihre Erfüllung überprüft. Außerdem werden offene Fragen geklärt und Grundsatzentscheidungen gefällt. Team-Meetings werden grundsätzlich von mindestens einer Person protokolliert. Jedes Teammitglied ist verantwortlich für die Erfüllung der eigenen To-Dos und den Status des To-Dos auf dem Kanban-Board. In den wöchentlichen Meetings mit dem Betreuer wird der Fortschritt des aktuellen Sprints besprochen und offene Fragen geklärt. Meetings mit dem Betreuer werden grundsätzlich von mindestens zwei Personen protokolliert. Protokolle von Meetings werden abgeglichen und das zusammengeführte Protokoll allen Teammitgliedern bereitgestellt.

4.3 Eingesetzte Tools

Es werden ausschließlich voll digitale, papierlose Tools verwendet. Als Kanban-Board wird Jira verwendet. Für den Teamchat und die wöchentlichen Teammeetings wird Discord eingesetzt. Die wöchentlichen Meetings mit dem Betreuer werden meist in Person abgehalten, falls dies nicht möglich ist, kommt Webex zum Einsatz. Die Dokumentation und die Meeting-Protokolle sowie Notizen und Ideen und organisatorische Informationen befinden sich auf HajTex, dem Online-LaTeX-Editor von fachschaften.org, um eine einfache und zeitgleiche Bearbeitung durch alle Teammitglieder zu ermöglichen. Die Zeiterfassung wird über das zentralisierte Online-Tool Clockify verwaltet.

5 Entwicklungsumgebung

In diesem Kapitel wird die technische Umgebung beschrieben, die für die Entwicklung der Fullstack Applikation LeihSy verwendet wurde.

5.1 Eingesetzte IDE

Für die Entwicklung der Anwendung wird die IDE IntelliJ IDEA Ultimate verwendet. Die IDE bietet umfangreiche Funktionen wie Syntax-Highlighting, Auto vervollständigung sowie integriertes Debugging und Git-Unterstützung. Des Weiteren unterstützt die IDE die Frontend- sowie die Backend-Entwicklung, wodurch die Entwicklung einer Fullstack-Applikation effizienter und einheitlich gestaltet werden kann.

5.2 Eingesetzte Frameworks

5.2.1 Backend Framework

Für die Backend-Entwicklungen der Applikation wird das Framework Spring-Boot verwendet. Spring-Boot ist ein Java-basiertes Framework und funktioniert nach dem Prinzip “Convention over Configuration”, was den Entwicklern bei den Konfigurations-Entscheidungen Zeit spart, weil das Framework vordefinierte Standardwerte verwendet. Des Weiteren unterstützt das Framework die Entwicklung von RESTful APIs, welche für die Datenübertragung zwischen Client und Server zuständig sind. Zudem ist Spring Boot weit verbreitet und es existieren umfangreiche Dokumentation zur Benutzung des Frameworks.

5.2.2 Weitere Backend-Bibliotheken

Neben Spring Boot werden im Backend weitere Bibliotheken eingesetzt, um die Entwicklung zu vereinfachen und den Quellcode übersichtlicher zu halten. Die Bibliothek Lombok reduziert wiederkehrenden Code, indem sie über einfache Annotationen automatisch Methoden wie Getter und Setter erzeugt. Dadurch bleibt der Code schlanker und besser lesbar, ohne dass auf Funktionalität verzichtet wird.

Die Bibliothek MapStruct wird verwendet, um Daten zwischen den JPA-Entitäten und den Data Transfer Objects (DTOs) zu übertragen. Anstatt die Felder manuell zu kopieren, werden die benötigten Mapper-Klassen aus Interfaces generiert. Das verringert die Fehleranfälligkeit, erleichtert Änderungen an den Datenklassen und unterstützt eine klare Trennung zwischen Persistenzschicht und API-Schicht.

5.2.3 Frontend Framework

Für die Frontend-Entwicklung der Applikation wird das Framework Angular verwendet. Angular ist ein TypeScript-basiertes Framework und bietet die Entwicklung einer komponentenbasierten Frontend-Architektur, welche die Anwendung gemäß des DRY-Prinzips in wiederverwendbare Komponenten unterteilt, was die Wartbarkeit und Effizienz

während der Entwicklung steigert. Architektonisch wird konsequent auf den Standalone-Komponenten-Ansatz gesetzt. Dies bedeutet, dass auf die klassische Modul-Struktur verzichtet wird, was die Anwendung modularer macht und so zur zuvor angesprochenen Wartbarkeit beiträgt. Für das Zustandsmanagement der Komponenten kommen Signals zum Einsatz. Diese ermöglichen eine reaktive Aktualisierung der Benutzeroberfläche, sobald sich Daten ändern. Zusätzlich dazu besitzt Angular ein integriertes Routing-System, welches die Navigation in der Applikation unterstützt und durch Routing-Guards einen Zugriffsschutz für bestimmte Seiten und Funktionen bereitstellt, durch Prüfung der Berechtigung des Nutzers.

5.2.4 Weitere Frontend-Bibliotheken

Neben Angular werden im Frontend weitere Bibliotheken bzw. Frameworks eingesetzt, um die Entwicklung zu vereinfachen und das Entwicklerteam zu entlasten. Die Bibliothek **PrimeNG** wird zur Erstellung der UI-Komponenten verwendet und ist mit dem modernen Aura-Theme konfiguriert. Dies stellt sicher, dass alle interaktiven Elemente wie Kalender, Dropdown-Menüs und Eingabefelder ein einheitliches, professionelles Design aufweisen.

Für das Gestalten des Layouts und die Feinjustierung des Designs wird **Tailwind CSS** genutzt. Dies ermöglicht ein vollständig responsives Design zu erstellen, das sich dynamisch an verschiedene Bildschirmgrößen (Desktop, Tablet, Smartphone) anpasst, ohne dass komplexe separate Stylesheets notwendig sind.

5.2.5 Lizenzen der eingesetzten Frameworks und Bibliotheken

Die verwendete Software muss, wie in Abschnitt 2.6 bereits erläutert, Open-Source sein und somit freie verfügbar und kommerzielle Verwendung erlauben, in der folgenden Tabelle 3 werden alle eingesetzten Frameworks oder Bibliotheken mit den entsprechenden Lizenz aufgelistet.

Komponente	Lizenz
Angular	MIT License
PrimeNG	MIT License
Tailwind CSS	MIT License
Spring Boot	Apache License 2.0
Lombok	MIT License
MapStruct	Apache License 2.0

Tabelle 3: Lizenzen der eingesetzten Frameworks und Bibliotheken

5.2.6 Integration in bereits bestehende Infrastruktur

Beide Frameworks werden bereits für das Inventarisierungssystem der Hochschule Esslingen, kurz InSy, verwendet und bieten somit eine gute Anschlussfähigkeit in die schon bestehende Infrastruktur.

5.3 Server

Die Web-Anwendung wird auf einem Server der bwCloud betrieben und dient dazu, die Applikation mittels Docker in einer containerisierten Umgebung bereitzustellen, wodurch neue Features einfacher integriert werden können. Des Weiteren existieren Instanzen, welche von der bereits bestehenden Infrastruktur genutzt werden.

5.4 Datenbank

Für die Speicherung und Verwaltung der Daten wird die Datenbank PostgreSQL verwendet. PostgreSQL ist ein Open-Source-Datenbankmanagementsystem, das leicht skalierbar und stabil ist. Es ermöglicht den Betrieb von relationalen Datenbanken und bearbeitet Transaktionen nach dem ACID Prinzip. Zur Visualisierung und Verwaltung der Daten wird pgAdmin verwendet, welches ebenfalls eine Open-Source-Software ist, welche speziell für PostgreSQL entwickelt wurde.

5.4.1 Entwicklungsumgebung mit H2

In der frühen Projektphase wurde für die lokale Entwicklung die In-Memory-Datenbank H2 eingesetzt. H2 ist eine leichtgewichtige, in Java geschriebene Datenbank, die keine separate Installation erfordert und bei jedem Anwendungsstart eine frische Datenbankinstanz erstellt. Dies ermöglichte uns, unabhängig voneinander zu arbeiten, ohne auf eine zentrale Datenbankinfrastruktur angewiesen zu sein.

Die H2-Konsole war während der Entwicklung unter `http://localhost:8080/h2-console` erreichbar und bot eine webbasierte Oberfläche zur direkten Inspektion der Datenbank. Dies war besonders hilfreich beim Debugging von JPA-Queries und der Überprüfung von Entity-Beziehungen.

Nachdem die PostgreSQL-Datenbank auf dem Entwicklungsserver eingerichtet und über VPN erreichbar war, wurde die Entwicklung vollständig auf PostgreSQL umgestellt. H2 wird seitdem nicht mehr aktiv verwendet, die Konfiguration bleibt jedoch als Fallback erhalten.

5.4.2 Spring Profiles

Um flexibel zwischen verschiedenen Datenbankumgebungen wechseln zu können, wurden Spring Profiles implementiert. Die Konfiguration ist auf drei Dateien aufgeteilt:

Datei	Profil	Beschreibung
application.properties	–	Basis-Konfiguration, legt aktives Profil fest
application-dev.properties	dev	H2 In-Memory-Datenbank für lokale Entwicklung
application-prod.properties	prod	PostgreSQL-Verbindung zum Entwicklungsserver

Tabelle 4: Übersicht der Konfigurationsdateien

Das aktive Profil wird in der Hauptkonfiguration festgelegt:

```
# application.properties
spring.profiles.active=prod
```

Alternativ kann das Profil beim Anwendungsstart als Parameter übergeben oder über Umgebungsvariablen gesetzt werden. Die profilspezifischen Konfigurationsdateien enthalten jeweils die Datenbankverbindungsparameter, Hibernate-Einstellungen und Logging-Konfiguration für die entsprechende Umgebung.

5.4.3 Datenbankverbindung

Die Verbindung zur PostgreSQL-Datenbank auf dem Entwicklungsserver erfolgt über ein VPN (WireGuard). Die Verbindungsparameter sind in `application-prod.properties` hinterlegt:

```
spring.datasource.url=jdbc:postgresql://<HOST>:5432/<DATABASE>
spring.datasource.username=<USERNAME>
spring.datasource.password=<PASSWORD>
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
```

Die Einstellung `ddl-auto=update` sorgt dafür, dass Hibernate das Datenbankschema automatisch an Änderungen in den Entity-Klassen anpasst, ohne bestehende Daten zu löschen. Für einen späteren Produktivbetrieb sollte diese Einstellung auf `validate` geändert werden, um unbeabsichtigte Schemaänderungen zu verhindern.

5.4.4 Automatische Testdaten

Für die Entwicklung mit H2 wurde ein `DataInitializer` implementiert, der beim Anwendungsstart automatisch Testdaten in die Datenbank einfügt. Die Klasse ist mit der Annotation `@Profile("dev")` versehen, sodass die Testdaten ausschließlich im Entwicklungsprofil geladen werden:

```

@Component
@Profile("dev")
public class DataInitializer implements CommandLineRunner {
    @Override
    public void run(String... args) {
        // Kategorien, Standorte, Produkte und Items anlegen
    }
}

```

Bei Verwendung des Produktionsprofils wird der `DataInitializer` nicht ausgeführt, sodass die PostgreSQL-Datenbank nicht mit Testdaten überschrieben wird. Dies stellt sicher, dass Entwicklungs- und Produktdaten strikt getrennt bleiben.

5.5 Development-Server

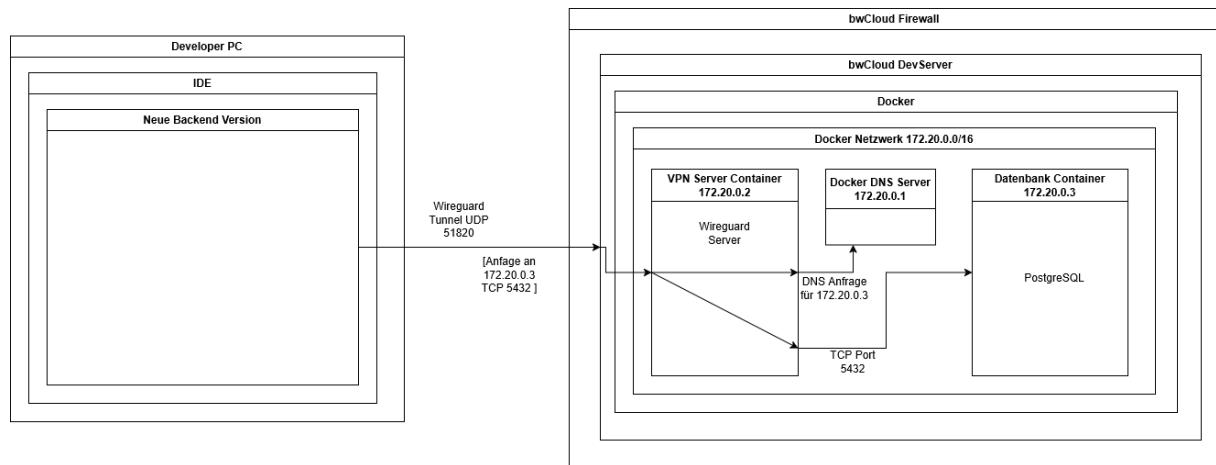


Abbildung 1: Netzwerkübersicht Development-Server

Wir haben für die Entwicklung einen eigenen Server eingerichtet, auf dem eine Datenbank läuft. Diese Datenbank ist mit Mock-Daten gefüllt, damit alle Entwickler mit denselben Informationen arbeiten können und die Ergebnisse vergleichbar bleiben.

Der Zugriff auf die Datenbank erfolgt nicht direkt, sondern über ein VPN, das wir mit WireGuard umgesetzt haben. WireGuard haben wir gewählt, weil es einfach einzurichten ist, zuverlässig funktioniert und eine sichere Verbindung bietet.

Man kann sich über WireGuard mit dem VPN verbinden. Sobald die Verbindung steht, können die Entwickler auf die Datenbank zugreifen und dort mit den vorbereiteten Mock-Daten arbeiten. So haben wir eine gemeinsame Grundlage geschaffen, die für alle gleich ist und die Entwicklung deutlich erleichtert.

Damit ist die Basis gelegt, also ein Server, eine Datenbank mit einheitlichen Testdaten und ein VPN für den sicheren Zugang. Mehr war für unseren aktuellen Stand nicht nötig, aber es sorgt dafür, dass wir stabil und konsistent arbeiten können.

Mit der Weiterentwicklung der Applikation wurde es notwendig auch Mock-Bilder Teamintern zu synchronisieren. Daher wurde im gleichen Docker Netzwerk ein Webdav-Container eingerichtet. Dieser stellt über HTTP ein Netzlaufwerk bereit das in Windows File Explorer als Netzlaufwerk hinzugefügt werden kann. HTTP ist in diesem Fall ausreichend da der Webdav-Server nur mit aktivem VPN erreichbar ist. Im Windows File Explorer kann dann eine Verknüpfung auf den Mockbilder-Ordner auf dem Webdav Netzwerk erstellt werden. Diese Verknüpfung kann in den lokalen Projektordner hinzugefügt werden. Somit hat die lokal laufende Applikation Zugriff auf die Mockbilder die auf dem Webdav-Laufwerk des Dev-Servers liegen.

5.6 Version Control management System

Für die Versionsverwaltung der Applikation wird Git verwendet, mit einem Repository auf GitHub. Dadurch werden Änderungen am Source-Code dokumentiert und eine Zusammenarbeit von mehreren Entwicklern organisiert. Außerdem kann durch das Erstellen von Branches für einzelne Feature und das Mergen durch eine Pull-Request der Fehleranfälligkeit entgegengewirkt werden.

5.6.1 CI/CD Pipeline

Zur Automatisierung des Entwicklungsprozesses wurde eine CI/CD Pipeline entworfen. In Abb. 2 ist der erste Entwurf der CI/CD-Pipeline zu sehen. Die Pipeline ist Pipeline wurde anfangs mit folgenden Stationen geplant:

1. Branch-Protection der Main, um zu verhindern das fehlerhafter Code gepushed und deployed wird
2. Durchführung von Lintingprozessen bei jedem Push auf eine beliebige Branch, der syntaktisch unsauberen Code anzeigen soll
3. Statische Codeanalysen mit SonarQube sowie Unit-Tests bei jedem Push auf eine beliebige Branch, zur Ermittlung der Code Qualität und Code Coverage
4. Falls ein Quality gate erreicht wurde, ist es möglich auf die Main zu Pushen, durch eine Pull-Request
5. Aus dem Code wird ein Docker Image erstellt, das auf ein Test-Enviroment deployed wird, in welchem Integrations-Tests durchgeführt werden, um die App als ganzes zu testen

- Falls alle Tests positiv durchlaufen wird das Image zur Registry hinzugefügt und ist bereit auf dem Production-Server zu laufen

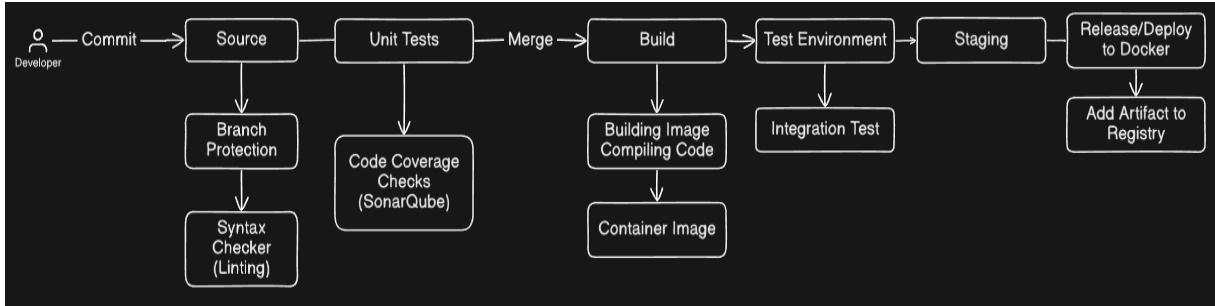


Abbildung 2: Alter Entwurf der CI/CD Pipeline Architektur in GitHub

Die Pipeline musste aufgrund der Diskrepanz des Entwurfs und der Möglichkeiten der Umsetzung angepasst werden. Die Probleme bei der Umsetzung des ersten Entwurfs werden im folgenden erläutert:

- Branch-Protection für private Repositories greift erst nach Erwerb von GitHub Premium; ohne Premium bleiben die Branches ungeschützt.
- Der eingesetzte Linter meldet Fehler bei der Verwendung von Komponenten oder Funktionen aus externen Frameworks und Bibliotheken (z. B. PrimeNG-Syntax).
- Die kostenlose Version von SonarQube führt statische Code-Analysen ausschließlich auf dem main-Branch sowie bei Pull Requests auf den main-Branch aus, jedoch nicht auf Feature-Branches.
- In der kostenlosen Version von SonarQube ist es nicht möglich, benutzerdefinierte Quality Gates zu erstellen.
- Die kostenlose Version von SonarQube erstellt keine automatischen Code-Coverage-Berichte.
- Der Aufwand für die Einrichtung und den Betrieb von Integrationstests auf einem dedizierten Testserver ist zu hoch.

Bei der finalen CI/CD-Pipeline (siehe Abb. 3)sind folgende Prozesse verblieben

1. Unit-Tests bei jedem Push auf eine beliebige Branch, zur Ermittlung der Code Coverage, welche durch Jacoco und Jasmin - Karma erstellt wird
2. Statische Codeanalysen mit SonarQube für die Main-Branch

3. Falls das Ergebnis positiv ist, wird das Docker-Image auf den Test-Server deployed

Zudem wurde SonarQube lokal in die IDEs hinzugefügt. Außerdem wurde eine manuelle Testinfrastruktur für die API-Endpunkte erstellt, die in Abschnitt 5.7 erläutert wird.

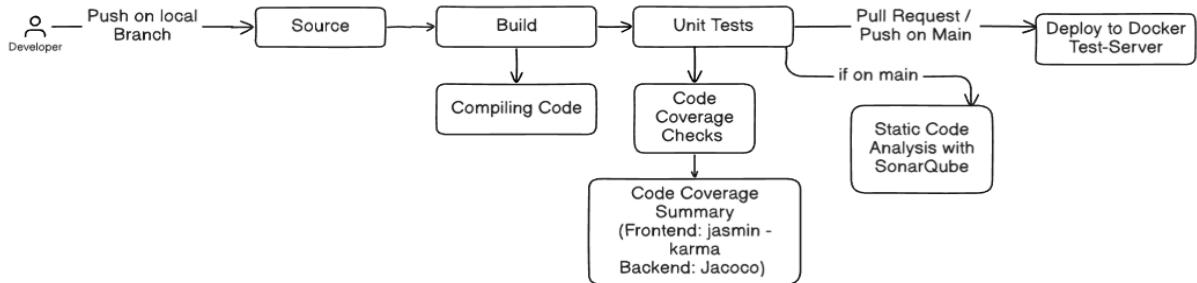


Abbildung 3: Finale CI/CD Pipeline in GitHub

Durch die CI/CD-Pipeline wird die Codequalität konstant gehalten und die Wartbarkeit der Software verbessert. Außerdem werden Fehler durch die Tests frühzeitig erkannt und können so schneller und effizienter behoben werden.

Das automatische Deployment als Docker-Container auf dem Testserver jeder neuen main Branch ist als Github Workflow realisiert. Dazu wurde in `/.github/workflows` eine yml-Datei angelegt die folgende Schritte durchläuft:

1. Der Prozess wird erst gestartet wenn die Code Coverage und SonarQube Tests erfolgreich durchlaufen wurden.
2. Im Repository sind SSH-Key, IP Adresse und User des Servers bei bwCloud als Secret hinterlegt. Diese werden verwendet um sich per SSH mit dem Server zu verbinden.
3. Dort wird in den Deployment-Ordner gewechselt und die aktuelle main Branch aus dem Github Repository gecloned.
4. Im Repository befindet sich ein Dockerfile, mit dem auf dem Server lokal ein Docker-Image gebaut wird. Dieses wird mit der ebenfalls im Repository enthaltenen docker-compose.yml als Container auf dem Server gestartet.
5. Dann wird durch eine Probeanfrage bestätigt dass der Container ordnungsgemäß in Betrieb ist.
6. falls diese Probeanfrage scheitert wird automatisch auf dem Server ein Rollback zur letzten funktionierenden Version der main Branch durchgeführt. Diese wird ebenfalls wieder gebaut und als Container gestartet so dass immer eine funktionierende Version der Anwendung auf dem Testserver zu Testzwecken bereit steht.

5.7 API-Testing mit Postman

5.7.1 Motivation

Für das systematische Testen der Backend-APIs wurde eine Postman-Collection erstellt. Die Herausforderung bestand darin, die Keycloak-Authentifizierung zu automatisieren, sodass nicht vor jedem Request manuell ein Token geholt werden muss.

5.7.2 Automatisches Token-Management

Ein Collection-weites Pre-request Script wurde implementiert, das folgende Aufgaben übernimmt:

- Prüfung ob ein gültiges Access Token in den Collection Variables vorhanden ist
- Validierung der Token-Gültigkeit anhand des Ablaufdatums (exp Claim)
- Automatisches Holen eines neuen Tokens von Keycloak bei Ablauf oder fehlendem Token
- Speicherung des Tokens und Ablaufdatums in Collection Variables für nachfolgende Requests

Das Script nutzt den OAuth2 Password Grant Flow (Resource Owner Password Credentials) mit dem Keycloak Token-Endpoint unter `/realms/insy/protocol/openid-connect/token`.

5.7.3 Collection Variables

Folgende Variablen wurden konfiguriert:

- `base_url`: Backend-URL (`http://localhost:8080/api`)
- `keycloak_url`: Keycloak-Server (`https://auth.insy.hs-esslingen.com`)
- `client_id`: Keycloak Client (temporär: insy-backend)
- `username`: Hochschul-Account des Test-Users
- `password`: Passwort des Test-Users
- `access_token`: Wird automatisch gefüllt vom Pre-request Script
- `token_expiration`: Unix-Timestamp des Token-Ablaufs

Alle Requests nutzen die Variable `{{access_token}}` im Authorization-Header mit Bearer-Schema.

5.7.4 Test-Requests

Die Collection enthält Requests für alle Booking-Endpoints:

- GET /bookings/users/me - Eigene Buchungen abrufen
- GET /bookings/lenders/me/pending - Offene Anfragen als Verleiher
- POST /bookings - Neue Buchung erstellen
- PUT /bookings/{id}/confirm - Buchung bestätigen mit Terminvorschlägen
- PUT /bookings/{id}/select-pickup - Abholtermin auswählen
- PUT /bookings/{id}/pickup - Ausgabe dokumentieren
- PUT /bookings/{id}/return - Rückgabe dokumentieren
- DELETE /bookings/{id} - Buchung stornieren

5.7.5 Vorteile

Dieses Setup ermöglicht effizientes Testing ohne manuelle Token-Verwaltung. Entwickler können die gesamte Booking-API mit wenigen Klicks durchspielen und den kompletten Workflow (Anfrage → Bestätigung → Ausgabe → Rückgabe) validieren.

6 UI-Prototyp

Der vollständige Prototyp ist direkt über den folgenden Figma-Link einsehbar: [Figma Dummy](#).

6.1 Farbpalette

6.1.1 Version 1 (Erstkonzept)

Schon in der ersten Version haben wir Grundfarben eingesetzt, um der Oberfläche ein stimmiges Erscheinungsbild zu geben. Zwar lag der Schwerpunkt noch auf der Struktur und dem Aufbau - also auf Suche, Listen und Detailseiten, aber eine schlichte Farbgestaltung war bereits vorhanden und wurde später beibehalten.

6.1.2 Version 2 (Wireframe)

Wir haben die Farben aus Version 1 übernommen und leicht verfeinert, um mehr Klarheit und Konsistenz zu schaffen. Dabei setzen wir auf neutrale UI-Farben:

- Primär: #0A0AOA

- Sekundär: #717182
- Ausgewählt: #E9EBEF
- Buttons: #000000
- Dazu die Statusfarben: #00A63E (verfügbar) und #C10007 (ausgeliehen).

6.1.3 Finale, farbige Version

In der finalen Version werden die hochschultypischen Farben verwendet:

- Primär: #012E58
- Sekundär: #32424A
- Ausgewählt: #32424A
- Buttons: #253359
- sowie wieder die Statusfarben: #00A63E (verfügbar) und #C10007 (ausgeliehen).

6.2 Erste Prototypen

6.2.1 Version 1 – Grundidee

- **Anmeldung & Rollen:** Eine einfache Anmeldemaske mit klarer Rollenauswahl (Studierende, Dozierende, Personal). (Siehe Abbildung 4)

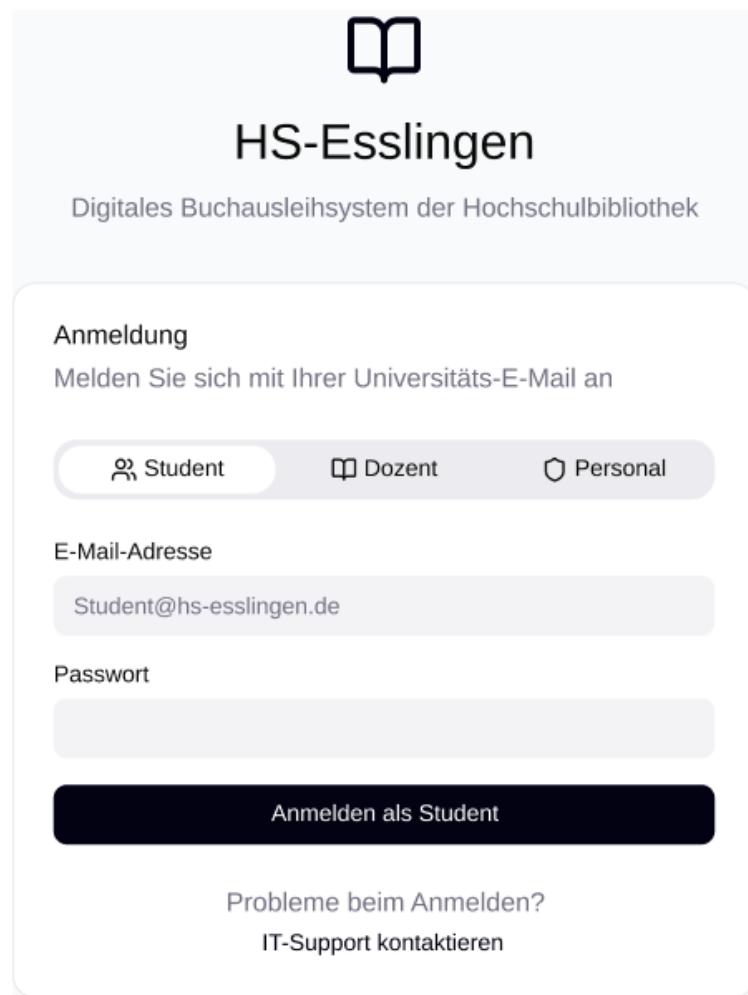
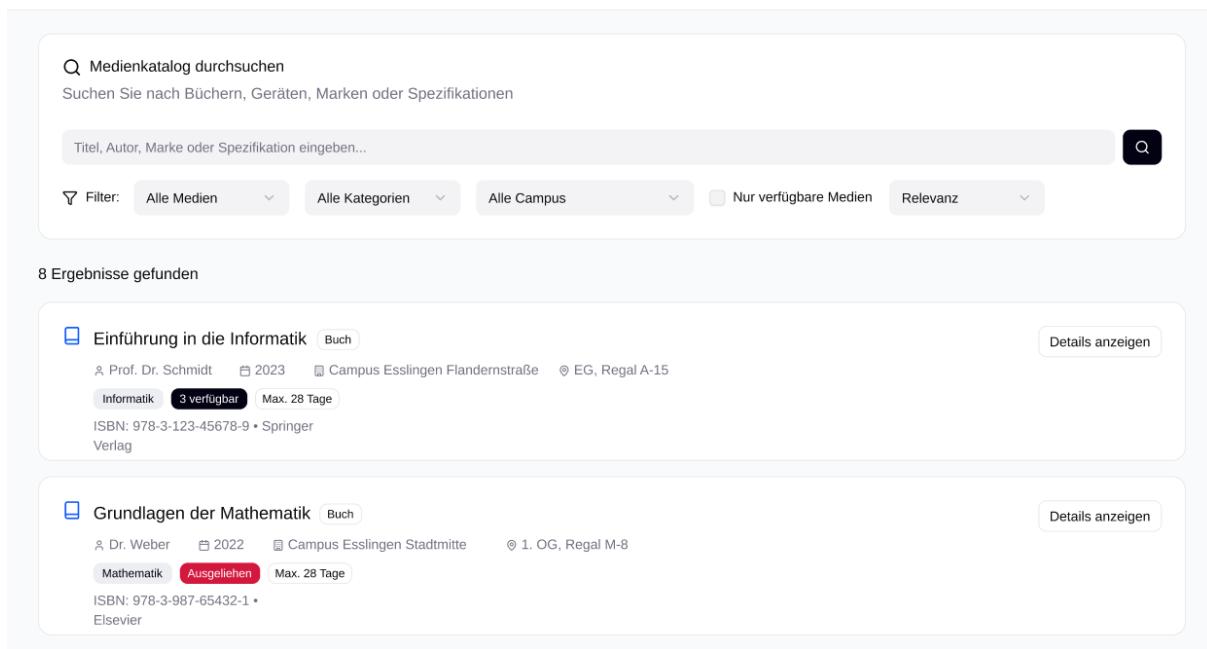


Abbildung 4: Login screen

- **Suche & Trefferliste:** Suche über Bücher und Geräte mit Filtern (Medientyp, Kategorie, Campus, Verfügbarkeit, Sortierung nach „Relevanz“). Die Trefferliste zeigt direkt Verfügbarkeit, Standort und maximale Ausleihdauer. (Siehe Abbildung 5)



The screenshot shows the HS-Esslingen Medienkatalog search interface. At the top, there is a search bar labeled "Q Medienkatalog durchsuchen" and a placeholder "Suchen Sie nach Büchern, Geräten, Marken oder Spezifikationen". Below the search bar are several filter options: "Filter:" dropdown set to "Alle Medien", "Alle Kategorien" dropdown, "Alle Campus" dropdown, "Nur verfügbare Medien" checkbox, and "Relevanz" dropdown. A search button with a magnifying glass icon is located to the right of the filters. Below the filters, it says "8 Ergebnisse gefunden". Two search results are displayed:

- Einführung in die Informatik** (Buch)
A Prof. Dr. Schmidt | 2023 | Campus Esslingen Flandernstraße | EG, Regal A-15
Informatik | 3 verfügbar | Max. 28 Tage
ISBN: 978-3-123-45678-9 • Springer Verlag
- Grundlagen der Mathematik** (Buch)
A Dr. Weber | 2022 | Campus Esslingen Stadtmitte | 1. OG, Regal M-8
Mathematik | Ausgeliehen | Max. 28 Tage
ISBN: 978-3-987-65432-1 • Elsevier

Each result has a "Details anzeigen" button to the right.

Abbildung 5: Inventarkatalog

- **Detailseite (Beispiel VR-Gerät):** Beschreibung, technische Spezifikationen, Zubehör, Verfügbarkeit je Campus, Ausleihmöglichkeit „Jetzt ausleihen“, Inventarnummer, Modell sowie die Ausleihbedingungen. Öffnungszeiten sind ebenfalls ersichtlich. (Siehe Abbildung 6)

The screenshot displays the HS-Esslingen library's device detail page for the Meta Quest 3 VR headset. At the top, there are navigation links for 'Katalog' and 'Mein Bereich', and an email address 'asesit00@hs-esslingen.de' for 'Student'.

Device Details:

- Name:** Meta Quest 3
- Type:** VR-Brille
- Model:** Meta Quest 3 128GB
- Status:** Verfügbar (Available)
- Description:** Hochmoderne VR-Brille für immersive virtuelle Erfahrungen. Ideal für Forschungsprojekte, Entwicklung und Lehrzwecke in den Bereichen Game Design, Medientechnik und Informatik.
- Technical Specifications:** 128GB Storage, 2064x2208 per eye, 90/120Hz
- Accessories:** Ladekabel, Reinigungstuch, Anleitung, Controller
- Keywords:** Virtual Reality, Immersive Technologie, Game Development, Medientechnik
- Number:** 1215
- Model:** Quest 3 128GB

Availability:

- Gesamt: 4 Exemplare
- Verfügbar: 2 Exemplare
- Ausgeliehen: 2 Exemplare

Borrowing: Campus: Campus Esslingen Flandernstraße, Standort: Medienausgabe, Schrank VR.1. A large button labeled 'Jetzt ausleihen' (Borrow now) is present.

Categorization: VR/AR (Main category)

Opening Hours:

- Wochentags:** Mo-Fr: 8:00-20:00
- Samstag:** Sa: 10:00-16:00
- Sonntag:** So: geschlossen (closed)

Borrowing Conditions:

- Ausleihzeit:** 7 Tage (7 days)
- Verlängerungen:** Nach Verfügbarkeit (After availability)
- Vormerkungen:** Bis zu 5 aktive Vormerkungen (Up to 5 active reservations)
- Hinweis:** Geräte müssen vollständig und funktionsfähig zurückgegeben werden (Devices must be returned fully functional and intact).

Contact Information:

- Öffnungszeiten:**
 - Campus Esslingen Flandernstraße: Mo-Fr: 8:00-20:00, Sa: 10:00-16:00, So: geschlossen
 - Campus Esslingen Stadtmitte: Mo-Fr: 9:00-18:00, Sa: 10:00-14:00, So: geschlossen
 - Göppingen: Mo-Fr: 8:30-19:00, Sa: 9:00-15:00, So: geschlossen
- Kontakt:**
 - Bibliothekspersonal: bibliothek@hs-esslingen.de, +49 711 397-49
 - Support & Probleme: bib-support@hs-esslingen.de, Bei Problemen mit Ausleihen, Rückgaben oder Ihrem Benutzerkonto
- Standorte:**
 - Campus Flandernstraße: Flandernstraße 101, 73732 Esslingen am Neckar
 - Campus Stadtmitte: Kanalstraße 33, 73728 Esslingen am Neckar
 - Campus Göppingen: Robert-Bosch-Straße 1, 79307 Göppingen
- Hochschulbibliothek:**
 - Die Bibliothek der Hochschule Esslingen bietet umfassende Literatur- und Informationsversorgung für Studium, Lehre und Forschung.
 - Medienbestand: Über 180.000 Medien
 - E-Books: Über 50.000 E-Books
 - Zeitschriften: 400+ Abonnements

At the bottom, a footer note states: © 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten. Datenschutz • Impressum • Barrierefreiheit.

Abbildung 6: Geräte-Details Seite

- **Mein Bereich:** Übersichtsseite mit aktuellen Ausleihen, Fälligkeiten, Gebühren und Verlängerungsmöglichkeit. (Siehe Abbildung 7)

HS-Esslingen
Hochschulbibliothek

Katalog Mein Bereich asesit00@hs-esslingen.de Student

Willkommen zurück!
Angemeldet als: asesit00@hs-esslingen.de (Student)

3 Aktuelle Ausleihen 3 Bald fällig € 3.50€ Offene Gebühren

⚠ Sie haben überfällige Medien mit ausstehenden Gebühren. Bitte geben Sie diese schnellstmöglich zurück.

Meine Ausleihen
Übersicht Ihrer aktuell ausgeliehenen Medien

Einführung in die Informatik Buch Prof. Dr. Schmidt
Ausgeleihen: 15.9.2025 Fällig: 15.11.2025 Bald fällig 325 Tage überfällig 0 Verlängerungen: 1/3

Meta Quest 3 VR-Brille Meta Quest 3 128GB
Ausgeleihen: 15.9.2025 Fällig: 15.11.2025 Bald fällig 359 Tage überfällig 0 Verlängerungen: 0/1

ThinkPad X1 Carbon Laptop Lenovo X1 Carbon Gen 11
Ausgeleihen: 15.9.2025 Fällig: 15.11.2025 Überfällig 363 Tage überfällig Gebühr: 3.50€ Verlängerungen: 1/2

Kürzlich zurückgegebene Medien Ihre letzten Rückgaben

Datenbanken verstehen Buch Prof. Dr. Fischer
Ausgeleihen: 1.8.2025 Zurückgegeben: 15.9.2025 Pünktlich zurückgegeben

Canon EOS R6 Mark II Kamera Canon EOS R6 Mark II
Ausgeleihen: 20.9.2025 Zurückgegeben: 27.9.2025 Pünktlich zurückgegeben

iPad Pro 12.9" Tablet Apple iPad Pro 12.9" M2
Ausgeleihen: 15.7.2025 Zurückgegeben: 20.8.2025 3 Tage

Kontoinformationen

Ausleihlimit 10 Medien
Ausleihzeit 7-30 Tage*
Aktuelle Ausleihen 3 von 10
Max. Verlängerungen Je nach Medientyp
* Bücher: 30/60/90 Tage • Geräte (VR, Kameras): 7/14/21 Tage • Laptops/Tablets: 14/21/28 Tage

Offene Gebühren 3.50€
Gebühren bezahlen

Öffnungszeiten Kontakt Standorte Hochschulbibliothek

Campus Esslingen Flandernstraße
Mo-Fr: 8:00-20:00
Sa: 10:00-16:00
So: geschlossen

Campus Esslingen Stadtmitte
Mo-Fr: 9:00-18:00
Sa: 10:00-14:00
So: geschlossen

Göppingen
Mo-Fr: 8:30-19:00
Sa: 9:00-15:00
So: geschlossen

Bibliothekspersonal
bibliothek@hs-esslingen.de
+49 711 397-49

Support & Probleme
bib-support@hs-esslingen.de
Bei Problemen mit Ausleihen, Rückgaben oder Ihrem Benutzerkonto

Campus Flandernstraße
Flandernstraße 101
73732 Esslingen am Neckar

Campus Stadtmitte
Kanalstraße 33
73728 Esslingen am Neckar

Campus Göppingen
Robert-Bosch-Straße 1
73037 Göppingen

Die Bibliothek der Hochschule Esslingen bietet umfassende Literatur- und Informationsdienste für Studium, Lehre und Forschung.
Medienbestand: Über 180.000 Medien
E-Books: Über 50.000 E-Books
Zeitschriften: 400+ Abonnements

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
Datenschutz Impressum Barrierefreiheit

Abbildung 7: Persönlicher Bereich

6.2.2 Version 2 – Änderungen gegenüber Version 1

- **Mehr Fokus:**

- Bücher wurden komplett entfernt. Das System ist nun klar auf die Geräteausleihe ausgerichtet – nicht auf die Bibliothek.

- **Schnellere Übersicht:**

- In der Übersicht sieht man die Verfügbarkeit pro Campus direkt auf der Karteansicht. (Siehe Abbildung 8)

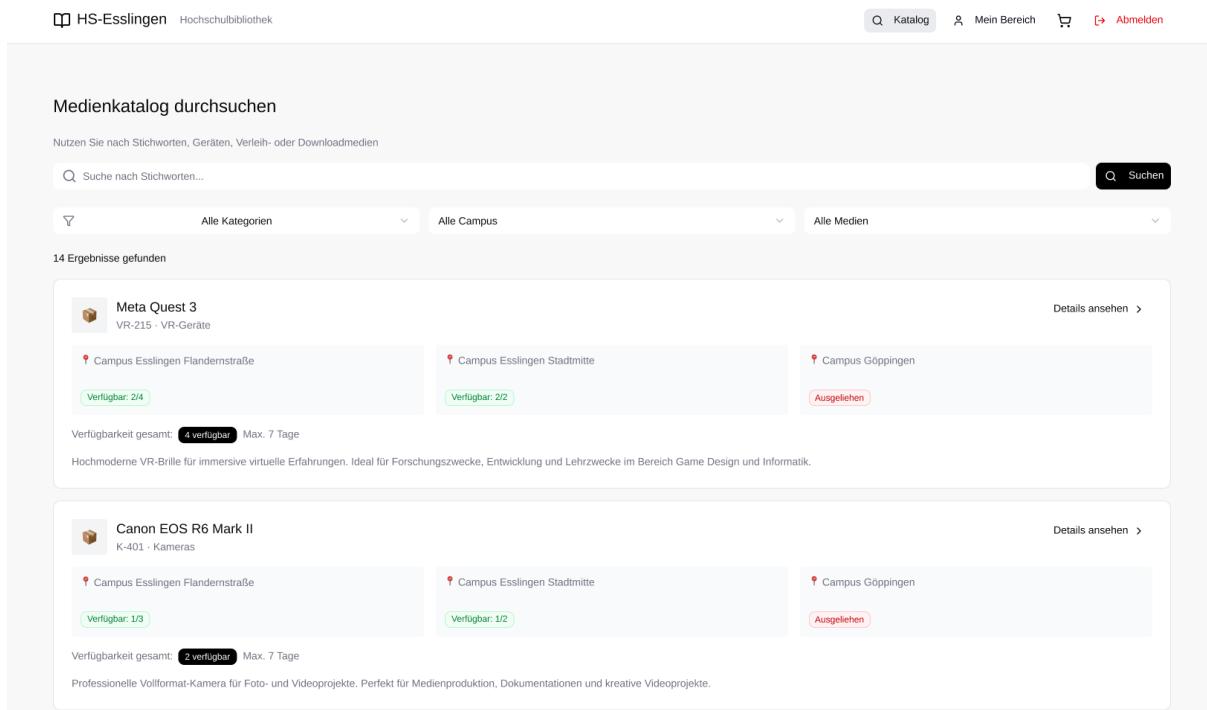


Abbildung 8: Aktualisierter Inventarkatalog

- **Gerätedetailseite sinnvoll erweitert**

- Auf Wunsch von dem Kunden wurde auf der Detailseite ein Kalender mit auswählbaren Zeitslots integriert. Nutzer können Datum und Zeitfenster direkt wählen und die Reservierung abschließen. (Siehe Abbildungen 9 und 10)

[Zurück zur Suche](#)

Meta Quest 3
VR-Geräte

Verfügbar
VR-215

Beschreibung

Hochmoderne VR-Brille für immersive virtuelle Erfahrungen. Ideal für Forschungszwecke, Entwicklung und Lehrzwecke im Bereich Game Design und Informatik.

Technische Spezifikationen

Speicher:	128GB
Sensor:	256x256 pixel eye, 90/120Hz
Zubehör:	Ladekabel Reinigungstuch Anleitung Controller

Enthaltenes Zubehör

Virtual Reality | Immersive Technologie | Game Development | Medientechnik

Schlagwörter

Virtual Reality | Immersive Technologie | Game Development | Medientechnik

Nummer: VR-215 · Modell: Meta Quest 3

Ausleihbedingungen

Ausleihzeit: 7 Tage
Verlängerungen: Nach Verfügbarkeit
Vormerkungen: Bis zu 5 aktive Vormerkungen
Hinweis: Bis zu 5 aktive Vormerkungen. Hinweis: Geräte müssen vollständig und funktionstüchtig zurückgegeben werden

Verfügbarkeit

Gesamt:

4 Exemplare verfügbar
4 Exemplare ausgeliehen

Campus:

Campus Esslingen Flandernstraße

Standort: Gebäude 01 - F 01.406

2/4 verfügbar

[In den Warenkorb](#)

Abholtermin wählen

Abholdatum:

November 2025

27	28	1	2	3	4
5	6	7	8	9	10
11	12	13	14	15	16
17	18	19	20	21	22
23	24	25	26	27	28
29	30	31	1	2	3

[Done](#)

Support

geraetelei@hs-esslingen.de
+49 711 397-3456
Für technische Probleme oder Fragen zur Ausleihe

Ausgabestellen

Medienausgabe Flandernstraße Raum F-301, 3. OG
Geräteausgabe Stadtmitte Gebäude 1. Erdgeschoss
IT-Ausgabe Göppingen Bibliothek, Raum G-104

LeihSy

Professionelles System für Studierende und Lehrende der Hochschule Esslingen.
VR-Geräte: 12+ Headsets
Kameras: 15+ Profi-Kameras
Lichtsets & Equipment: 15+ Sets

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
[Datenschutz](#) · [Impressum](#) · [Barrierefreiheit](#)

Abbildung 9: Kalender in Produktdetails

47

[Zurück zur Suche](#)

Meta Quest 3
VR-Geräte

Verfügbar VR-215

Beschreibung
Hochmoderne VR-Brille für immersive virtuelle Erfahrungen. Ideal für Forschungszwecke, Entwicklung und Lehrzwecke im Bereich Game Design und Informatik.

Technische Spezifikationen

Speicher:	128GB
Sensor:	256x256 pixel eye, 90/120Hz
Zubehör:	Ladekabel Reinigungstuch Anleitung Controller

Enthaltenes Zubehör

Virtual Reality Immersive Technologie Game Development Medientechnik

Schlagwörter

Virtual Reality Immersive Technologie Game Development Medientechnik

Nummer: VR-215 · Modell: Meta Quest 3

Verfügbarkeit

Gesamt:

4 Exemplare verfügbar

4 Exemplare ausgeliehen

Campus:

Campus Esslingen Flandernstraße

Standort:
Gebäude 01 - F 01.406

2/4 verfügbar

In den Warenkorb

Abholtermin wählen

Abholdatum:

13.11.2025

Zeitfenster:

Zeitfenster wählen

08:00 - 12:00 Uhr
01:00 - 17:00 Uhr

Kategorisierung

VR-Geräte

Support

✉ geraeteverleih@hs-esslingen.de
 ☎ +49 711 397-3456
 Für technische Probleme oder Fragen zur Ausleihe

Ausgabestellen

⌚ Medienausgabe Flandernstraße
 Raum F-301, 3. OG
 ⌚ Gerätelausgabe Stadtmitte
 Gebäude 1, Erdgeschoss
 ⌚ IT-Ausgabe Göppingen
 Bibliothek, Raum G-104

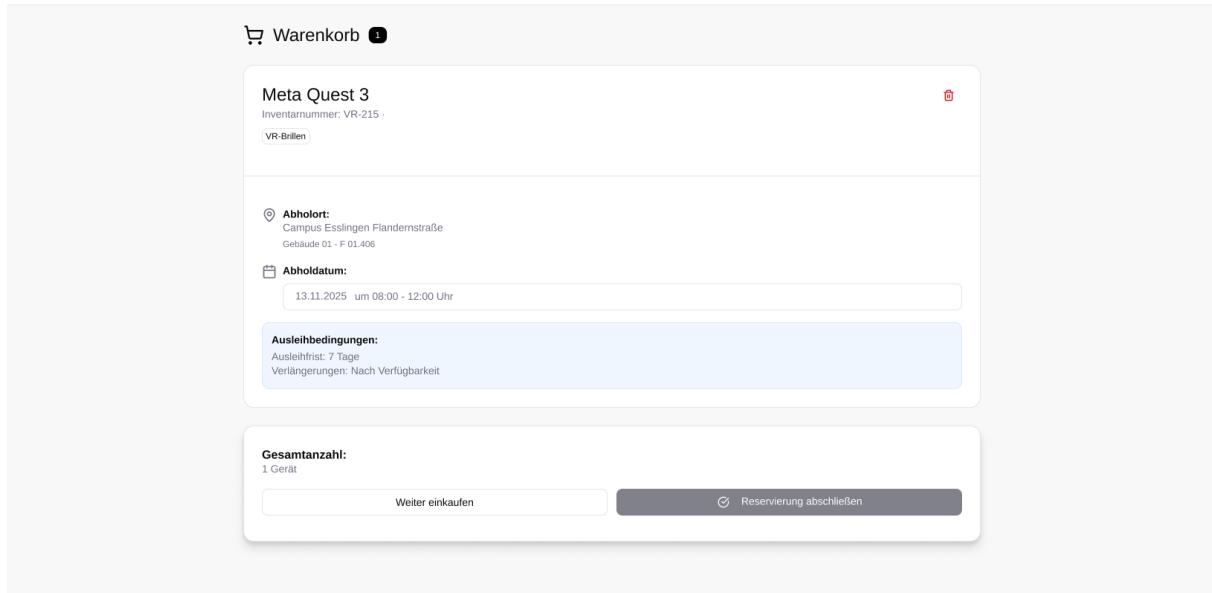
LeihSy
 Professionelles System für Studierende und Lehrende
 der Hochschule Esslingen.

⌚ VR-Geräte: 12+ Headsets
 ⌚ Kameras: 15+ Profi-Kameras
 ⌚ Lichtsets & Equipment: 15+ Sets

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
[Datenschutz](#) · [Impressum](#) · [Barrierefreiheit](#)

Abbildung 10: Abholungszeitfenster

- Auf Wunsch des Kunden wurde ein Warenkorb integriert. Darin können Nutzer ihre gewünschten Geräte direkt hinzufügen, einsehen und verwalten. (Siehe Abbildung 11)

**Support**

✉ geraeteverleih@hs-esslingen.de
 ☎ +49 711 397-3456
 Für technische Probleme oder Fragen zur Ausleihe

Ausgabestellen

- ⌚ Medienausgabe Flandernstraße Raum F-301, 3. OG
- ⌚ Geräteausgabe Stadtmitte Gebäude 1. Erdgeschoss
- ⌚ IT-Ausgabe Göppingen Bibliothek, Raum G-104

LeihSy

Professionelles System für Studierende und Lehrende der Hochschule Esslingen.
 ⓘ VR-Geräte: 12+ Headsets
 ⓘ Kameras: 15+ Profi-Kameras
 ⓘ Lichtsets & Equipment: 15+ Sets

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
[Datenschutz](#) · [Impressum](#) · [Barrierefreiheit](#)

Abbildung 11: Warenkorb

- Auf der Gerätedetailseite wird oberhalb des Buttons „In den Warenkorb“ der Standort angezeigt. So erkennen die Nutzer sofort, wo sie das gewählte Gerät erhalten können. (Siehe Abbildung 12)

[Zurück zur Suche](#)

Meta Quest 3
 VR-Geräte

Verfügbar VR-215

Beschreibung

Hochmoderne VR-Brille für immersive virtuelle Erfahrungen. Ideal für Forschungszwecke, Entwicklung und Lehrzwecke im Bereich Game Design und Informatik.

Technische Spezifikationen

Speicher:	128GB
Sensor:	256x256 pixel eye, 90/120Hz
Zubehör:	Ladekabel Reinigungstuch Anleitung Controller

Enthaltenes Zubehör

Virtual Reality | Immersive Technologie | Game Development | Medientechnik

Schlagwörter

Virtual Reality | Immersive Technologie | Game Development | Medientechnik

Nummer: VR-215 · Modell: Meta Quest 3

Verfügbarkeit

Gesamt:

4 Exemplare verfügbar
4 Exemplare ausgeliehen

Campus:

Campus Esslingen Flandernstraße

Standort:
Gebäude 01 - F 01.406
2/4 verfügbar

In den Warenkorb

Abholtermin wählen

Abholdatum:

13.11.2025

Zeitfenster:

Zeitfenster wählen
 08:00 - 12:00 Uhr
 01:00 - 17:00 Uhr

Kategorisierung

VR-Geräte

Support

geraeetelei@hs-esslingen.de
 +49 711 397-3456
 Für technische Probleme oder Fragen zur Ausleihe

Ausgabestellen

Medienausgabe Flandernstraße
 Raum F-301, 3. OG
 Geräteausgabe Stadtmitte
 Gebäude 1, Erdgeschoss
 IT-Ausgabe Göppingen
 Bibliothek, Raum G-104

LeihSy

Professionelles System für Studierende und Lehrende der Hochschule Esslingen.

VR-Geräte: 12+ Headsets
 Kameras: 15+ Profi-Kameras
 Lichtsets & Equipment: 15+ Sets

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
[Datenschutz](#) · [Impressum](#) · [Barrierefreiheit](#)

Abbildung 12: Standort des Geräts

- Der Bereich „Mein Bereich“ zeigt übersichtlich alle aktuellen Ausleihen, Reservierungen und offenen Gebühren. Zusätzlich können mehrere Geräte gleichzeitig verlängert werden. (Siehe Abbildung 13)

HS-Esslingen Hochschulbibliothek

Katalog Mein Bereich Abmelden

Willkommen zurück

3 Aktive Ausleihen

1 Reservierungen

3.50€ Offene Gebühren

Sie haben zurzeit überfällige Medien in Vorlage gehabt bzw. Rücknahme nicht erfolgt. Bitte geben Sie diese umgehend zurück.

Meine Ausleihen

Übersicht Ihrer aktuellen Ausleihen und Rückgabedaten

Sony A7 IV
Inventar-Nr.: K-512
Campus Esslingen Flandernstraße
Ausgeliehen am: 10.09.2025
Rückgabe bis: 17.09.2025
Verbleibende Zeit 0 Tage
0/1 Verlängerungen genutzt Verlängern Details

Meta Quest 2
Inventar-Nr.: VR-225
Campus Esslingen Flandernstraße
Ausgeliehen am: 10.10.2025
Rückgabe bis: 17.10.2025
Verbleibende Zeit 0 Tage
0/1 Verlängerungen genutzt Verlängern Details

Lichtset Aputure 300d II
Inventar-Nr.: L-450
Campus Esslingen Flandernstraße
Ausgeliehen am: 18.10.2025
Rückgabe bis: 25.10.2025
Verbleibende Zeit 5 Tage
0/1 Verlängerungen genutzt Verlängern Details

Warenkorb

Meta Quest 3
VR-215
Campus Esslingen Flandernstraße

Zur Ausleihe

Kontoubersicht

Aktive Ausleihen: 3
Kamera: 1
VR-Brillen: 1
Lichtset: 1
Offene Gebühren: 3.50€
Gebühren bezahlen

Kontoinformation

Matrikelnummer: 0768274
Gültig bis: 31.02.2026
Maximale Ausleihen: 20 Medien

Schnelle Verlängerungen

Verlängern Sie mehrere Medien gleichzeitig

Sony A7 IV
K-512
Meta Quest 2
VR-225
Lichtset Aputure 300d II
L-450

Ausgewählte verlängern

Zukünftige Abholungen

Ihre reservierten Geräte zur Abholung

Canon EOS R6 Mark II
K-401 - Campus Esslingen Flandernstraße
Abholung am 17.10.2024 - 10:00-14:00 Uhr
Bereit zur Abholung
Termin ändern

Support

geraeteverleih@hs-esslingen.de
+49 711 397-3456
Für technische Probleme oder Fragen zur Ausleihe

Ausgabestellen

Medienausgabe Flandernstraße
Raum F-301, 3. OG
Geräteausgabe Stadtmitte
Gebäude 1, Erdgeschoss
IT-Ausgabe Göppingen
Bibliothek, Raum G-104

LeihSy

Professionelles System für Studierende und Lehrende
der Hochschule Esslingen.
VR-Geräte: 12+ Headsets
Kameras: 15+ Profi-Kameras
Lichtsets & Equipment: 15+ Sets

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
Datenschutz · Impressum · Barrierefreiheit

Abbildung 13: Aktualisierter persönlicher Bereich

6.3 Visuelle Umsetzung der Funktionen

6.3.1 Login-Seite

Hier ist die neue Version der Login-Seite. Sie orientiert sich am Standard Design der Hochschule Esslingen. Die blaue Farbe und das Logo wurden bewusst gewählt, sodass die Seite nicht fremd wirkt. (Siehe Abbildung 14)

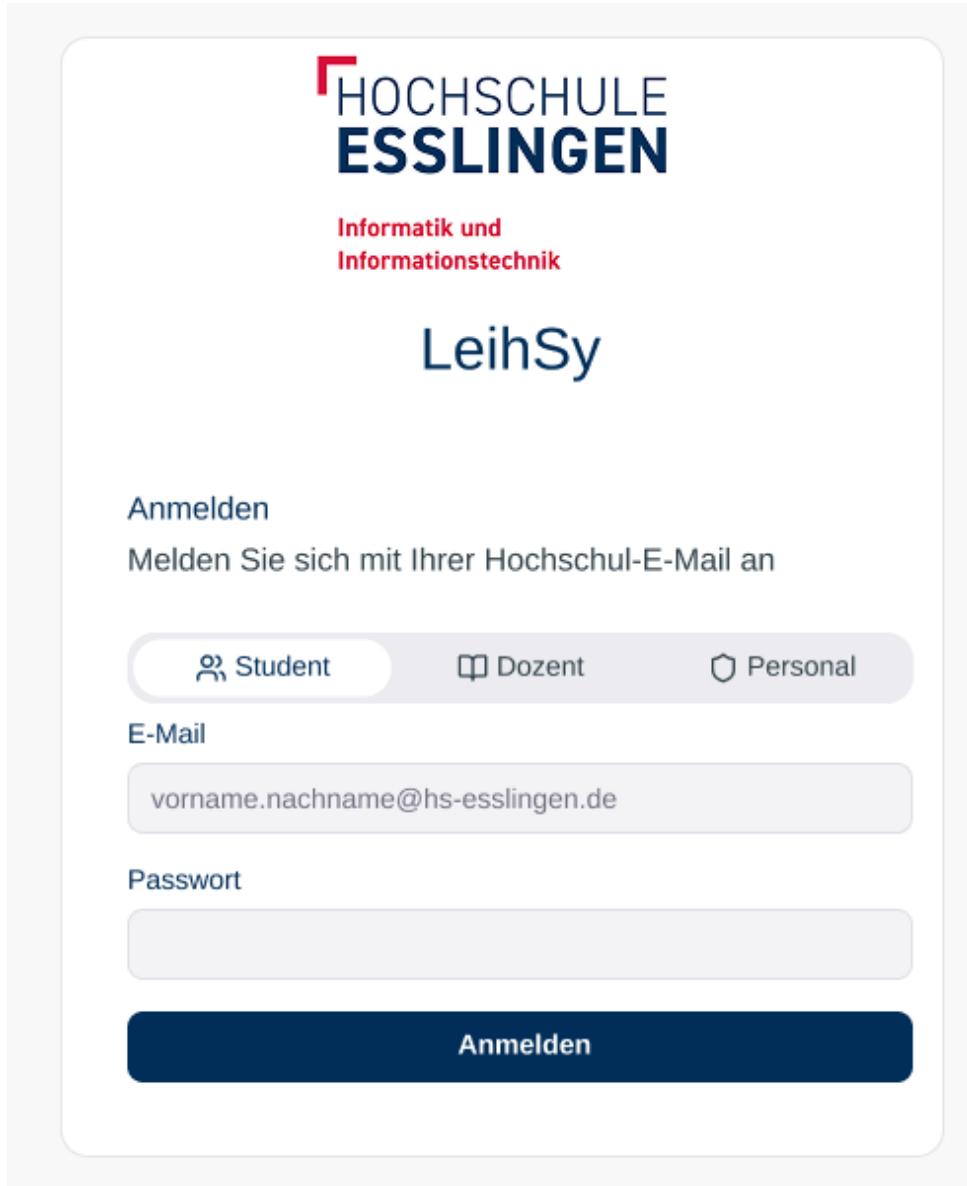


Abbildung 14: Aktualisierter Login screen

6.3.2 Geräte-Details

Oben links haben wir das Logo der Hochschule eingefügt, und die Farben wurden einheitlich angepasst. Die finale Version entspricht genau unserer Vorstellung. (Siehe Abbildung 15)

The screenshot displays the Hochschulbibliothek website's product detail page for a "Meta Quest 3 VR-Geräte".

Product Information:

- Name:** Meta Quest 3
- Type:** VR-Geräte
- Status:** Verfügbar (Available)
- Description:** Hochmoderne VR-Brille für immersive virtuelle Erfahrungen. Ideal für Forschungszwecke, Entwicklung und Lehrzwecke im Bereich Game Design und Informatik.
- Technical Specifications:**
 - Speicher: 128GB
 - Sensor: 256x256 pixel eye, 90/120Hz
 - Zubehör: Ladekabel, Reinigungstuch, Anleitung, Controller
- Included Accessories:** Virtual Reality, Immersive Technologie, Game Development, Medientechnik
- Keywords:** Virtual Reality, Immersive Technologie, Game Development, Medientechnik
- Notes:** Nummer: VR-215 - Modell: Meta Quest 3

Lending Conditions:

- Ausleihzeit: 7 Tage
- Verlängerungen: Nach Verfügbarkeit
- Vormerkungen: Bis zu 5 aktive Vormerkungen
- Hinweis: Bis zu 5 aktive Vormerkungen. Hinweis: Geräte müssen vollständig und funktionstüchtig zurückgegeben werden

Availability:

- Gesamt: 4 Exemplare verfügbar
- 4 Exemplare ausgeliehen
- Campus: Campus Esslingen Flandernstraße
- Standort: Gebäude 01 - F 01.406
- 2/4 verfügbar

Buttons:

- In den Warenkorb (Add to Cart)
- Abholtermin wählen (Select pick-up date)
- Zeitfenster wählen (Select time window)
- Termin reservieren (Reserve time)
- Kategorisierung (Categorization)
- VR-Geräte (VR Devices)

Support	Ausgabestellen	LeihSy
<p>E-Mail: geraeteverleihs@hs-esslingen.de Telefon: +49 711 397-3456 Für technische Probleme oder Fragen zur Ausleihe</p>	<p>Medienausgabe Flandernstraße Raum F-301, 3. OG</p> <p>Geräteausgabe Stadtmitte Gebäude 1, Erdgeschoss</p> <p>IT-Ausgabe Göppingen Bibliothek, Raum G-104</p>	<p>Professionelles System für Studierende und Lehrende der Hochschule Esslingen.</p> <p>VR-Geräte: 12+ Headsets Kameras: 15+ Profi-Kameras Lichtsets & Equipment: 15+ Sets</p>

© 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten.
Datenschutz · Impressum · Barrierefreiheit

Abbildung 15: Geräte-Details Seite mit aktualisierten Design

6.3.3 Warenkorb

Wir haben das Warenkorb Logo so gestaltet, dass oben die Anzahl der hinzugefügten Geräte in Nummern dargestellt wird. Außerdem haben wir den Bereich für Terminreservierungen in Rot dargestellt, damit er sofort ins Auge fällt und die Aufmerksamkeit des Betrachters auf sich zieht. (Siehe Abbildung 16)



Verfügbarkeit

Gesamt:

4 Exemplare verfügbar

4 Exemplare ausgeliehen

Campus:

Campus Esslingen Flandernstraße

Standort:
Gebäude 01 - F 01.406

2/4 verfügbar

In den Warenkorb

Abholtermin wählen

Abholdatum:

13.11.2025

Zeitfenster:

08:00 - 12:00 Uhr

Termin reserviert

Abbildung 16: Verbesserter Warenkorb

7 User Research

7.1 Proto-Personas

7.1.1 Persona A – Lena Schmid (Studierende)

- **Profil:** Bachelor Medieninformatik, organisiert mobil am Smartphone, wenig Geduld für Papier & Rückfragen.
- **Ziele:** schnell verfügbare Geräte finden, Slots planen, transparente Status-/Historienansicht.
- **Bedürfnisse:** Online-Reservierung mit klarer Live-Verfügbarkeit, automatische Bestätigungen & Erinnerungen, einfache Verlängerung, transparente Regeln/Gebühren.
- **Pain Points:** unklare Verfügbarkeit, Wartezeiten/keine Antwort → wünscht Auto-Storno, fehlende Erinnerungen → Überfälligkeit.



ALTE 26
BERUF Medieninformatik Studentin

USER PERSONA
Lena Schmid

LENA ist eine Medieninformatik Studentin im Bachelor und arbeitet nebenbei als Werkstudentin. Sie organisiert vieles unterwegs am Smartphone, hasst Papierkram und unnötige Rückfragen. Wichtig sind ihr klare Verfügbarkeiten, schnelle Bestätigungen/Erinnerungen und transparente Statusanzeigen. Wenn niemand reagiert, erwartet sie automatisches Storno statt langem Hinterherlaufen.

ZIELE	BEDÜRFNISSE
<ul style="list-style-type: none">• Schnell sehen, was verfügbar ist & einfach anfragen• Zeitfenster planen & bei Konflikten klare Hinweise• Transparenz über Status & Historie	<ul style="list-style-type: none">• Vorgänge schnell und unkompliziert online erledigen, statt Zettel auszufüllen oder manuell bei der dem Professor_in einzureichen.• Klare Übersicht über verfügbare Zeiten und Ressourcen, um ohne Rückfragen planen zu können.
PAIN POINTS	
<ul style="list-style-type: none">• Unklare Verfügbarkeit/Konflikte• Lange Wartezeiten/keine Antwort → Auto-Storno hilft• Fehlende Erinnerungen → Überfälligkeit	<ul style="list-style-type: none">• Automatische Bestätigungen und Erinnerungen, damit keine Anfragen vergessen oder überfällig werden.• Transparente Nachverfolgung des eigenen Status und früherer Anfragen, ohne extra nachfragen zu müssen.

Abbildung 17: Persona Lena Schmid

7.1.2 Persona B – Max Schmidt (IT-Admin / Systemverantwortlicher)

- **Profil:** IT-Administrator (Hochschule), prozess- und sicherheitsorientiert.
- **Ziele:** Benutzer/Rollen zentral steuern, Inventar & Sets pflegen, geringe Supportlast.

- **Bedürfnisse:** SSO (Hochschul-Account), klares Rechte, sauberes Protokoll, Medienmanagement.
- **Pain Points:** Datenmüll, mangelnde Nachvollziehbarkeit, aufwändige Pflege.

USER PERSONA

Max Schmidt

Max Schmidt ... ist ein zielorientierter IT-Administrator, der an einer Hochschule arbeitet. Er hat eine Leidenschaft für Technologie, Sicherheit und stabile IT-Systeme. Trotz der oft komplexen und stressigen Aufgaben im Hochschul-IT-Bereich achtet Max darauf, den Überblick zu behalten und seine Projekte effizient umzusetzen. Neben seiner Arbeit interessiert er sich für neue Softwarelösungen, Automatisierung und die Optimierung von Prozessen, um die digitale Infrastruktur der Hochschule zu verbessern.

ALTE	55
BERUF	IT-Administratorin / Systemverantwortliche für das Hochschul-Leihsystem

ZIELE

- Benutzer, Rollen & Berechtigungen zentral steuern
- Inventar anlegen/bearbeiten/sets pflegen, Bilder managen
- Gute Nutzererfahrung für alle Anwender
- Weniger Supportaufwand durch einfache Bedienung

BEDÜRFNISSE

- Single Sign-On (SSO): Nutzer sollen sich mit ihrem Hochschul-Account anmelden können – ohne zusätzliche Logins.
- Rechte- und Rollenmanagement: Klare Zugriffsebenen für Admins, Mitarbeitende und Studierende.
- Inventar-Formulare, Set-Erstellung mit Auto-Nummerierung

PAIN POINTS

- Datenmüll/Dubletten → Single Source (Keycloak), sparsame Datenspeicherung
- Nachvollziehbarkeit → unveränderbares Protokoll
- Medienmanagement → Größen/Formate/Thumbnails

Abbildung 18: Persona Max Schmidt

7.1.3 Persona C – Prof. Tom Fischer (Lehrender/Verleiher)

- **Profil:** Professor/Verantwortlicher für Leihgeräte (z. B. VR).
- **Ziele:** Anfragen schnell prüfen, saubere Ausgabe/Rückgabe vor Ort, Überblick über eigene Ressourcen.
- **Bedürfnisse:** Dashboard offener Anfragen mit Filter, Vorschläge für Alternativtermine, E-Mail-Trigger für Abholung/Rückgabe, zentrale Plattform (ideal integrierbar in Hochschul-Umgebung).
- **Pain Points:** viele Anfragen gleichzeitig → Priorisierung schwer, keine klare Verfügbarkeit, kein Überblick über bestehende Reservierungen.

USER PERSONA

Tom Fischer

Herr Prof. Fischer ... ist ein zielorientierter Professor an einer Universität in München, der sich leidenschaftlich für Technologie, Innovation und Nachhaltigkeit einsetzt. Sein Ziel ist es, den Hochschulalltag umweltfreundlicher und effizienter zu gestalten – durch den Einsatz digitaler Lehrmethoden, energieeffizienter Prozesse und nachhaltiger Campuslösungen.

ALTE	55
BERUF	Professor

ZIELE

- Schnell prüfen & entscheiden: Anfragen annehmen/ablehnen/Termine abstimmen
- Eigene Gegenstände im Blick behalten
- Vor Ort: Ausgabe/Rückgabe sauber dokumentieren
- Nachhaltigkeit fördern: Ressourcenverbrauch senken

BEDÜRFNISSE

- Dashboard offene Anfragen + Filter/Sortierung
- Alternativtermine vorschlagen
- E-Mail-Trigger für Abholung, Rückgabe
- Zentrale Plattform: z. B. Integration in die Hochschul-App

PAIN POINTS

- Zu viele Anfragen gleichzeitig → schwer zu priorisieren oder zu überblicken
- Unklare Verfügbarkeiten → es ist nicht ersichtlich, wann ein Gerät wirklich frei ist
- Kein Überblick über bestehende Reservierungen → führt zu Doppelbuchungen oder Leerlaufzeiten

Abbildung 19: Persona Tom Fischer

7.2 Interview Auswertung

Für das Projekt „Leihsy“ wurden Studierende der Hochschule interviewt. Dabei wurde Ihnen unser UI-Prototyp gezeigt, und es wurden folgende Fragen gestellt.

- „Finde eine VR-Brille, die am Campus Flandernstraße verfügbar ist.“
- „Ist klar, wo das Gerät abgeholt wird (Standort)?“
- „Ist *Bald fällig* vs. *Überfällig* eindeutig?“
- „Wenn du eine Sache ändern könntest, was zuerst?“
- 1–5-Rating: „Wie einfach war es, X zu erledigen?“

Bisher sind alle Studierenden gut mit dem UI-Prototyp zurechtgekommen und konnten alle wichtigen Funktionen problemlos finden. Die Farbgestaltung wurde als passend empfunden, und auf den ersten Blick war alles klar erkennbar. Daher haben wir in unserem Ranking 4,5 von 5 Punkten erhalten.

Der halbe Punkt Abzug ergibt sich aus der Darstellung des Raums für die Geräteabholung. Da über dem Raum der Campus mit einem Standort-Emoji angezeigt wird, war es den Studierenden nicht sofort klar, wo genau sich der Raum befindet. Das Emoji hat die Aufmerksamkeit vom Raum leicht abgelenkt.

7.3 Auswertung (Online-Umfrage, n = 28)

7.3.1 Stichprobe & Nutzung

- In den letzten 24 Monaten ausgeliehen: 6 (21 %); nicht ausgeliehen: 22 (79 %).
- Genannte Kategorien (Mehrfachauswahl): VR-Brillen 4, Sonstiges 2, Kamera 0.

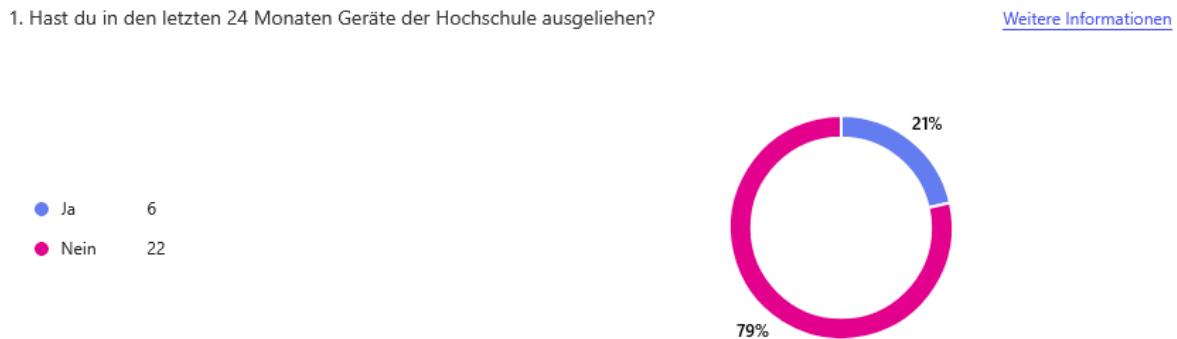


Abbildung 20: Anteil Studenten mit kürzlichen Ausleihen

7.3.2 Zufriedenheit (Ist-Prozess)

Ø-Bewertung: 3,00 (mittelmäßig) → klarer Verbesserungsbedarf.

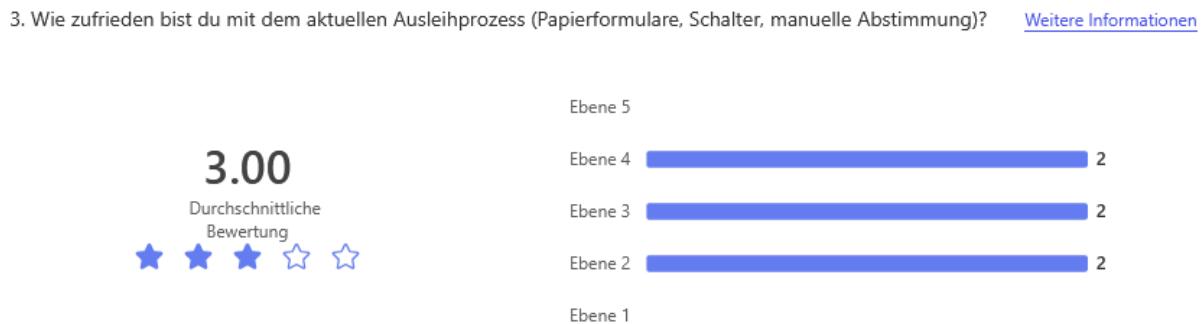


Abbildung 21: Zufriedenheit mit aktuellem Prozess

7.3.3 Größte Pain Points (Top-Nennungen)

- Formulare/Unterschriften 5
- Unklare Verfügbarkeit 3

- Rückgabe unflexibel 3
- Abholung nicht planbar 2

⇒ Papier & manuelle Abstimmung sind die Hauptbremser; Echtzeit-Infos & flexible Slots fehlen.

4. Was nervt dich am meisten?

[Weitere Informationen](#)

● Unklare Verfügbarkeit	3
● Formulare/Unterschriften	5
● Abholung nicht planbar	2
● Rückgabe unflexibel	3
● Sonstiges	0

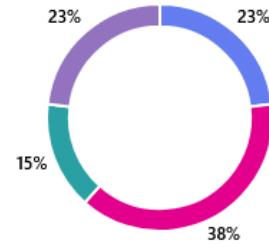


Abbildung 22: aktuelle Pain Points

7.3.4 Adoptionsbereitschaft für ein digitales System

- Ja 13 + Eher ja 12 → ~89 % positiv.
- Unsicher 2 (~7 %), Nein 1 (~4 %).

⇒ Die hohe Nutzungsintention spricht klar für die Umsetzung.

5. Wenn es ein zentrales, digitales System gäbe (Suchen, Verfügbarkeit je Campus, Online-Reservierung, Abhol-/Rückgabe-Slots): würdest du es nutzen?

[Weitere Informationen](#)

● Ja	13
● Eher ja	12
● Unsicher	2
● Eher nein	0
● Nein	1

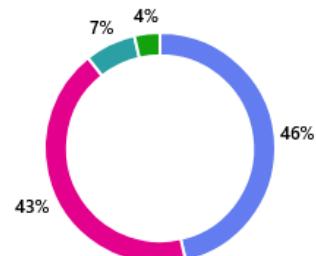


Abbildung 23: Adoptionsbereitschaft für digitales System

7.3.5 Top-Features (max. 3 Stimmen)

- Erinnerungen vor Rückgabe – 17

- Live-Verfügbarkeit – 16
- Online-Verlängerung – 12
- Standort-Filter (Campus) – 11
- Maximale Ausleihdauer sichtbar – 8
- Transparentes Gebühren-/Zubehör-Listing – 7

6. Was wäre dir dabei am wichtigsten? (max. 3 auswählen)

[Weitere Informationen](#)

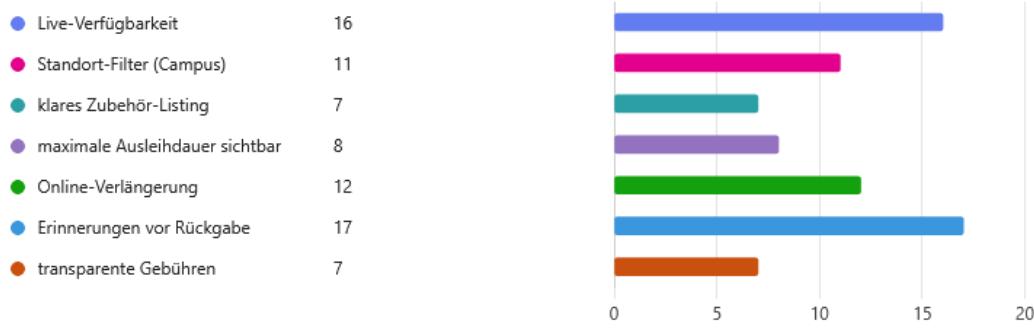


Abbildung 24: Wichtigste Features

7.3.6 Unverzichtbare Filter

- Nur verfügbare Geräte – 19 (Must-Have)
- Maximale Ausleihdauer – 5
- Campus – 4

7. Welche Filter wären für dich unverzichtbar? (Mehrfachauswahl möglich)

[Weitere Informationen](#)



Abbildung 25: Unverzichtbare Filter

7.3.7 Verfügbarkeit je Campus

- sehr hilfreich – 14 (50 %)
- eher hilfreich – 10 (35,7 %)
- eher nicht hilfreich – 3 (10,7 %)
- gar nicht hilfreich – 1 (3,6 %)

⇒ Erkenntnis: Insgesamt 85,7 % der Befragten empfinden die Anzeige als hilfreich. Die Funktion gilt somit als klarer Mehrwert und sollte im System standardmäßig integriert werden.

8. Wie hilfreich ist die Anzeige „Verfügbarkeit je Campus“ für deine Planung?

[Weitere Informationen](#)

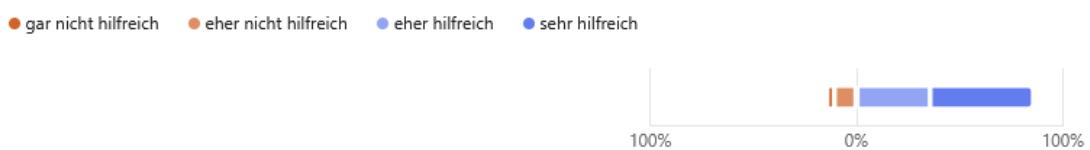


Abbildung 26: Wichtigkeit Verfügbarkeit nach Campus

7.3.8 Storno-Fenster (Abhol-/Rückgabe-Slots)

- bis 24 h vorher – 14 (50 %)
- bis 12 h vorher – 8 (29 %)
- bis 5 h vorher – 6 (21 %)

⇒ Empfehlung: Standard 24 h, ggf. 12 h für „Kurzfristig“.

9. Wie kurzfristig sollen Abhol-/Rückgabe-Slots stornierbar sein?

[Weitere Informationen](#)



Abbildung 27: Länge Stornierbarkeit

8 Softwarearchitektur

8.1 Entity-Relationship-Diagramme

8.1.1 Erster Entwurf

Das in Abbildung 28 dargestellte Entity Relationship Diagramm zeigt unseren ersten Entwurf der Datenbank Strukturierung

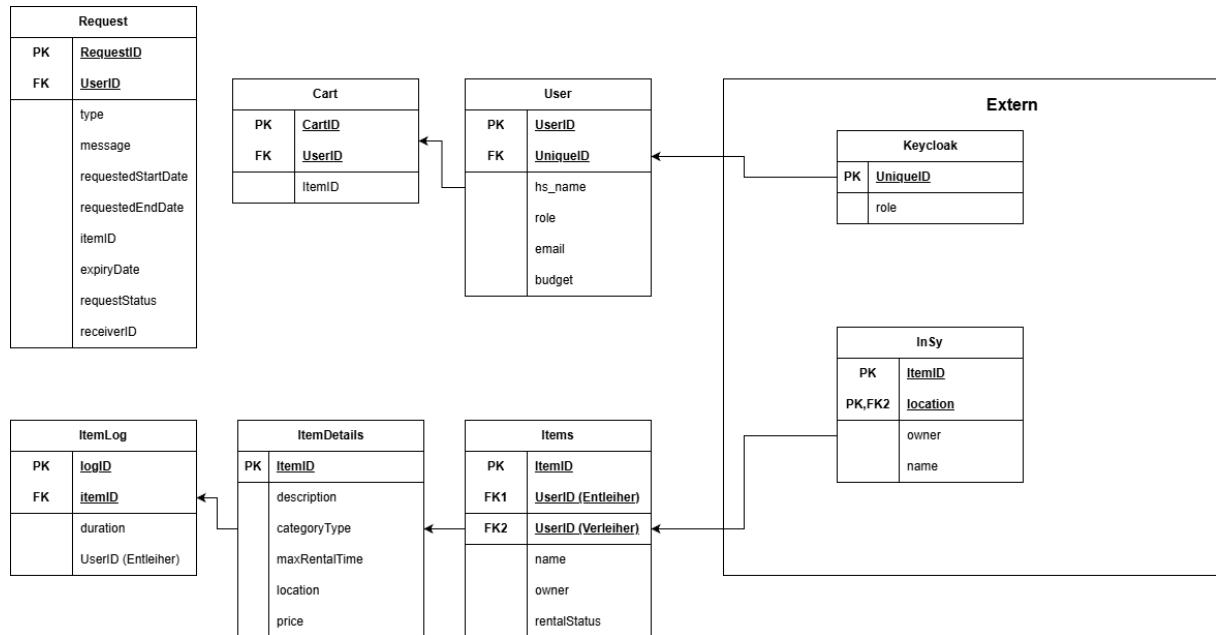


Abbildung 28: Erster Entwurf Entity-Relationship-Diagramm

8.1.2 Änderungen vom ersten Entwurf zur finalen Version

Im Vergleich zum in Abbildung 28 dargestellten ersten Entwurf wurde das Entity-Relationship-Modell in mehreren Bereichen strukturell überarbeitet, normalisiert und an die funktionalen Anforderungen des Systems angepasst. Die wichtigsten Änderungen lassen sich wie folgt zusammenfassen:

8.1.2.1 1. Umbenennung zentraler Entitäten

- Die Entität **Request** wurde durch die Entität **Bookings** ersetzt. Dabei wurden Attribute wie Start-, End- und Rückgabedatum, Status und Nachrichten in ein einheitliches Buchungsmodell überführt.
- Die frühere Entität **User** wurde zu **Users** umbenannt und um Historisierungsattribute (`created_at`, `updated_at`, `deleted_at`) erweitert.

- Die Beziehung zur externen Authentifizierung (`Keycloak`) wurde neu modelliert und verwendet nun einen expliziten Primärschlüssel anstelle der im Entwurf genutzten `UniqueID`-Referenz.

8.1.2.2 2. Neuorganisation der Produkt- und Itemstruktur

- Die im Entwurf bestehenden Entitäten `Items`, `ItemDetails`, `ItemLog` und `InSy` wurden vollständig restrukturiert.
- Die Entität `Products` ersetzt die zuvor getrennten Entitäten `Items` und `ItemDetails` als übergeordnete Produktbeschreibung.
- Physische Einheiten eines Produkts wurden in eine neue, klar abgegrenzte Entität `Items` ausgelagert, die nur produkt- und inventarspezifische Attribute enthält.
- Attribute wie `categoryType`, `maxRentalTime`, `price` und `location` wurden in die Produkt- bzw. Standortentitäten überführt und dort normalisiert.

8.1.2.3 3. Einführung neuer Entitäten

- **Categories:** Eine neue Entität zur Klassifizierung von Produkten, die im Entwurf lediglich als Attribut existierte.
- **Locations:** Eine eigene Entität zur räumlichen Zuordnung der Produkte; ersetzt das frühere einfache Attribut `location`.
- **Sets:** Neue n:m-Assoziation zur Gruppierung mehrerer Produkte zu vordefinierten Sets.

8.1.2.4 4. Beziehungsanpassungen und Normalisierung

- Die im Entwurf teilweise unklaren oder nicht normalisierten Beziehungen wurden zu eindeutigen 1:n- oder n:m-Beziehungen restrukturiert.
- Die frühere direkte Bindung zwischen `Items` und `User` (Verleiher/Entleiher) wurde vollständig entfernt; Buchungen regeln nun alle Nutzerinteraktionen.
- Die externe Entität `InSy` wurde funktional reduziert und stellt nun lediglich Stammdaten für Items bereit.

8.1.2.5 5. Vereinheitlichung der Zeit- und Statusfelder

- Einführung systemweiter Felder für Historisierung: `created_at`, `updated_at`, `deleted_at`.
- Vereinheitlichung der Statusparameter: `requestStatus` aus dem Entwurf wurde durch das Attribut `status` in `Bookings` ersetzt.

8.1.2.6 6. Entfernte oder ersetzte Entitäten

- Cart wurde vollständig entfernt und vom Buchungskonzept ersetzt.
- ItemLog wurde nicht übernommen; Logik wird künftig über Buchungen oder Auditing gelöst.
- ItemDetails ging in Products und Categories auf.

Diese Anpassungen führten zu einer klaren Trennung zwischen Produktdefinition, physischen Einheiten, Benutzerinteraktionen und externen Systemen. Dadurch ist die finale Version konsistenter, normalisierter und deutlich besser wartbar als der ursprüngliche Entwurf.

8.1.3 Finale Umsetzung

Das in Abbildung 26 dargestellte Entity-Relationship-Modell beschreibt die strukturellen Zusammenhänge der in *LeihSy* verwalteten Entitäten und bildet die Grundlage für die relationale Implementierung der Datenbank. Das Modell verfolgt das Ziel, die Prozesse rund um die Verwaltung von Ausleihen, Produkten und zugehörigen Ressourcen konsistent und nachvollziehbar abzubilden.

8.1.3.1 Zentrale Entitäten Die Entität **Users** repräsentiert alle im System registrierten Benutzer. Neben Identifikationsmerkmalen umfasst sie Informationen wie Name, Budget sowie Metadaten zur Erstellung und Historisierung. Die Authentifizierung erfolgt nicht direkt im System, sondern über das externe Identitätsmanagement *Keycloak*, das für jede Benutzerinstanz anhand einer 1:1-Beziehung eindeutige Rollen- und E-Mail-Informationen bereitstellt.

Die Entität **Bookings** bildet den Kernprozess der Ausleihverwaltung ab. Eine Buchung wird von einer Benutzerinstanz initiiert und kann mehrere Items enthalten. Neben zeitlichen Attributen wie etwa Start-, End- und Rückgabedatum verwaltet die Entität Zustandsinformationen sowie potenzielle Vorschlags- oder Ausleihzeitpunkte. Die Beziehung zwischen **Users** und **Bookings** ist als 1:n ausgeprägt, da ein Benutzer mehrere Buchungen anlegen kann.

8.1.3.2 Ressourcen und Kategorien Mit der Entität **Products** werden alle ausleihbaren Produktarten strukturiert erfasst. Jedes Produkt ist einer Kategorie zugeordnet und kann mehrere physische Einheiten, sogenannte *Items*, besitzen. Die Entität enthält unter anderem Beschreibungen, Preisangaben, ein Bildverweisfeld sowie Verweise auf Kategorien und Standorte. Die Beziehung zwischen **Products** und **Items** ist folglich 1:n. Darüber hinaus kann ein Produkt Teil eines Sets sein.

Die physische Ebene wird durch die Entität `Items` repräsentiert. Jedes Item ist eine konkrete Instanz eines Produkts und verweist über einen Fremdschlüssel auf `InSy`. Zusätzlich werden Eigentumsinformationen, Inventarnummern sowie Metadaten zur Bestandsführung gespeichert.

Die Klassifizierung der Produkte erfolgt über die Entität `Categories`, die eine hierarchisch flache Struktur abbildet. Kategorien ermöglichen eine semantische Gruppierung der Produktpalette und unterstützen die spätere Filter- und Suchlogik im System.

8.1.3.3 Sets und Standorte Ein weiteres strukturelles Element stellen `Sets` dar. Ein Set ist eine definierte Kombination mehrerer Produkte, die gemeinsam ausgeliehen werden können. Die Beziehung ist n:m, sodass ein Set mehrere Produkte enthalten kann und ein Produkt mehreren Sets zugeordnet werden kann. Die Umsetzung erfolgt über eine assoziative Entität mit Verweisen auf Produkte sowie entsprechenden Metadaten.

Standortinformationen werden über die Entität `Locations` repräsentiert. Dies ermöglicht die räumliche Zuordnung einzelner Produkte und Items. Jeder Standort kann mehreren Produkten zugeordnet sein, während jedes Produkt genau einem Standort zugewiesen ist (1:n-Beziehung).

8.1.3.4 Externe Systeme Die Entität `InSy` ist ein externes Inventarisierungssystem, welches Produktinformationen zur Verfügung stellt.

8.1.3.5 Konsistenz und Historisierung Mehrere Entitäten enthalten optionale `deleted_at`-Attribute, die eine logische Löschung ermöglichen. Damit wird eine einfache Datenhaltung unterstützt, ohne physische Datensätze vollständig zu entfernen oder die Struktur der Tabellen zu verändern. Ergänzend werden `created_at`- und `updated_at`-Attribute zur transparenten Nachvollziehbarkeit von Zustandsänderungen eingesetzt.

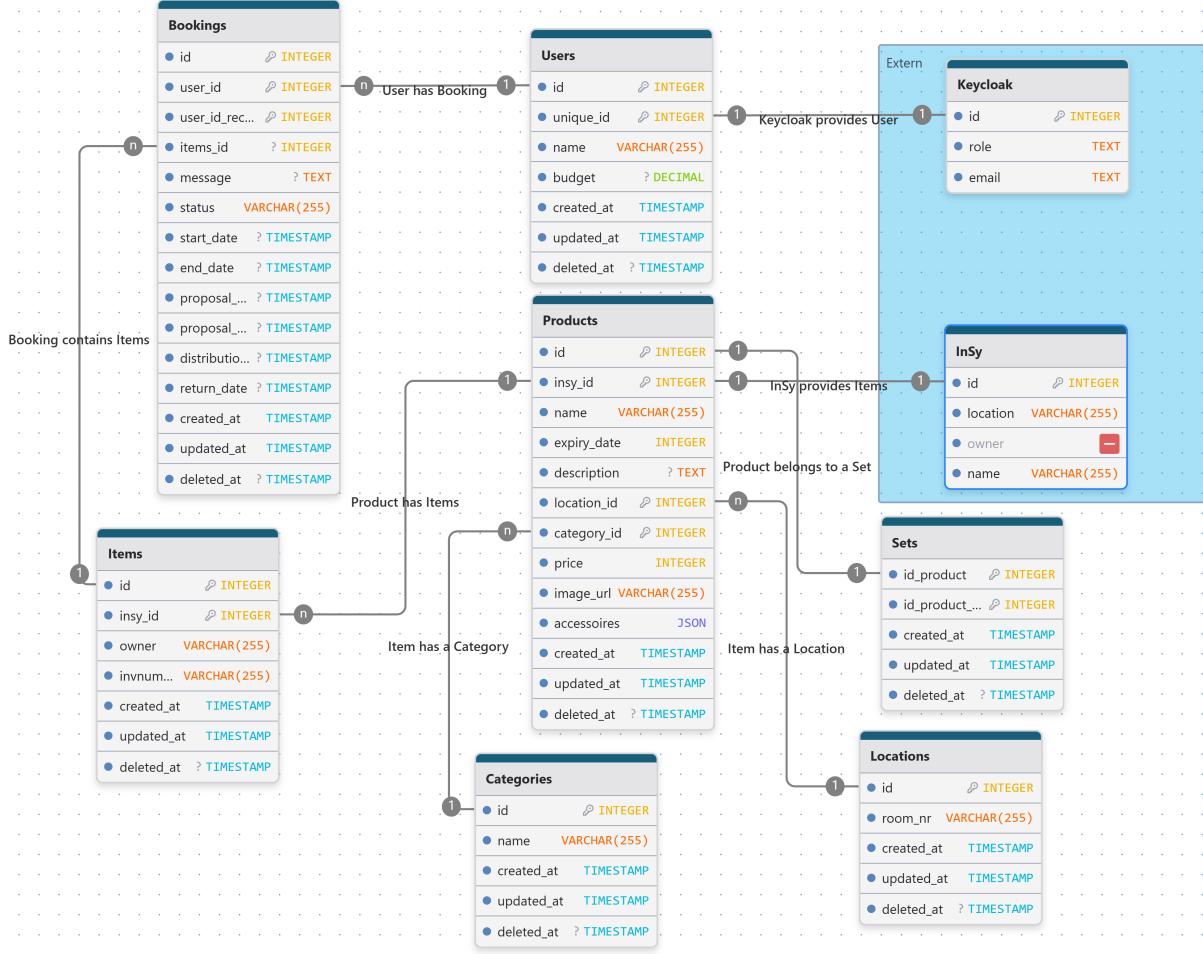


Abbildung 29: Entity-Relationship-Diagramm

8.2 Logische Sichten

8.2.1 Verteilungssicht

Die Anwendung folgt einem klassischen Client-Server-Architekturmmodell. Der Client ist der Webbrower, der auf dem Endgerät des Benutzers ausgeführt wird. Möchte der Benutzer die Anwendung aufrufen, sendet sein Browser zunächst HTTPS-Anfragen an einen Webserver, der als Docker-Container auf einem bwCloud-Server betrieben wird. Alle eingehenden Anfragen passieren zuvor einen Reverse Proxy, der die Requests anhand der Ziel-URL an die jeweils zuständigen Container weiterleitet.

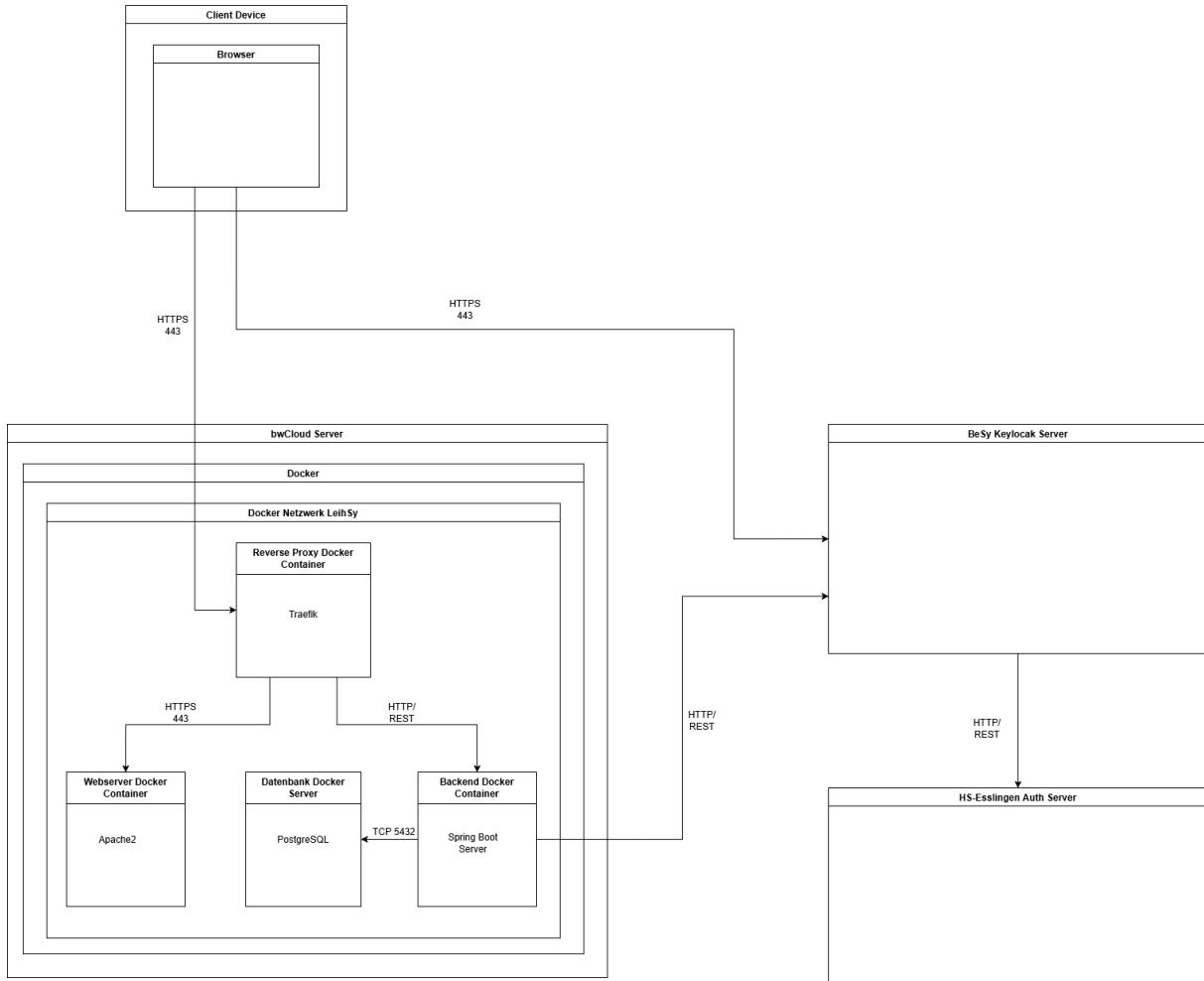


Abbildung 30: Diagramm zur Verteilungssicht

Der Webserver stellt dem Browser die benötigten statischen Ressourcen – HTML-, CSS- und JavaScript-Dateien – zur Verfügung, welche die Benutzeroberfläche der Anwendung aufbauen. Für den Zugriff auf die Daten des Ausleihsystems sendet der Browser anschließend HTTP/REST-Anfragen an die in der JavaScript-Anwendung definierten API-Endpunkte des Backends. Diese Anfragen werden ebenfalls vom Reverse Proxy empfangen und intern an den Backend-Container weitergeleitet.

Zur Persistierung von Daten kommuniziert das Backend über den TCP-Port 5432 mit der PostgreSQL-Datenbank. Da sich sowohl Datenbank-Container als auch Backend, Webserver und Reverse Proxy im gleichen Docker-Netzwerk befinden, erfolgt die Kommunikation direkt über dieses interne Netzwerk und ist von außen isoliert.

Für die Authentifizierung nutzt das System einen zentralen Keycloak-Server, der gemeinsam mit dem Projekt BeSy betrieben wird. Möchte ein Benutzer sich anmelden, erhält dessen Browser vom Backend eine Redirect-URL zum Keycloak-Auth-Endpoint. Der Browser ruft diesen Endpoint über HTTPS auf und der Benutzer authentifiziert sich dort mittels seines Hochschulaccounts.

Nach erfolgreicher Anmeldung leitet Keycloak den Browser mit einem Authorization Code zurück zum Backend, das diesen Code serverseitig gegen Access- und ID-Tokens eintauscht. Für die weitere Kommunikation speichert das Backend die Sitzung (z. B. mittels Session-Cookie) oder validiert eingehende Tokens lokal anhand des öffentlichen Schlüssels von Keycloak.

8.2.2 Struktursicht

Das Komponentendiagramm in Abb. 31 beschreibt die Architektur des LeihSy-Systems, das aus einem modular aufgebauten Frontend, einem funktionsorientierten Backend sowie mehreren externen Diensten besteht. Die Darstellung zeigt die Interaktionen der einzelnen Komponenten und verdeutlicht ihre funktionalen Abhängigkeiten.

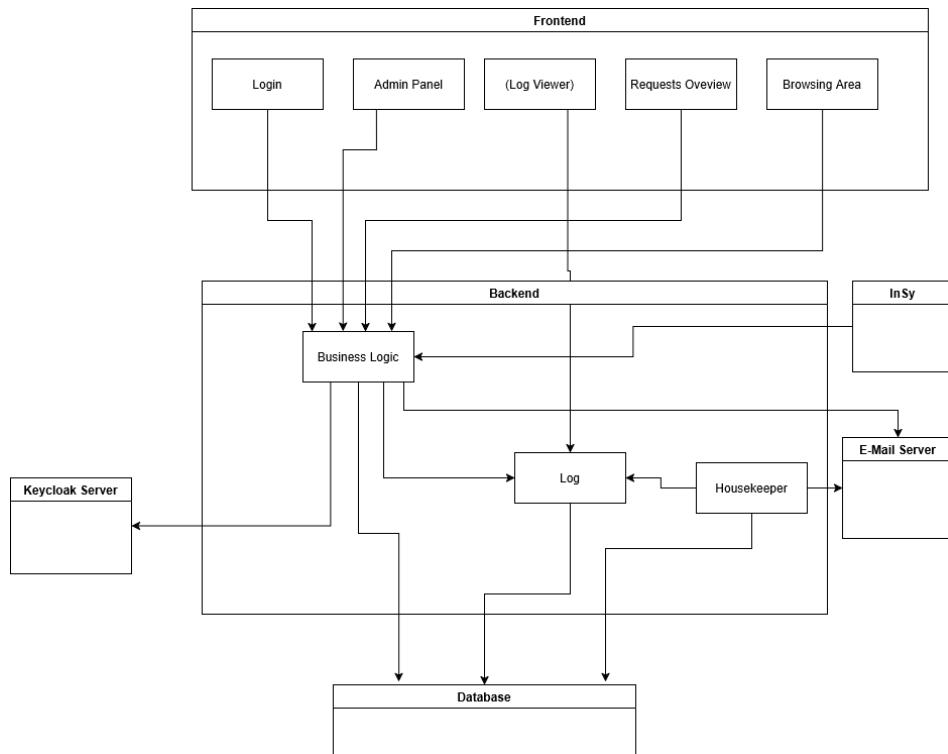


Abbildung 31: Komponentendiagramm

Das Frontend bildet die Präsentationsschicht des Systems und dient als Schnittstelle zwischen den Benutzerinnen und Benutzern und der Anwendungslogik. Es umfasst mehrere eindeutig abgegrenzte Funktionsbereiche:

- **Login:** Komponente zur Benutzerauthentifizierung.
- **Admin Panel:** Administrationsoberfläche für Konfigurations- und Verwaltungsaufgaben.

- **Log Viewer:** Optionaler oder eingeschränkter Zugriff auf System- und Anwendungsprotokolle.
- **Requests Overview:** Übersichtskomponente zur Darstellung und Verwaltung eingehender Anfragen.
- **Browsing Area:** Bereich zur Recherche oder Durchsicht von Datenobjekten.

Alle Frontend-Komponenten interagieren ausschließlich über definierte Schnittstellen mit dem Backend und enthalten keine eigene Geschäftslogik.

Das Backend stellt den zentralen Verarbeitungs- und Koordinationsmechanismus des Gesamtsystems dar. Es implementiert die Geschäftslogik und übernimmt sämtliche Datenzugriffe.

Die Komponente Business Logic bildet das funktionale Kernstück der Architektur. Ihre Hauptaufgaben umfassen:

- die Verarbeitung aller durch das Frontend ausgelösten Anfragen,
- die Integration externer Systeme,
- die Steuerung aller Datenzugriffe auf die Datenbank,
- die Weiterleitung relevanter Informationen an Logging- und Housekeeping-Dienste.

Die Log-Komponente aggregiert und persistiert systemweit auftretende Ereignisse, Fehler und Statusinformationen. Sie unterstützt damit:

- Systemmonitoring,
- Fehlerdiagnose,
- Nachvollziehbarkeit von Vorgängen.

Der Housekeeper führt periodische Hintergrundprozesse aus, darunter:

- die Bereinigung veralteter oder temporärer Einträge,
- die Ausführung zeitgesteuerter Systemaufgaben,
- die Interaktion mit dem E-Mail-Server zum Versand automatisierter Benachrichtigungen,
- die Dokumentation relevanter Abläufe über die Log-Komponente.

Der Keycloak-Server dient der Authentifizierung und Autorisierung. Anmeldevorgänge werden vom Backend an Keycloak delegiert. Die resultierenden Identitäts- und Berechtigungsinformationen werden anschließend von der Business Logic verarbeitet.

Das angebundene Inventarisierungssystem InSy übermittelt bereits inventarisierte Gegenstände über POST-Requests an das Backend. Die Business Logic validiert und verarbeitet diese Daten und integriert sie in das interne Datenmodell.

Der E-Mail-Server wird primär durch den Housekeeper angesteuert. Typische Anwendungsfälle sind:

- der Versand automatisierter Benachrichtigungen,
- die Übermittlung von Fehlermeldungen,
- die regelmäßige Statuskommunikation.

Die Datenbank dient als persistente Speicherkomponente des Systems. Sowohl die Business Logic als auch die Log-Komponente führen direkte Lese- und Schreiboperationen auf der Datenbank aus. Persistiert werden alle langfristig relevanten Informationen wie:

- Datenobjekte und Anfragen,
- Status- und Verlaufsdaten,
- systeminterne Strukturen, sofern diese nicht extern (z. B. in Keycloak) verwaltet werden.

Diese Struktur gewährleistet eine klare Trennung der Verantwortlichkeiten, hohe Wartbarkeit sowie eine robuste und nachvollziehbare Systemarchitektur.

- die Bereinigung veralteter oder temporärer Einträge,
- die Ausführung zeitgesteuerter Systemaufgaben,
- die Interaktion mit dem E-Mail-Server zum Versand automatisierter Benachrichtigungen,
- die Dokumentation relevanter Abläufe über die Log-Komponente.

8.2.3 Verhaltenssicht

8.2.3.1 Sequenzdiagramme In Abb 32 wird der Ausleihe Prozess modelliert, in welcher der Nutzer ein Gegenstand sucht, den Gegenstand seinem Warenkorb hinzufügt und diesen anschließend bestellt. Der Verleiher kann die Ausleih-Anfrage ablehnen oder zustimmen. Bei Letzterem schickt der Verleiher seine Termine an welchen er Zeit hat. Falls keine Reaktion des Verleiher erfolgt wird die Anfrage storniert.

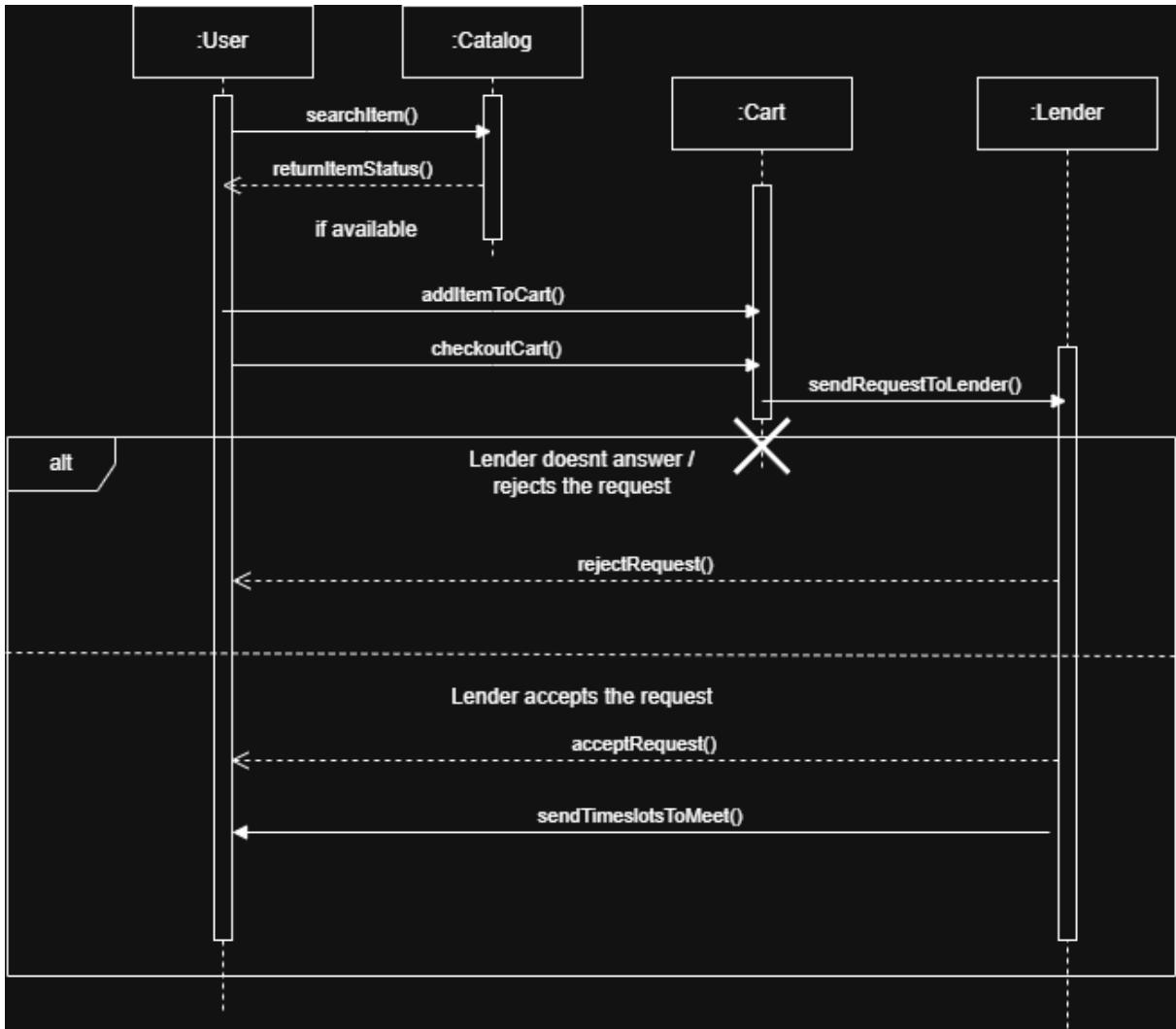


Abbildung 32: Sequenzdiagramm zum Ausleihprozess

Bei der Rückgabe bekommt der Nutzer eine Nachricht des Verwaltungssystems, dass der Gegenstand bald zurückgegeben werden müsse. Anschließend vereinbart der Ausleiher einen Termin mit dem Verleiher. Während des Termins schickt der Nutzer eine Bestätigungsanfrage an den Verleiher, das der Verleiher die Gegenstände erhalten hat und aktualisiert das Verwaltungssystem.

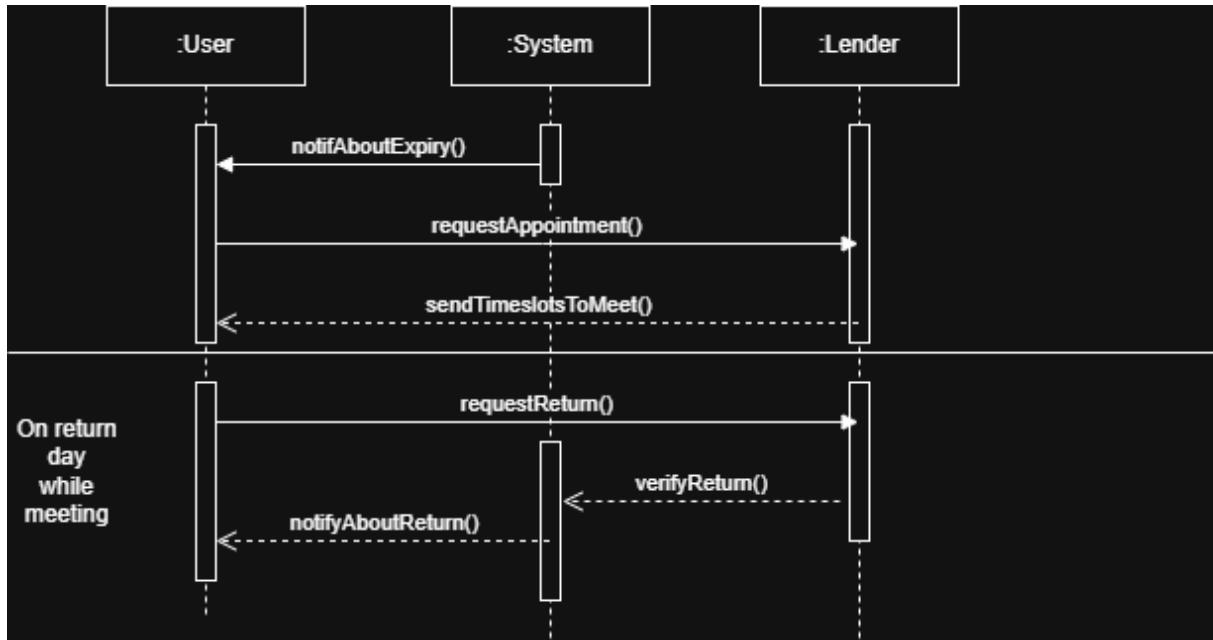


Abbildung 33: Sequenzdiagramm zum Rückgabeprozess

Der Login funktioniert über die KeyCloak API und muss dementsprechend von der Anwendung an KeyCloak übermittelt werden. Anschließend überprüft KeyCloak die Anmeldeinformationen. Wenn die Anmeldeinformationen falsch sind, wird der Zugriff verweigert. Wenn die Anmeldeinformationen korrekt sind, wird der Zugriff erlaubt und die WebApp lädt mit den Daten von KeyCloak die jeweiligen Profildaten in die Anwendung.

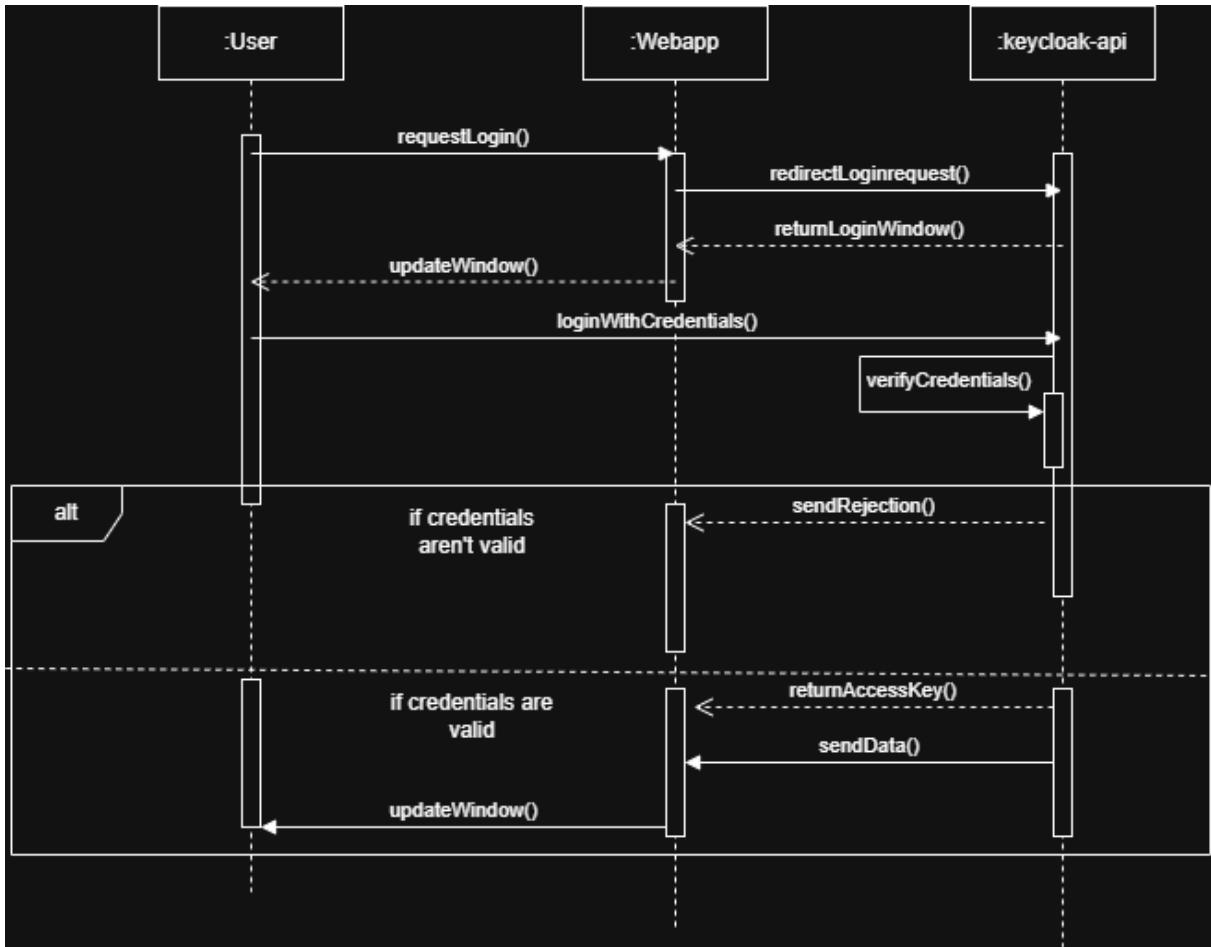


Abbildung 34: Sequenzdiagramm zum Login

8.3 API-Dokumentation

Die API-Dokumentation wird halbautomatisch durch Swagger erstellt. Sie wird von Swagger unter der URL `http://<Spring Boot-IP>:8080/swagger-ui/index.html` bereitgestellt. Dazu wurde Swagger für Spring Boot als Dependency in Maven hinzugefügt. Swagger analysiert die REST-Controller von Spring Boot automatisch und erstellt eine Übersicht über die API-Endpunkte, gibt anhand der Return-Values in den Controllern Beispieldaten aus und zeigt an, welche Parameter in einer REST-Anfrage enthalten sein können oder müssen. Die Beschreibungen zu den Parametern und der Bedeutung von Antwortcodes können händisch als Annotations in den REST-Controllern hinzugefügt werden. Diese werden dann ebenfalls automatisch von Swagger erkannt und in der API-Dokumentation dargestellt. Dies hat unter anderem den Vorteil dass die API-Dokumentation auch im Sourcecode enthalten ist und ein Nachschlagen in der Dokumentation gegebenenfalls übersprungen werden kann.

8.3.1 API-Umstrukturierung nach REST Best Practices

8.3.1.1 Übersicht

Dieses Kapitel dokumentiert die Umstrukturierung der LeihSy REST-API nach den Zalando API Guidelines und allgemeinen REST Best Practices. Die Änderungen wurden in drei Phasen durchgeführt und betreffen die Endpoints für Locations, Categories, Products, Bookings sowie die Ressourcen-Hierarchie.

8.3.1.2 Motivation

Im Weekly Meeting wurden mehrere API-Design-Probleme identifiziert:

- Verb-basierte Endpoints (z.B. `/upload`, `/search`) statt ressourcen-orientierter URLs
- Inkonsistente HTTP-Methoden (PUT statt PATCH für Teil-Updates)
- Falsche Ressourcen-Hierarchie (Sub-Ressource vor Parent-Ressource)
- Status-Filter als eigene Endpoints statt Query-Parameter
- Fehlende CRUD-Operationen für Locations und Categories

Die Umstrukturierung erfolgte schrittweise in drei Phasen, um Regressionsfehler zu minimieren und nach jeder Phase Tests durchführen zu können.

8.3.1.3 Phase 1: CRUD-Vervollständigung und Verb-Entfernung

8.3.1.4 LocationController

Hinzugefügte Endpoints:

- POST `/api/locations` – Erstellt neue Location mit Validierung der roomNr
- DELETE `/api/locations/{id}` – Löscht Location (Soft-Delete) mit Prüfung ob Items zugeordnet sind

Validierungslogik: Beim Löschen wird über `ItemRepository.countByProductLocationId()` geprüft, ob noch Items dieser Location zugeordnet sind. Falls ja, wird HTTP 400 mit Fehlermeldung zurückgegeben.

Repository-Erweiterung: Neue Query-Methode `countByProductLocationId()` im ItemRepository mit JPQL-Query über verschachtelte Beziehung (Item → Product → Location).

8.3.1.5 CategoryController

Hinzugefügte Endpoints:

- POST /api/categories – Erstellt neue Category mit Validierung des Namens
- DELETE /api/categories/{id} – Löscht Category (Soft-Delete) mit Prüfung ob Products zugeordnet sind

Validierungslogik: Analog zu Locations wird beim Löschen über `ProductRepository.countByCategory` geprüft, ob noch Products dieser Category zugeordnet sind.

Repository-Erweiterung: Neue Query-Methode `countByCategoryId()` im `ProductRepository`.

8.3.1.6 ImageController

Endpoint-Umbenennung:

- Alt: POST /api/images/upload
- Neu: POST /api/images

Begründung: Die HTTP-Methode POST impliziert bereits einen Upload. Der Subpath /upload ist redundant und nicht REST-konform.

8.3.1.7 Ergebnis Phase 1

Nach Phase 1 sind alle CRUD-Operationen für Locations und Categories vorhanden und verb-basierte Endpoints wurden eliminiert. Alle Änderungen wurden mit Postman getestet.

8.3.1.8 Phase 2: Ressourcen-Hierarchie und Product.isActive

8.3.1.9 Ressourcen-Hierarchie-Korrektur

In REST APIs sollte die Parent-Ressource immer vor der Child-Ressource in der URL stehen. Vorher war dies inkonsistent umgesetzt.

Geänderte Endpoints:

- /api/bookings/user/{userId} → /api/users/{userId}/bookings
- /api/bookings/lender/{lenderId} → /api/lenders/{lenderId}/bookings
- /api/items/products/{productId} → /api/products/{productId}/items

Controller-Struktur:

- UserController erhielt neuen Endpoint GET `/users/{userId}/bookings` mit Query-Parameter `deleted` für Soft-deleted Bookings
- Neuer LenderController mit Endpoint GET `/lenders/{lenderId}/bookings` und Status-Filter
- ProductController erweitert um GET `/products/{productId}/items`

BookingController Bereinigung: Folgende Endpoints wurden aus dem BookingController entfernt, da sie nun in UserController bzw. LenderController liegen:

- GET `/bookings/lenders/me/pending`
- GET `/bookings/lenders/me`
- GET `/bookings/users/me`
- GET `/bookings/users/{userId}`
- GET `/bookings/lenders/{lenderId}`
- GET `/bookings/lenders/{lenderId}/pending`

Die `/me` Endpoints wurden komplett entfernt, da das Frontend die `userId` bzw. `lenderId` aus dem JWT-Token extrahieren kann.

8.3.1.10 Product.isActive Feld

Entity-Erweiterung: Product-Entity erhielt neues Feld `isActive` (Boolean, default true).

Verwendungszweck: Inaktive Produkte können in der Suche ausgegraut angezeigt werden und sind nicht ausleihbar, bleiben aber in der Datenbank erhalten.

Datenbank-Migration: PostgreSQL-Tabelle `products` wurde um Spalte `is_active` BOOLEAN DEFAULT `true` erweitert. Bei H2 (Development) erfolgt die Schema-Erstellung automatisch.

ItemMapper-Erweiterung: Mapping für `isAvailable` wurde hinzugefügt, welches die Methode `item.isAvailable()` aufruft. Ursprünglich hatte das DTO das Feld `available`, was zu Problemen mit MapStruct führte. Das Feld wurde in `isAvailable` umbenannt, um der Java Bean Naming Convention zu entsprechen.

8.3.1.11 Service-Methoden-Anpassungen

BookingService: Die Methoden `getBookingsByUserId()`, `getBookingsByLenderId()` und `getPendingBookingsByLenderId()` erhielten einen zusätzlichen Parameter `includeDeleted` (boolean). Bei `includeDeleted=true` werden auch Soft-deleted Bookings über `findAll()` geladen und im Service gefiltert.

ItemService: Methode `getItemsByProduct()` wurde in `getItemsByProductId()` umbenannt für Konsistenz mit anderen Service-Methoden.

8.3.1.12 ProductController Suchfunktionen

Der Endpoint GET `/products/search` wurde entfernt. Stattdessen unterstützt der Haupt-Endpoint GET `/products` nun Query-Parameter:

- `search` – Volltext-Suche in Name und Beschreibung
- `categoryId` – Filterung nach Category
- `locationId` – Filterung nach Location

Die Priorisierung erfolgt: `search > categoryId > locationId > alle Produkte.`

8.3.1.13 Ergebnis Phase 2

Die API folgt nun konsistent dem Prinzip der Ressourcen-Hierarchie. Parent-Ressourcen stehen in der URL vor Child-Ressourcen. Alle Service-Methoden unterstützen das Filtern von Soft-deleted Entities.

8.3.1.14 Phase 3: Query-Parameter statt Sub-Routes

8.3.1.15 Problem mit Status-Filter-Endpoints

Vorher existierten dedizierte Endpoints für verschiedene Booking-Status:

- GET `/api/bookings/overdue`
- GET `/api/bookings/pending`

Dies ist nicht REST-konform und verschwendet URL-Namespace. Filter sollten als Query-Parameter realisiert werden.

8.3.1.16 Durchgeführte Änderungen

BookingController: Der `/overdue` Endpoint wurde entfernt. Stattdessen gibt es einen zentralen Endpoint:

- GET `/api/bookings?status=overdue`
- GET `/api/bookings?status=pending`
- GET `/api/bookings?status=confirmed`
- GET `/api/bookings?status=picked_up`

- GET /api/bookings?status=returned
- GET /api/bookings?status=cancelled
- GET /api/bookings?status=expired
- GET /api/bookings?status=rejected
- GET /api/bookings – Alle Bookings (ohne Filter)

Endpoint-Reihenfolge: Wichtig war die korrekte Anordnung der Mapping-Annotations im Controller. Der `@GetMapping` ohne Path-Variable muss VOR dem `@GetMapping(/ {id})` stehen, da Spring die Endpoints von oben nach unten matched.

8.3.1.17 Repository-Queries für Status-Filter

Die Service-Methode `getAllBookings(String status)` verwendet keine Stream-Filterung mehr, sondern dedizierte Repository-Queries. Dies verbessert die Performance erheblich, da die Filterung in der Datenbank statt im Application-Server erfolgt.

Neue Repository-Queries:

- `findAllPending(threshold)` – Bookings mit `createdAt > now - 24h`, keine proposedPickups
- `findAllConfirmed(threshold)` – Bookings mit `confirmedPickup < now + 24h`, noch nicht abgeholt
- `findAllCancelled(threshold)` – Bookings mit `createdAt < now - 24h`, nicht bestätigt
- `findAllExpired(threshold)` – Bookings mit `confirmedPickup < now - 24h`, nicht abgeholt
- `findAllPickedUp()` – Bookings mit `distributionDate` gesetzt, `returnDate` null
- `findAllReturned()` – Bookings mit `returnDate` gesetzt
- `findAllRejected()` – Bookings mit `deletedAt` gesetzt
- `findAllActive()` – Alle Bookings mit `deletedAt` null

Die Queries bilden die gleiche Logik ab wie die `calculateStatus()` Methode in der Booking-Entity, verwenden jedoch JPQL statt Java-Code. Der 24-Stunden-Threshold wird als Parameter übergeben.

8.3.1.18 Ergebnis Phase 3

Die API folgt nun konsistent dem REST-Prinzip, dass Filter als Query-Parameter statt als Sub-Routes implementiert werden. Die Performance wurde durch datenbankbasierte Filterung deutlich verbessert. Die Endpoint-Struktur ist zukunftssicher und ermöglicht einfaches Hinzufügen weiterer Filter-Parameter.

8.3.1.19 Phase 4: PATCH statt PUT und DELETE für Reject

8.3.1.20 Motivation

HTTP-Methoden haben in REST APIs eine spezifische semantische Bedeutung. PUT ist für vollständige Ressourcen-Ersetzungen gedacht (Replace), während PATCH für Teil-Updates konzipiert ist. Die bestehenden Booking-Status-Endpoints verwendeten PUT für Teil-Updates einzelner Felder, was nicht REST-konform ist.

Zusätzlich wurde die Reject-Funktionalität über einen separaten PUT-Endpoint realisiert, obwohl ein Reject technisch eine Löschung darstellt (Soft-Delete). REST Best Practices empfehlen DELETE für Löschoperationen.

8.3.1.21 Ausgangssituation

Der BookingController enthielt sechs separate PUT-Endpoints für verschiedene Status-Übergänge:

- PUT /api/bookings/{id}/confirm – Verleiher bestätigt mit Terminvorschlägen
- PUT /api/bookings/{id}/select-pickup – Student wählt Abholtermin
- PUT /api/bookings/{id}/propose – Gegenvorschlag (Ping-Pong)
- PUT /api/bookings/{id}/pickup – Ausgabe dokumentieren
- PUT /api/bookings/{id}/return – Rückgabe dokumentieren
- PUT /api/bookings/{id}/reject – Booking ablehnen

Zusätzlich existierte ein DELETE-Endpoint für Stornierungen durch Student/Admin. Dies führte zu Redundanz, da Reject und Cancel technisch beide ein Soft-Delete durchführen.

8.3.1.22 Probleme der alten Struktur

Semantische HTTP-Methoden-Verletzung: PUT impliziert das Ersetzen einer gesamten Ressource. Die Endpoints aktualisierten jedoch nur einzelne Felder (z.B. `distributionDate` bei `pickup`), was eine klassische PATCH-Operation ist.

URL-Namespace-Verschwendung: Sechs separate Endpoints für Operationen auf derselben Ressource führten zu unnötiger API-Komplexität und erschwerten die Wartung.

Inkonsistente Löschsemantik: Reject nutzte PUT statt DELETE, obwohl technisch ein Soft-Delete (Setzen von `deletedAt`) durchgeführt wurde.

Frontend-Komplexität: Das Frontend musste für jede Status-Änderung unterschiedliche Endpoints kennen und Request-Bodies konstruieren.

8.3.1.23 Neue Struktur mit PATCH

Generischer PATCH-Endpoint: Alle Status-Updates erfolgen nun über einen einzigen Endpoint PATCH `/api/bookings/{id}` mit einem action-Parameter im Request-Body.

BookingStatusUpdateDTO: Neue DTO-Klasse mit drei Feldern:

- `action` (String, required) – Art der Status-Änderung
- `proposedPickups` (List<LocalDateTime>, optional) – Für confirm und propose
- `selectedPickup` (LocalDateTime, optional) – Für select_pickup

Unterstützte Actions:

- `confirm` – Verleiher bestätigt Booking, benötigt proposedPickups
- `select_pickup` – Student wählt Termin aus, benötigt selectedPickup
- `propose` – Gegenvorschlag machen, benötigt proposedPickups
- `pickup` – Ausgabe dokumentieren, keine zusätzlichen Parameter
- `return` – Rückgabe dokumentieren, keine zusätzlichen Parameter

8.3.1.24 DELETE statt PUT für Reject

Der `/reject` Endpoint wurde komplett entfernt. Stattdessen übernimmt der DELETE-Endpoint beide Funktionen:

- Verleiher lehnt Booking ab → DELETE setzt `deletedAt`
- Student storniert Booking → DELETE setzt `deletedAt`

Technisch führen beide Operationen die gleiche Aktion aus (Soft-Delete), daher ist eine Unterscheidung auf API-Ebene nicht notwendig. Die Business-Logik kann über Rollen und Berechtigungen gesteuert werden.

8.3.1.25 Service-Layer-Anpassungen

BookingService.updateStatus(): Neue zentrale Methode, die per Switch-Case die entsprechende bestehende Service-Methode aufruft. Die Action-Validierung erfolgt ebenfalls hier.

Die Methode extrahiert zunächst den aktuellen User aus dem Security Context (für propose-Action benötigt). Anschließend wird basierend auf der Action die entsprechende bestehende Service-Methode aufgerufen:

- `confirm` → `confirmBooking(id, proposedPickups)`
- `select_pickup` → `selectPickupTime(id, selectedPickup)`
- `propose` → `proposeNewPickups(id, userId, proposedPickups)`
- `pickup` → `recordPickup(id)`
- `return` → `recordReturn(id)`

Ungültige Actions oder fehlende Required-Parameter führen zu `IllegalArgumentException` mit aussagekräftiger Fehlermeldung.

Vorteile dieser Struktur:

- Bestehende Service-Methoden bleiben unverändert (keine Regression)
- Zentrale Validierung der Action und Parameter
- Einfaches Hinzufügen neuer Actions in Zukunft
- Klare Fehlerbehandlung mit spezifischen Meldungen

8.3.1.26 Controller-Vereinfachung

Der BookingController wurde von 290 Zeilen auf 180 Zeilen reduziert. Die sechs PUT-Endpoints und ihre zugehörigen Request-DTO-Klassen wurden entfernt. Übrig blieben:

- GET `/api/bookings` – Alle Bookings mit optionalen Filtern
- GET `/api/bookings/{id}` – Einzelne Booking
- POST `/api/bookings` – Neue Booking erstellen
- PATCH `/api/bookings/{id}` – Status-Update
- DELETE `/api/bookings/{id}` – Ablehnen/Stornieren

Diese fünf Endpoints bilden nun die komplette CRUD-Funktionalität sowie alle Status-Übergänge ab.

8.3.1.27 Swagger-Dokumentation

Der PATCH-Endpoint erhielt umfassende OpenAPI-Annotations:

- @Operation mit Beschreibung aller unterstützten Actions
- @ApiResponses für 200 (Success), 400 (Invalid Action/Parameters), 401 (Unauthorized), 404 (Not Found)
- @Parameter-Annotation für die Booking-ID
- @RequestBody mit Schema-Referenz auf BookingStatusUpdateDTO

Das BookingStatusUpdateDTO selbst enthält detaillierte @Schema-Annotations mit Beispielwerten für jedes Feld, sodass die Swagger-UI vollständige Request-Beispiele generieren kann.

8.3.1.28 Fehlerbehandlung

Die Service-Methode `updateStatus()` wirft spezifische Exceptions:

- `IllegalArgumentException` – Bei ungültiger Action oder fehlenden Required-Parametern
- `RuntimeException` – Bei nicht gefundener Booking (durch `getBookingById()`)
- `IllegalStateException` – Bei Business-Logic-Violations (z.B. Pickup ohne confirmedPickup)

Der GlobalExceptionHandler fängt diese Exceptions und generiert aussagekräftige HTTP-Responses mit passenden Status Codes (400 für Validierungsfehler, 404 für Not Found, 409 für Konfliktfehler).

8.3.1.29 Vorteile der neuen Struktur

REST-Konformität: PATCH wird semantisch korrekt für Teil-Updates verwendet, DELETE für Löschoperationen.

Vereinfachte API: Statt sechs Endpoints nur noch einer für alle Status-Updates, was die API-Oberfläche deutlich reduziert.

Konsistente Request-Struktur: Alle Status-Updates folgen dem gleichen Pattern mit action-Parameter, was die Frontend-Integration vereinfacht.

Erweiterbarkeit: Neue Actions können einfach im Switch-Case hinzugefügt werden ohne zusätzliche Endpoints zu erstellen.

Bessere Testbarkeit: Ein Endpoint statt sechs reduziert die Anzahl der API-Tests erheblich.

Type-Safety: Die Action-Parameter sind in der BookingStatusUpdateDTO typsicher definiert und dokumentiert.

Das Frontend muss entsprechend angepasst werden, um die neuen Endpoint-Strukturen zu verwenden.

9 Implementierung

9.1 Keycloak

Für die Benutzer- und Rollenverwaltung wird die Open-Source-Software Keycloak eingesetzt. Das im Projekt verwendete Realm wird bereits von BeSy und InSy benutzt. Im Realm „Hochschule Esslingen“ wurden für das Projekt drei Clients eingerichtet, die jeweils verschiedene technische Einsatzumgebungen abbilden:

- `leihsy-frontend-dev` - Client für die lokale Entwicklungsumgebung,
- `leihsy-frontend-prod` - Client für den produktiven Betrieb der Webanwendung,
- `leihsy-frontend-test` - Client für Tests und Integrationsumgebungen.

Alle Clients verwenden OpenID Connect (OIDC) als Authentifizierungsstandard, um die Identität der Benutzer sicherzustellen. Für die Verwaltung der Zugriffsrechte wird darüber hinaus OAuth 2.0 genutzt. Über die im Realm definierten Rollen kann Keycloak Berechtigungen an die Anwendung übergeben, welche diese dann zur Zugriffskontrolle weiterverarbeitet.

Die für das Projekt eingerichteten Rollen lassen sich in drei Kategorien gliedern:

- **User**
- **Lender**
- **Admin**

Die Rolle User stellt die Standardrolle dar, die allen Personen zugewiesen wird, die sich im System anmelden können. Die Rolle Lender repräsentiert Verleiher. Sie umfasst alle Berechtigungen der Standardrolle sowie zusätzlich den Zugriff auf das erweiterte Verleiher-Dashboard. Die Rolle Admin ermöglicht die Verwaltung von Benutzern in Keycloak und von ausleihbaren Gegenständen innerhalb der Anwendung.

9.2 Frontend

9.2.1 User Stories

- Medienkatalog filtern & Zeitraum prüfen
 - Als Studierende*r möchte ich auf der Katalogseite alle Geräte sehen und den Katalog nicht nur nach Kategorien (z. B. VR-Geräte, Kameras), sondern auch nach einem konkreten Datumszeitraum, nach Campus sowie nach „nur verfügbare Geräte“ filtern können, damit ich sofort erkenne, welche Geräte für mein geplantes Projektfenster tatsächlich verfügbar sind.
- Technische Spezifikationen einsehen
 - Als Studierende*r möchte ich detaillierte technische Daten (z. B. Sensorgröße, Speicher) und das enthaltene Zubehör einsehen können, um die Eignung eines Geräts für mein Vorhaben zu bewerten.
- Mobile Nutzung
 - Als mobile Nutzer*in möchte ich über ein für Smartphones optimiertes Menü navigieren, um auch unterwegs Verfügbarkeiten bequem prüfen zu können.
- Gerätedetails und Ausleihbarkeit prüfen
 - Als Studierende*r möchte ich auf der Detailseite eines Geräts ein Abholdatum wählen können. Das System soll mir dabei visuell (z. B. durch eine rote Markierung) anzeigen, wenn das Gerät an diesem Datum nicht verfügbar ist, damit ich Zeit spare und direkt einen verfügbaren Termin wählen kann.

9.2.2 Struktursicht

Die Anwendung ist als Single Page Application (SPA) konzipiert. Die Struktur gliedert sich logisch in drei Ebenen:

1. **Seiten-Komponenten:** Diese repräsentieren ganze Ansichten (z. B. „Katalogseite“, „Detailseite“). Sie sind für die Kommunikation mit der Datenlogik verantwortlich.
2. **UI-Komponenten:** Wiederverwendbare Bausteine wie Kopfzeile, Fußzeile oder Informationskarten für Campus-Standorte. Sie dienen rein der Darstellung und erhalten ihre Daten von den übergeordneten Seiten.
3. **Dienste:** Im Hintergrund agierende Einheiten, die die Datenhaltung und Geschäftslogik kapseln. Sie bilden die Schnittstelle zwischen Komponenten und den später folgenden Backend-Daten.

10 Features

10.1 Admin Dashboard

10.1.1 Admin-Menü

Das Menü für den Admin befindet sich oben rechts im Header unter dem Punkt "**Verwaltung**", dieses Menü kann man ausschließlich aufrufen, wenn einem die Administrator Rolle zugewiesen wurde. Im Menü befinden sich alle Funktionen die ausschließlich dem Administrator vorbehalten sind und werden in den folgenden Abschnitten näher beschrieben. Die Menüpunkte sind Listenartig angeordnet und jeder Menüpunkt enthält:

- **Icon**
- **Titel,**
- **Beschreibung des Menüs.**

10.1.1.1 Änderungshistorie

Name	Menü Komponente
Beschreibung	Erstellung einer Menü-Komponente zur einheitlicher Darstellung, der Menüpunkte.
Begründung	Verbesserung der Übersichtlichkeit und Vereinheitlichung der Darstellung aller Administrationsfunktionen.
Auswirkungen	Schnellere Produktpflege und konsistentes UI-Design.

Name	Änderung des Menü-Designs
Beschreibung	Änderung des Designs von Quadratischen Kacheln zu flachen schmalen Kacheln.
Begründung	Es passen mehr kleinere Kacheln auf den Screen.
Auswirkungen	Schnellere Orientierung.

Name	Änderung des Menü-Layouts
Beschreibung	Änderung des Layouts von zwei quadratischen Menükacheln in einer Zeile zu einer schmalen Menükachel pro Zeile.
Begründung	Klarere Anordnung mit mehr Kacheln auf dem Bildschirm.
Auswirkungen	Bessere Anordnung und Orientierung.

10.1.2 Admin-Menü: Alle Gegenstände (Übersicht)

Das Menü **Alle Gegenstände (Übersicht)** zeigt eine vollständige Übersicht über sämtliche Gegenstände im System. Sie dient Administratoren dazu, alle Produktinstanzen unabhängig vom übergeordneten Produkt zentral zu verwalten.

Im Header werden drei Kennzahlen hervorgehoben dargestellt:

- **Gesamtanzahl der Gegenstände,**
- **Anzahl der aktuell verfügbaren Gegenstände,**
- **Anzahl der ausgeliehenen Gegenstände.**

Darunter befindet sich ein globales Suchfeld, mit dem nach Produktnamen, Kategorien oder Inventarnummern gesucht werden kann.

Die Gegenstände sind nach ihren zugehörigen Produkten gruppiert. Jede Produktgruppe zeigt:

- den Produktnamen,
- die Kategorie,
- den Standort,
- die Anzahl verfügbarer Gegenstände.

Innerhalb jeder Produktgruppe werden die einzelnen Gegenstände in Tabellenform dargestellt. Die Tabelle enthält folgende Spalten:

- **Inventarnummer,**
- **Besitzer,**
- **Status** (mit visueller Hervorhebung),

Die admin-spezifische Darstellung ermöglicht eine schnelle Kontrolle der Systembestände, erleichtert das Auffinden bestimmter Gegenstände und unterstützt die effiziente Verwaltung großer Inventare.

Des Weiteren wird man bei einem Klick auf den Gegenstand auf eine Detail-Seite weitergeleitet. Auf dieser sieht die Details des Gegenstandes und dem übergeordneten Produkt, sowie Verlinkungen zu den Bearbeitungsmenüs. Außerdem werden alle Ausleihen des Gegenstandes tabellarisch aufgelistet mit folgenden Spalten:

- **Ausleihen,**
- **Von & Bis,**
- **Status** (mit visueller Hervorhebung),

10.1.2.1 Änderungshistorie

Name	Verwendung der Tabellen-Komponente
Beschreibung	Verwendung der selbst erstellten Tabellen-Komponente zur einheitlichen Darstellung von Daten in Tabellen (Anzeige von Ausleihen und Gegenständen).
Begründung	Erhöhung der Wiederverwendbarkeit und Wartbarkeit des UI-Designs.
Auswirkungen	Schnellere Komponentenpflege sowie konsistentes UI-Design.

Name	Bearbeitungsverlinkungen
Beschreibung	Hinzufügen von direkten Links zu den Bearbeitungsmenüs der Produkte und Gegenstände.
Begründung	Vereinfachung der Navigation zu den Menüs, bei Änderungsbedarf.
Auswirkungen	Schnellere und einfachere Navigation, sowie bessere Nutzerfahrung.

10.1.3 Admin-Menü: Produktgruppen verwalten

Das Menü "Produktgruppen verwalten" zeigt eine tabellarische Übersicht aller im System vorhandenen Produkte. Sie dient der zentralen Verwaltung des Produktbestands und ermöglicht den schnellen Zugriff auf Bearbeitungs- und Löschfunktionen.

Im oberen Bereich befindet sich ein Suchfeld, über das Produkte nach Name, Beschreibung oder Kategorie gefiltert werden können.

Die Tabelle enthält folgende Spalten:

- **ID:** Eindeutige Produkt-ID.
- **Name:** Bezeichnung des Produkts.
- **Kategorie:** Zugeordnete Produktkategorie.
- **Standort:** Aktueller Lagerort.
- **Preis/Tag:** Tagesmietpreis in Euro.
- **Aktionen:** Schaltflächen zum Bearbeiten oder Löschen des Produkts.

Die Listenansicht ist scrollbar ausgeführt und ermöglicht dadurch auch bei großen Produktbeständen eine übersichtliche Darstellung.

Um neue Produkte anzulegen muss man auf den "Neues Produkt" Button klicken. Dadurch gelangt man zu einem Formular in welchem man Produkte anlegen kann. Das Formular enthält folgende Felder im Tab **Allgemeine Informationen**:

- **Name:** Bezeichnung des Produkts (z. B. Kameramodell).
- **Kategorie:** Zuordnung zu einer übergeordneten bereits erstellten Kategorie.
- **Beschreibung:** Ausführliche Beschreibung und technische Details.
- **Produktbild:** Upload eines Bildes im Format JPG, PNG oder WebP.
- **Zubehör:** Liste optionaler Komponenten wie Kabel, Netzteile oder Controller, die im Produkt enthalten sind.

Das Formular enthält folgende Felder im Tab **Detailinformationen**:

- **Max. Ausleihdauer (Tage):** Maximale Anzahl an Tagen, für die ein Produkt ausgeliehen werden darf.
- **Preis / Tagessatz (€):** Täglicher Mietpreis des Produkts.
- **Raumnummer:** Auswahl des physischen Lagerortes des Produkts.
- **Location ID:** Automatisch ermittelte Standortkennung, die aus der angegebenen Raumnummer generiert wird.

Pflichtfelder sind entsprechend im Formular gekennzeichnet. Über die Schaltfläche „Erstellen“ wird das Produkt erstellt.

Wenn man in der Tabelle auf Bearbeiten klickt wird man auf das Bearbeitungsformular weitergeleitet. Dieses ist gleich aufgebaut wie das Erstellungsformular, mit dem Unterschied, dass die Produktinformationen bereits eingetragen sind und bearbeitet werden können. Über die Schaltflächen „Aktualisieren“ und „Abbrechen“ können Änderungen gespeichert oder verworfen werden.

Die Funktionen ermöglichen das zentrale Erstellen und Bearbeiten von Produktgruppen, denen anschließend einzelne Gegenstände zugeordnet werden können.

10.1.3.1 Änderungshistorie

Name	Verwendung der Tabellen-Komponente
Beschreibung	Verwendung der selbst erstellten Tabellen-Komponente zur einheitlichen Darstellung von Daten in Tabellen (Anzeige der Produktgruppen).
Begründung	Erhöhung der Wiederverwendbarkeit und Wartbarkeit des UI-Designs.
Auswirkungen	Schnellere Komponentenpflege sowie konsistentes UI-Design.
Name	Verwendung Button-Komponenten
Beschreibung	Verwendung der selbst-erstellten Button-Komponenten zur einheitlichen Darstellung von Aktionen.
Begründung	Wiederverwendbarkeit, Wartbarkeit sowie einheitliche Farben für Aktionen.
Auswirkungen	Schnellere Komponentenpflege, sowie konsistentes und intuitives UI-Design.
Name	Dropdownmenüs
Beschreibung	Hinzufügen von Dropdownmenüs für Kategorie- und Standortauswahl.
Begründung	Schnelleres Aufinden von Kategorien und Standorten, ohne ID zu kennen.
Auswirkungen	Bessere Usability und intuitives UI-Design.

10.1.4 Admin-Menü: Einzelne Gegenstände verwalten

Das Menü „Einzelne Gegenstände verwalten“ stellt alle im System erfassten Produkte in einer übersichtlichen, kategorisierten Listenstruktur dar. Ziel der Seite ist es, dem Nutzer einen schnellen Überblick über alle verfügbaren Produkte sowie deren zugehörige Einzelgegenstände (Instanzen) zu ermöglichen.

Im oberen Bereich befindet sich ein Suchfeld, über das nach Produktnamen, Kategorie oder Standort gefiltert werden kann.

Jedes Produkt wird in einem eigenen Abschnitt dargestellt. Der Kopfbereich eines Produktes enthält folgende Informationen:

- den Produktnamen als Titel,
- die zugehörige Kategorie,
- den Standort des Produktes,
- die Anzahl verfügbarer bzw. insgesamt vorhandener Gegenstände.

Über ein Aufklapp-Element kann der Nutzer die Liste der Einzelgegenstände des Produktes einsehen. Für jeden Gegenstand werden folgende Attribute angezeigt:

- **ID** des Gegenstandes,
- **Inventarnummer**,
- **Besitzer** bzw. Verantwortlicher,
- **Verleiher** inklusive E-Mail-Adresse und System-ID,
- **Status** (z. B. „Verfügbar“),
- **Aktionen** zum Bearbeiten oder Löschen.

Im rechten oberen Bereich eines Produktabschnitts befindet sich ein Plus Button zum Hinzufügen neuer Gegenstände. Die Darstellung ermöglicht eine schnelle Einschätzung der Verfügbarkeit jedes Produktes sowie eine effiziente Verwaltung einzelner Einheiten.

Durch das klicken des Plus Buttons kommt man zum Administrationsformular welches, zur Bearbeitung einzelner physischer Gegenstände genutzt werden kann. Dieses Menü ermöglicht das Anlegen, Ändern oder Löschen eines bestimmten Inventarobjekts.

Das Formular umfasst:

- **Inventarnummer-Präfix:** Grundkennzeichnung, an die automatisch eine Nummer angehängt wird.
- **Besitzer User ID:** Zuordnung zu einem Besitzer.
- **Verleiher User ID / Benutzername:** Angabe des administrativen Verantwortlichen.
- **Anzahl zu erstellender Instanzen:** Anzahl der Instanzen, die erstellt werden sollen
- **Produkt:** Zugehörige Produktgruppe.
- **Verfügbarkeitsstatus:** Markierung, ob das Gerät aktuell ausgeliehen werden kann.

Pflichtfelder sind entsprechend im Formular gekennzeichnet. Über die Schaltfläche „Erstellen“ wird der Gegenstand erstellt. Durch den Button

Abbrechen wird der Vorgang abgebrochen.

10.1.4.1 Änderungshistorie

Name	Verwendung der Tabellen-Komponente
Beschreibung	Verwendung der selbst erstellten Tabellen-Komponente zur einheitlichen Darstellung von Daten in Tabellen (Anzeige der Produktgruppen).
Begründung	Erhöhung der Wiederverwendbarkeit und Wartbarkeit des UI-Designs.
Auswirkungen	Schnellere Komponentenpflege sowie konsistentes UI-Design.
Name	Verwendung von Button-Komponenten
Beschreibung	Verwendung selbst erstellter Button-Komponenten zur einheitlichen Darstellung von Aktionen.
Begründung	Verbesserung der Wiederverwendbarkeit, Wartbarkeit sowie Sicherstellung einheitlicher Farbgebung für Aktionen.
Auswirkungen	Schnellere Komponentenpflege sowie konsistentes und intuitives UI-Design.
Name	Vorschau bei Eingabe
Beschreibung	Hinzufügen einer automatischen Vorschau für Besitzer und Verleiher bei der Eingabe von Namen oder ID.
Begründung	Schnelleres Auffinden von Personen sowie Reduktion von Fehlern bei der Verleiher- und Besitzerzuweisung.
Auswirkungen	Geringere Fehleranfälligkeit bei Zuweisungen und verbesserte Usability.

10.1.5 Admin-Menü: Buchungen verwalten

Das Menü "Buchungen verwalten" bietet dem Admin die Möglichkeit alle Buchungen zu sehen. Die Buchungen werden tabellarisch aufgelistet beginnend mit den neusten Buchungen. Die Tabelle hat folgende Spalten:

- **Ausleihen:** Name des Ausleihers.
- **Produkt:** Bezeichnung des Produkts.
- **Inventarnummer:** Inventarnummer des Gegenstandes.
- **Verleiher:** Name des Verleiher.
- **Status:** Aktueller Status der Buchung.

- **Abholung:** Beginn der Ausleihe als Datum.
- **Rückgabe:** Ende der Ausleihe als Datum.
- **Erstellt am:** Erstellungsdatum.

Über der Tabelle befindet sich eine Suchleiste, mit welcher der Admin die Buchungen filtern kann. Außerdem sieht der Admin sechs kleine Anzeigen mit folgenden Informationen

- **Anzahl der aktuellen Ausleihen**
- **Anzahl der überfälligen Ausleihen**
- **Anzahl der Offenen Anfragen**
- **Anzahl der bestätigten Ausleihen, die noch nicht abgeholt wurden**
- **Anzahl der Ausleihen die in der Zukunft liegen**
- **Gesamtanzahl der Buchungen**

Beim Klicken auf die einzelnen Buchungen, wird die Detailseite für die Buchungen aufgerufen, auf welcher der Admin alle Informationen sieht die in der Buchung gespeichert werden, dazu gehören:

- **Informationen zum Ausleiher**
- **Informationen zum Ausleihzeitraum**
- **Informationen zum Verleiher**
- **Informationen zum Gegenstand**
- **Der gesamte Nachrichten bzw. Kommunikationsverlauf**
- **Timeline des Buchungsverlaufs**
- **Alle Termine & Daten aus dem Buchungsverlauf**

Beim Klicken auf den „Statistiken“ Button wird die Statistikseite geöffnet. Auf dieser kann ein Zeitraum ausgewählt werden, für welchen die Statistiken erstellt werden sollen, dabei kann man aus vordefinierten Zeiträumen aus wählen sowie benutzerdefinierten. Die Statistiken zeigen:

- **Gesamtanzahl der Buchungen**
- **Anzahl an verschiedenen Gegenständen die ausgeliehen wurden**

- Anzahl der Ausleihen des Produkts, welches am häufigsten ausgeliehen wird
- Graphische Darstellung der Verteilung des aktuellen Buchungsstatus, sowie ein tabellarische aufzählung dieser Verteilung
- Top zehn meistausgeliehene Produkte mit Anzahl der Ausleihen

Die Statistiken können als HTML oder PDF Datei exportiert werden. Für die Implementierung des PDF-exports wurde die Bibliothek jsPDF verwendet, welche unter der MIT Lizenz frei zu benutzen ist und eine Open Source Bibliothek darstellt.

10.1.5.1 Änderungshistorie

Name	Menü Komponente
Beschreibung	Erstellung einer Menü-Komponente zur einheitlicher Darstellung, der Menüpunkte.
Begründung	Verbesserung der Übersichtlichkeit und Vereinheitlichung der Darstellung aller Administrationsfunktionen.
Auswirkungen	Schnellere Produktpflege und konsistentes UI-Design.

Name	Änderung des Menü-Designs
Beschreibung	Änderung des Designs von Quadratischen Kacheln zu flachen schmalen Kacheln.
Begründung	Es passen mehr kleinere Kacheln auf den Screen.
Auswirkungen	Schnellere Orientierung.

Während der Entwicklung wurden folgende Änderungen am Menü durchgeführt:

- Verwendung der Tabellen-Komponenete zur einheitlicher Darstellung von Produkten in der Tabelle
- Verwendung einheitlicher selbst erstellter Button-Components,
- Verwendung von selbst erstellter Komponenten,

Die Änderungen sollen dem Administrator eine einheitliche Übersicht über aller Buchungen bieten, sowie Buchungsverläufe und -statistiken graphische darstellen. Außerdem soll durch das Verwenden von Komponenten ein einheitliches Design gewährleistet werden.

10.1.6 Admin-Menü: Standorte Verwalten

Das Menü SStandorte verwalten"bietet dem Admin die Funktion Standorte an denen Gegenstände gelagert werden hinzu zufügen. Das Menü listet alle Standorte tabellarisch auf mit folgenden Spalten:

- **ID:** ID des Standortes.
- **Raumnummer:** Hochschulkürzel der Räume.
- **Erstellt:** Erstellungsdatum.
- **Aktualisiert:** Aktualisierungsdatum.
- **Aktionen:** Button zum Löschen.

Über der Tabelle befindet sich ein Eingabe Feld und ein „Erstellen“ Button, mit welchem die Location hinzugefügt wird.

10.1.7 Admin-Menü: Gruppen verwalten

Das Menü "Gruppen verwalten" erlaubt es dem Admin Gruppen für Studentenprojekte zu erstellen, bearbeiten und löschen. Im Menü werden die Gruppen tabellarische aufgelistet, mit folgenden Spalten

- **Name:** Bezeichnung der Gruppe.
- **Beschreibung:** Beschreibung der Gruppe.
- **Mitglieder:** Anzahl der Mitglieder.
- **Erstellt:** Erstellungsdatum.
- **Aktionen:** Button zum Löschen und Bearbeiten.

Mit dem Button „Neue Gruppe“ wird das Formular für die Gruppenerstellung geöffnet, im welchem der Admin folgende Felder ausfüllen kann:

- **Name:** Bezeichnung der Gruppe.
- **Beschreibung:** Beschreibung der Gruppe.
- **Budget:** Betrag in Euro, welcher der Gruppe bereitgestellt wird.

Der Vorgang kann mit dem Gruppe speichern Button abgeschlossen bzw. mit dem Abbrechen Button abgebrochen werden. Das Bearbeitungsmenü ist analog aufgebaut. Beim Klicken auf eine der Gruppen in der Tabelle kommt man zur Detailansicht, in welcher die Gruppeninformationen aufgelistet werden sowie eine tabellarische Aufzählung aller Gruppenmitglieder, mit den folgenden Spalten

- **User ID:** UserID des Gruppenmitglieds.
- **Name:** Name des Users.

- **Eigentümer:** Boolean wer der Eigentümer ist (Ja oder -).

- **Aktionen:** Button zum Entfernen des Gruppenmitglieds

Über der Tabelle befindet sich der
Mitglieder hinzufügen Button, mit welchem man Gruppenmitglieder hinzufügen kann.

10.1.8 Admin-Feature: Kategorienverwaltung

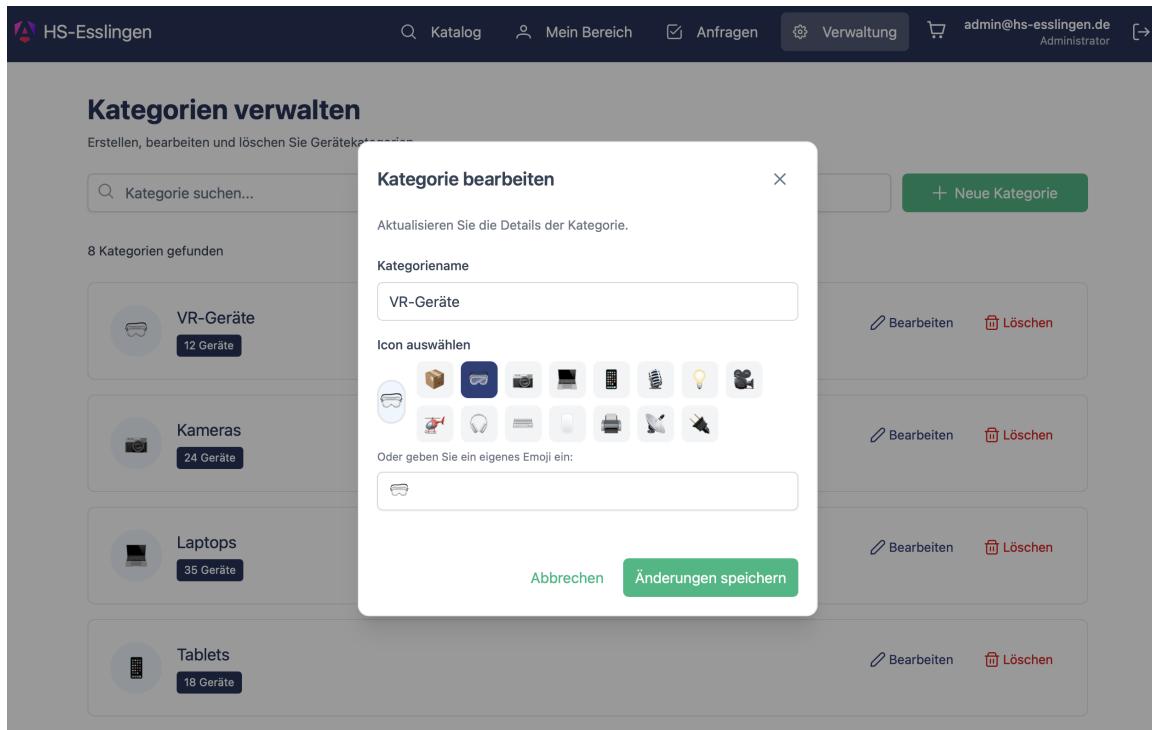


Abbildung 35: Kategorienverwaltung im Admin-Dashboard

Die Kategorienverwaltung ist Teil des Admin-Dashboards und ermöglicht es Administratoren, Gerätekategorien im System übersichtlich zu verwalten. Das Feature wurde mit Angular umgesetzt und nutzt **PrimeNG**-Komponenten für die Benutzeroberfläche.

10.1.8.1 Funktionen

- Anzeige aller vorhandenen Kategorien
- Suchfunktion zur Filterung der Kategorienliste
- Erstellen neuer Kategorien über eine Eingabe
- Bearbeiten bestehender Kategorien (Name & Icon)
- Löschen einer Kategorie, sofern keine Geräte zugeordnet sind

10.1.8.2 Datenmodell

Die Kategorien sind typisiert über ein Interface mit folgendem Aufbau:

```
interface Category {  
    id: string;  
    name: string;  
    icon: string;  
    deviceCount: number;  
}
```

Jede Kategorie besitzt einen Identifier, einen Namen, ein Icon (in diesem Fall als Emoji) und die Anzahl der verknüpften Geräte. Besonders `deviceCount` ist relevant, da Kategorien nicht gelöscht werden dürfen, wenn noch Geräte enthalten sind.

Zum Testen werden Beispiel-Daten genutzt:

```
categories: Category[] = [  
    { id: '1', name: 'VR-Geräte', icon: '', deviceCount: 12 },  
    { id: '2', name: 'Kameras', icon: '', deviceCount: 24 },  
    ...  
];
```

10.1.9 Admin-Feature: Verleiher-Zuordnung

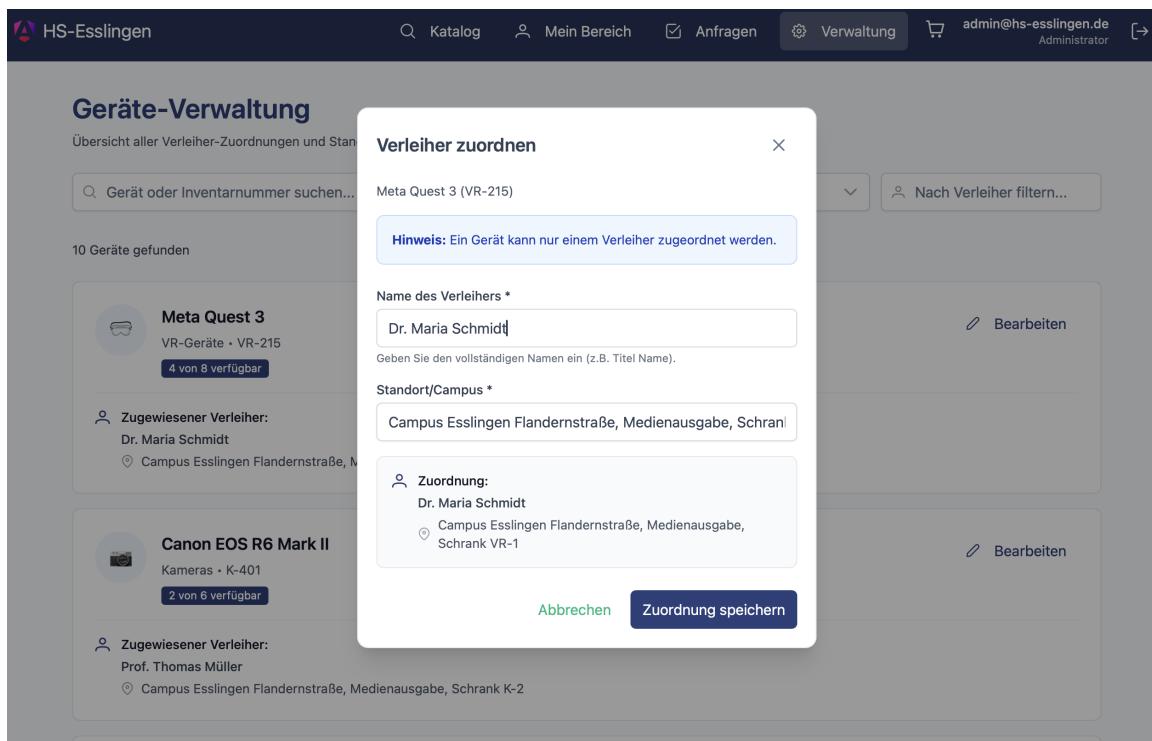


Abbildung 36: Dialog zur Verleiher-Zuordnung in der Geräteverwaltung

Die Verleiher-Zuordnung im Admin-Bereich ermöglicht es, Geräte im System zu verwalten und jedem Gerät einen Verleiher inklusive Standort/Campus zuzuordnen. Auf der Geräteliste werden Name, Kategorie, Inventarnummer, Verfügbarkeit und aktueller Verleiher angezeigt.

Über einen **Bearbeiten**-Button pro Gerät kann ein Dialog geöffnet werden (siehe Abbildung 36), in welchem der Name des Verleiher sowie der Standort eingetragen oder geändert werden kann. Unterhalb der Felder wird eine Zusammenfassung der Zuordnung angezeigt, bevor diese gespeichert wird.

10.1.9.1 Funktionen

- Übersicht aller Geräte (z. B. *Meta Quest 3, Canon EOS R6 Mark II*)
- Suchfeld zur Suche nach Gerätenamen oder Inventarnummern
- Filter nach Kategorie (VR-Geräte, Kameras, Laptops, usw.)
- Optionaler Filter nach Verleihernamen
- Bearbeiten und Ändern der Verleiher-Zuordnung über Dialog
- Aktualisierung der Geräteliste nach dem Speichern

10.1.9.2 Backend-Implementierung

Um eine flexible und dezentrale Verwaltung des Inventars zu ermöglichen, wurde die Zuordnung von Verantwortlichkeiten direkt auf die Ebene der physischen Exemplare (**Item**) verlagert. Anstatt Produkte nur global einem Administrator zuzuweisen, erlaubt das Backend nun, jedes einzelne Gerät einem spezifischen Verleiher zuzuordnen. Dies ist essenziell, um Bestände desselben Produkts auf verschiedene Verantwortliche oder Standorte verteilen zu können. Technisch wurde dies durch eine Erweiterung des Datenmodells realisiert. Die **Item**-Entität erhielt eine direkte **Many-to-One**-Beziehung zur **User**-Entität. Diese Verknüpfung ist optional, sodass Geräte auch ohne expliziten Verleiher existieren können.

Die Verwaltung dieser Zuordnung erfolgt über den **ItemController**, der bei Erstellungs- und Änderungsoperationen eine optionale **lenderId** entgegennimmt:

- **POST /api/items**: Beim Anlegen neuer Exemplare kann optional eine **lenderId** übergeben werden, um das Gerät initial einem Verleiher zuzuordnen.
- **PUT /api/items/{id}**: Ermöglicht das nachträgliche Ändern oder Entfernen der Verantwortlichkeit, beispielsweise wenn ein Gerät von einem Verleiher an einen anderen übergeben wird.

Die Validierungslogik ist im `ItemService` gekapselt. Bevor eine Zuordnung in der Datenbank persistiert wird, prüft der Service die übergebene `lenderId` gegen das `UserRepository`. Existiert der referenzierte Nutzer nicht, wird eine Exception ausgelöst und die Operation abgebrochen. Ungültige Zuweisungen werden so frühzeitig abgefangen, um Dateninkonsistenzen zu vermeiden.

Für die Darstellung im Dashboard greift das Repository auf angepasste JPQL-Queries zurück. Diese erlauben es, Bestandslisten effizient nach der `lender`-Beziehung zu filtern. So kann das Backend performant Abfragen wie „Zeige alle Geräte von Verleiher XY“ bedienen, ohne dass im Speicher über alle Items iteriert werden muss. Durch diese Architektur bleibt die Verwaltung der physischen Assets entkoppelt von der Definition der abstrakten Produkte.

10.2 Verleiher-Menü

Das Menü für den Verleiher befindet sich oben rechts im Header unter dem Punkt **Änfragen**, dieses Menü kann man ausschließlich aufrufen, wenn einem die Administrator Rolle oder die Lender Rolle zugewiesen wurde. Im Menü befinden sich alle Funktionen die ausschließlich dem Administrator oder Verliehern vorbehalten sind und werden in den folgenden Abschnitten näher beschrieben. Die Menüpunkte sind Listenartig angeordnet und jeder Menüpunkt enthält:

- **Icon**
- **Titel,**
- **Beschreibung des Menüs.**

Im Verleiher Menü befindet sich außerdem ein Button QR-Code scannen unter dem Punkt Schnellzugriff, mit welchem sich ein Pop-up öffnet mit folgenden Funktionen,

- **Kamera-Scan:** Nutzung der Browser-API (`BarcodeDetector`) zum Scannen des QR-Codes.
- **Manuelle Eingabe:** Ein Textfeld zur Eingabe des 8-stelligen Tokens, falls keine Kamera verfügbar ist oder der Scan fehlschlägt.

Während der Entwicklung wurden folgende Änderungen am Menü durchgeführt:

- **Erstellung einer Menü-Komponente zur einheitlicher Darstellung, der Menüpunkte**
- **Änderung des Designs von Quadratischen Kacheln zu vertikal-schmalen Kacheln,**

- Änderung des Layouts von zwei quadratischen Menükacheln in einer Zeile zu einer schmalen Menükachel pro Zeile.

10.2.1 Verleiher-Menü: Meine Gegenstände

Das Menü "Meine Gegenstände" bietet eine vollständige Übersicht über sämtliche Gegenstände die dem Verleiher zugewiesen wurden. Sie dient Verleihern dazu, alle Produktinstanzen, welche in zugewiesen wurden zu überblicken.

Im Header werden drei Kennzahlen hervorgehoben dargestellt:

- **Gesamtanzahl der Gegenstände,**
- **Anzahl der aktuell verfügbaren Gegenstände,**
- **Anzahl der ausgeliehenen Gegenstände.**

Darunter befindet sich ein globales Suchfeld, mit dem nach Produktnamen, Kategorien oder Inventarnummern gesucht werden kann.

Die Gegenstände sind nach ihren zugehörigen Produkten gruppiert. Jede Produktgruppe zeigt:

- den Produktnamen,
- die Kategorie,
- den Standort,
- die Anzahl verfügbarer Gegenstände.

Innerhalb jeder Produktgruppe werden die einzelnen Gegenstände in Tabellenform dargestellt. Die Tabelle enthält folgende Spalten:

- **Inventarnummer,**
- **Besitzer,**
- **Status** (mit visueller Hervorhebung),

Die verleiher-spezifische Darstellung ermöglicht eine schnelle Kontrolle der Systembestände, erleichtert das Auffinden bestimmter Gegenstände und unterstützt die effiziente Verwaltung großer Inventare.

Des Weiteren wird man bei einem Klick auf den Gegenstand auf eine Detail-Seite weitergeleitet. Auf dieser sieht die Details des Gegenstandes und dem übergeordneten Produkt. Außerdem werden alle Ausleihen des Gegenstandes tabellarisch aufgelistet mit folgenden Spalten:

- **Ausleihen,**

- **Von & Bis**,
- **Status** (mit visueller Hervorhebung),

Während der Entwicklung wurden folgende Änderungen am Menü durchgeführt:

- **Verwendung der selbst-erstellten Tabellen-Komponenten zur einheitlicher Darstellung von Daten in einer Tabelle (Für das Anzeigen von Ausleihen und Gegenständen)**
- **Hinzufügen von direkten Links zu den Bearbeitungsmenüs der Produkte und Gegenstände,**

Die Änderungen sollen dem Verleiher eine einheitliche Übersicht über alle Gegenstände, welche dem Verleiher zugewiesen wurden sowie Ausleihen bieten.

10.2.2 Verleiher-Menü: Private Gegenstände verleihen

Die Seite "Private Gegenstände verleihen" ermöglicht es allen Personen, welchen die Verleiher Rolle zugewiesen wurde, private Gegenstände anzulegen und diese anschließend zu verleihen.

Das Menü gliedert sich in drei Bereiche, ein Menü zur Produkterstellung, Gegenstandserstellung sowie eine Übersicht über die vom Verleiher erstellten privaten Gegenstände.

Das Produkterstellungsmenü unterscheidet sich kaum vom Menü welches der Admin verwendet, um die Produkte zu erstellen, bis auf die Ausnahmen der Standortzuweisung, da diese fest auf Privat gesetzt ist und nicht verändert werden kann.

Das Gegenstandserstellungsmenü funktioniert ebenfalls analog zum Menü des Administratoren, um Gegenstände anzulegen, mit dem Unterschied, dass der Präfix-Kürzel für die Inventarnummer konstant auf PRV gestzt ist. Außerdem wird der Ersteller automatisch dem Gegenstand als Besitzer und Verleiher zugewiesen.

Bei der Übersicht werden die privaten Gegenstände tabellarisch aufgelistet, mit folgenden Spaltenname:

- **Inventarnummer:** Private Inventarnummer des Gegenstands
- **Produkt:** Bezeichnung des Produkts
- **Status:** Aktueller Status des Gegenstandes
- **Aktionen:** Buttons zum Löschen und Bearbeiten der Gegenstände

Während der Entwicklung wurden folgende Änderungen am Menü durchgeführt:

- **Verschiebung der Funktion ins Verleiher-Menü**
- **Entfernen der Funktion aus Nutzer- und Admin-Menü**
- **Ändern der Logik, damit Gegenstände und Produkte vom Nutzer nicht mehr als JSON String erstellt werden und dem Admin über geschickt werden,damit dieser den JSON String bei seiner Seite einfügt, um die Sachen zu erstellen,**
- **Verleiher kann Produkte und Gegenstände direkt anlegen ohne Hilfe des Admins**

Die Änderungen sollen dem Verleihern die Möglichkeit geben eigene Produkte und Gegenstände anzulegen ohne auf den Admin angewiesen zu sein, der Admin kann private Gegenstände und Produkte trotzdem entfernen.

10.2.3 Verleiher-Dashboard: Backend-Implementierung

Um die spezifischen Anforderungen der Verleiher abzubilden, wurde ein eigener **Lender Controller** eingeführt. Anstatt dass das Frontend alle Buchungen lädt und selbst filtert, stellt dieser Controller bereits vorsortierte Sichten zur Verfügung. Verleiher erhalten damit direkt die für sie relevanten Ausleihen, ohne zusätzliche Filterlogik im Browser implementieren zu müssen.

Gleichzeitig erhöht dieser Ansatz die Sicherheit: Alle Abfragen sind so gestaltet, dass nur Buchungen zurückgegeben werden, bei denen der angegebene Benutzer als Verleiher hinterlegt ist. In Kombination mit der rollenbasierten Zugriffskontrolle (Rolle `lender` im Security-Konzept) wird so verhindert, dass Verleiher versehentlich auf Buchungen anderer Verantwortlicher zugreifen können.

Der **LenderController** stellt insbesondere folgende Endpunkte bereit:

- **GET /api/lenders/{id}/upcoming:** Liefert bestätigte Buchungen, bei denen ein Abholtermin festgelegt wurde, die Ausgabe aber noch nicht erfolgt ist. Diese Sicht unterstützt Verleiher dabei, anstehende Abholungen im Blick zu behalten.
- **GET /api/lenders/{id}/active:** Zeigt alle aktuell laufenden Ausleihen an, bei denen die Geräte bereits ausgegeben, aber noch nicht zurückgegeben wurden. Die Ergebnisse sind nach geplantem Rückgabetermin sortiert, sodass besonders dringende Fälle oben erscheinen.
- **GET /api/lenders/{id}/overdue:** Filtert die Ausleihen, deren geplantes Rückgabedatum überschritten ist, obwohl noch keine Rückgabe verbucht wurde. Dadurch

können überfällige Geräte gezielt nachverfolgt und ggf. Erinnerungen ausgesprochen werden.

Die eigentliche Filterung und Sortierung erfolgt im `BookingService` und im zugrunde liegenden `BookingRepository`. Dort greifen speziell zugeschnittene JPQL-Queries ausschließlich auf Buchungen zu, bei denen der entsprechende Verleiher hinterlegt ist. Technisch wird der Status einer Ausleihe (z. B. „überfällig“) nicht als eigenes Feld in der Datenbank gespeichert, sondern zur Laufzeit aus den vorhandenen Zeitstempeln (`confirmed Pickup, distribution Date, end Date, return Date`) abgeleitet.

Durch diese dynamische Statusberechnung bleiben die Regeln für die Einordnung einer Buchung an einer zentralen Stelle im Backend gebündelt. Änderungen (z. B. ein angepasstes Zeitfenster für „überfällig“) können so vorgenommen werden, ohne die Datenbankstruktur anzupassen oder mehrere Komponenten im System nachziehen zu müssen.

10.2.4 Offene Anfragen für Verleiher: Backend-Implementierung

Damit Verleiher eingehende Ausleihanfragen effizient bearbeiten können, stellt das Backend eine spezialisierte Sicht auf „offene“ Anfragen bereit. Offene Anfragen sind dabei alle Buchungen, bei denen der Verleiher noch nicht reagiert hat, also insbesondere noch keine Abholtermine vorgeschlagen wurden. Ziel ist eine kompakte Liste, in der klar erkennbar ist, welche Anfragen zeitnah entschieden werden müssen.

Technisch wird diese Übersicht über den `LenderController` bereitgestellt. Ein Verleiher ruft die offenen Anfragen über den Endpoint `GET /api/lenders/{lenderId}/bookings?status=pending` ab. Der Controller prüft den Status-Parameter und delegiert in diesem Fall an den Service:

- `getPendingBookingsByLenderId`: Liefert ausschließlich Buchungen des angegebenen Verleiher, deren Status als „ausstehend“ eingestuft wird.

Im `BookingService` werden dazu die passenden Repository-Methoden aufgerufen. Die Kernlogik liegt in den JPQL-Queries des `BookingRepository`. Für die Übersicht der offenen Anfragen gilt:

- Es werden nur Buchungen berücksichtigt, bei denen der aktuelle Benutzer als Verleiher hinterlegt ist.
- Eine Anfrage gilt als offen, wenn noch keine Abholtermine vorgeschlagen wurden und die Buchung nicht gelöscht wurde.
- Die Ergebnisse werden nach dem Erstellungszeitpunkt (`createdAt`) sortiert, sodass neu eingegangene Anfragen oben in der Liste erscheinen.

Damit erhält das Frontend bereits eine vorgefilterte Liste von „Pending“-Anfragen und muss keine zusätzliche Statuslogik im Browser nachbauen.

Für feinere Auswertungen stellt das Repository zusätzlich eine Variante mit optionaler Filterung nach Gegenstand bereit. Über diese Methode kann die Menge der offenen Anfragen auf ein bestimmtes Item eingegrenzt werden. Damit ist die Grundlage geschaffen, dass das Dashboard später auch nach einzelnen Geräten gefiltert werden kann.

Neben Filterung und Sortierung spielt die zeitliche Dringlichkeit eine wichtige Rolle. Beim Mapping der `Booking`-Entities auf `BookingDTOs` berechnet der `BookingMapper` ein zusätzliches Feld `urgent`. Eine offene Anfrage wird als dringend markiert, wenn sie seit mehr als 20 Stunden existiert und der Status weiterhin PENDING ist.

Ergänzend dazu überwacht ein Scheduler im Backend alle offenen Anfragen. PENDING-Buchungen, die länger als die konfigurierte Frist (standardmäßig 24 Stunden) unbearbeitet bleiben, werden automatisch per Soft-Delete gestrichen. Dadurch werden Gegenstände nicht unendlich lange durch vergessene Anfragen blockiert und Verleiher sehen in ihrem Dashboard nur noch relevante Einträge.

Zusammengenommen entsteht so eine Backend-Lösung, die Verleiher eine aufbereitete Liste offener Anfragen mit Datumssortierung, optionaler Eingrenzung nach Gegenstand und klarer Hervorhebung dringender Fälle zur Verfügung stellt.

10.3 Nutzer-Menü

Das Menü für den Nutzer befindet sich oben rechts im Header unter dem Punkt **"Mein Bereich"**, dieses Menü kann man jeder aufrufen, der sich in der WebApp anmeldet. Im Menü befinden sich alle Funktionen für den Nutzer und werden in den folgenden Abschnitten näher beschrieben. Die Menüpunkte sind listenartig angeordnet und jeder Menüpunkt enthält:

- **Icon**
- **Titel,**
- **Beschreibung des Menüs.**

Während der Entwicklung wurden folgende Änderungen am Menü durchgeführt:

- **Erstellung einer Menü-Komponente zur einheitlicher Darstellung, der Menüpunkte**
- **Änderung des Designs von Quadratischen Kacheln zu vertikal-schmalen Kacheln,**
- **Änderung des Layouts von zwei quadratischen Menükacheln in einer Zeile zu einer schmalen Menükachel pro Zeile.**

10.3.1 Nutzer-Menü: Meine Buchungen

Die Seite "Meine Buchungen" bietet dem Nutzer die Möglichkeit alle Buchungen zu sehen, die er erstellt hat. Die Buchungen werden tabellarisch aufgelistet beginnend mit den neusten Buchungen. Die Tabelle hat folgende Spalten:

- **Produkt:** Bezeichnung des Produkts.
- **Iventarnummer:** Inventarnummer des Gegenstandes.
- **Ausleiher:** Name des Ausleiher.
- **Verleiher:** Name des Verleiher.
- **Status:** Aktueller Status der Buchung.
- **Abholung:** Beginn der Ausleihe als Datum.
- **Rückgabe:** Ende der Ausleihe als Datum.
- **Erstellt am:** Erstellungsdatum.

Über der Tabelle befindet sich eine Suchleiste, mit welcher der Admin die Buchungen filtern kann. Der Nutzer sieht außerdem im Header die drei folgenden Statistiken:

- **Anzahl der aller Buchungen des Nutzers**
- **Anzahl der aktiven Buchungen**
- **Anzahl der abgeschlossenen Buchungen**

Beim Klicken auf die einzelnen Buchungen, wird die Detailseite für die Buchungen aufgerufen, auf welcher der Admin alle Informationen sieht die in der Buchung gespeichert werden, dazu gehören:

- **Informationen zum Ausleihzeitraum**
- **Informationen zum Verleiher**
- **Informationen zum Gegenstand**
- **Der gesamte Nachrichten bzw. Kommunikationsverlauf**
- **Timeline des Buchungsverlaufs**
- **Alle Termine & Daten aus dem Buchungsverlauf**

Der Nutzer kann außerdem die Buchung stornieren. Falls die Buchung bestätigt wurde 3 Termine mit Uhrzeiten vorschlagen, an welchen der Nutzer den Gegenstand abholen kann, sowie Termine des Verleiher akzeptieren und eine Nachricht verfassen.

Bei der Abholung und Rückgabe der Gegenstände kann der Nutzer einen QR-Code vorzeigen, welcher vom Verleiher eingescannt werden kann, um die Abholung, sowie Rückgabe zu bestätigen, alternativ wird ein 8-stelliger Code generiert, welcher vom Verleiher ebenfalls verwendet werden kann um die Anholung und Rückgabe zu bestätigen. Die Generierung des QR-Codes erfolgt mit der Bibliothek Angularx-qrcode, welche mit der MIT Lizenz zu den Open Source Bibliotheken gehört und frei genutzt werden kann. Genaure Logik wird in folgenden Abschnitten genauer erklärt

Außerdem kann der Nutzer den aktuellen Stand der Buchung als PDF exportieren. Die Implementierung erfolgt über die Bibliothek jsPDF, welche ebenfalls unter die MIT Lizenz fällt.

10.3.2 Mein Gruppen

Die Seite "Meine Gruppen" erlaubt es dem Nutzer Gruppen für Studentenprojekte zu erstellen, bearbeiten und zu löschen. Im Menü werden die eigenen Gruppen tabellarisch aufgelistet, mit folgenden Spalten

- **Gruppenname:** Bezeichnung der Gruppe.
- **Beschreibung:** Beschreibung der Gruppe.
- **Mitglieder:** Anzahl der Mitglieder.
- **Erstellt am:** Erstellungsdatum.
- **Aktionen:** Button zum Löschen und Bearbeiten.

Mit dem Button „Neue Gruppe“ wird das Formular für die Gruppenerstellung geöffnet, im welchem der Admin folgende Felder ausfüllen kann:

- **Name:** Bezeichnung der Gruppe.
- **Beschreibung:** Beschreibung der Gruppe.
- **Budget:** Betrag in Euro, welcher der Gruppe bereitgestellt wird.

Der Vorgang kann mit dem

Gruppe speichern Button abgeschlossen bzw. mit dem

Abbrechen Button abgebrochen werden. Das Bearbeitungsmenü ist analog aufgebaut.

Beim Klicken auf eine der Gruppen in der Tabelle kommt man zur Detailansicht, in welcher die Gruppeninformationen aufgelistet werden sowie eine tabellarische Aufzählung aller Gruppenmitglieder, mit den folgenden Spalten

- **User ID:** UserID des Gruppenmitglieds.
- **Name:** Name des Users.
- **Eigentümer:** Boolean wer der Eigentümer ist (Ja oder -).
- **Aktionen:** Button zum Entfernen des Gruppenmitglieds

Mitglieder können über einen QR-Code hinzugefügt werden, der Scanner für den QR-Code befindet sich, in der Übersicht über die eigenen Gruppen

Während der Entwicklung wurden folgende Änderungen am Menü durchgeführt:

- **Funktionalität für die Erstellung von Gruppen, auch zum Nutzer-Bereich hinzugefügt, davor ausschließlich Admin-Bereich**
- **Hinzufügen des Beitriffs zur Gruppe mit QR-Code, davor manuell**

10.4 System-Funktionen

Dieses Kapitel beschreibt zentrale serverseitige Funktionen, mit denen das System den Leihprozess unterstützt und die Kommunikation zwischen Verleiher und Entleiher automatisiert. Dazu gehören die Bestätigung von Abholungen und Rückgaben sowie E-Mail-Benachrichtigungen bei Statusänderungen und anstehendem oder überfälligem Rückgabetermin.

10.4.1 Bestätigung der Abholung

Sobald ein Verleiher die Ausgabe eines Gegenstandes im System bestätigt – sei es manuell über das Dashboard oder durch das Scannen des QR-Tokens, löst das System automatisch den Prozess zur Bestätigung der Abholung aus.

Technisch wird hierbei im `BookingService` der Status der Buchung auf `PICKED_UP` gesetzt und der aktuelle Zeitstempel als `distributionDate`persistiert. Parallel dazu triggert das System über den `EmailService` eine asynchrone Benachrichtigung an den Studierenden. Diese E-Mail dient als digitaler Beleg für den Erhalt des Geräts und enthält alle relevanten Informationen wie den genauen Abholzeitpunkt, das Gerät sowie das vereinbarte Rückgabedatum. Der Versand erfolgt mittels `@Async`, sodass der Bestätigungs vorgang an der Ausgabetheke nicht durch die E-Mail-Erstellung blockiert wird.

10.4.2 Dokumentation der Rückgabe

Analog zur Abholung übernimmt das System die lückenlose Dokumentation der Rückgabe. Wenn ein Gegenstand zurückgebracht wird, aktualisiert der `BookingService` den Buchungsstatus auf `RETURNED` und setzt das `returnDate`.

Dieser Vorgang ist essenziell für die Einhaltung der Audit-Integrität des Systems. Nach erfolgreicher Aktualisierung generiert das System über den `EmailService` eine Rückgabebestätigung per E-Mail an den Studierenden. Dies schafft Transparenz und Sicherheit für beide Seiten, da der Abschluss des Leihvorgangs offiziell protokolliert ist und keine weiteren Forderungen bestehen. Auch hier erfolgt der Versand der E-Mail entkoppelt vom Hauptthread (`@Async`), um die Wartezeit an der Ausgabetheke so gering wie möglich zu halten.

10.4.3 Benachrichtigung bei Statusänderungen

Das System dient als aktiver Vermittler zwischen Verleiher und Entleiher und informiert die Beteiligten bei relevanten Statusänderungen einer Buchung. Jede Statusänderung löst ein entsprechendes Event aus, das in eine E-Mail-Benachrichtigung übersetzt wird. Typische Fälle sind:

- **Anfrage Bestätigung:** Akzeptiert ein Verleiher eine Anfrage, wird der Status beispielsweise von `PENDING` auf `CONFIRMED` gesetzt. Der Studierende erhält eine Benachrichtigung inklusive der bestätigten Abholzeiten und der Geräteinformationen.
- **Ablehnung oder Stornierung:** Wird eine Anfrage abgelehnt (`REJECTED`) oder eine bestehende Buchung storniert (`CANCELLED`), informiert das System den betroffenen Nutzer über den Vorgang.

Für die Generierung der E-Mails werden HTML-Templates verwendet, die dynamisch mit den Buchungsdaten (Gerätename, Zeiträume, Status) befüllt werden. Die technische Umsetzung erfolgt über den zentralen `EmailService`, der je nach Ereignis das passende Template auswählt und mit den konkreten Daten einer Buchung rendert.

10.4.4 Automatisierte Erinnerungen und Mahnwesen

Um Verspätungen proaktiv zu vermeiden und Verleiher bei der Überwachung laufender Ausleihen zu entlasten, verfügt das System über einen integrierten Zeitplaner (`Booking Scheduler`) sowie einen darauf aufbauenden Erinnerungs- und Mahnservice (`Reminder Service`). Diese Komponenten realisieren automatisierte Erinnerungsmails und unterstützen das Mahnwesen bei überfälligen Rückgaben.

Erinnerungsservice: Ein täglich ausgeführter Cronjob identifiziert alle laufenden Ausleihen, deren geplantes Rückgabedatum in genau 48 Stunden bevorsteht. Der `Reminder Service` erzeugt für diese Buchungen eine Erinnerungs-Mail und versendet sie über den `EmailService` an die betroffenen Studierenden. Die Nachricht weist auf die baldige Fälligkeit hin, nennt das Gerät sowie das konkrete Rückgabedatum.

Eskalationsmanagement: Erkennt der Scheduler, dass ein Rückgabedatum überschritten wurde, ohne dass der Status auf RETURNED gesetzt ist, greift das Mahnwesen. Der **ReminderService** versendet eine Eskalations-Benachrichtigung an den Studierenden, in der auf die überfällige Rückgabe hingewiesen wird. Um eine schnelle Klärung zu ermöglichen, wird bei dieser Art der Benachrichtigung der zuständige Verleiher automatisch in Kopie (CC) gesetzt. Dies stellt sicher, dass Verleiher frühzeitig über säumige Rückgaben informiert sind, ohne manuell Listen prüfen zu müssen, und schafft gleichzeitig eine nachvollziehbare Kommunikationsspur für spätere Auswertungen.

10.5 Warenkorb

Der Warenkorb stellt eine Benutzeroberfläche bereit um gesammelt gewünschte Ausleihanfragen abzusenden. Dazu wird zunächst ein Produkt auf der jeweiligen Produkt-Detailseite aufgerufen. Auf der rechten Seite kann das gewünschte Abholdatum ausgewählt werden. Mit einem Click auf SZum Warenkorb hinzufügen "Kann das Produkt zum Warenkorb hinzugefügt werden.

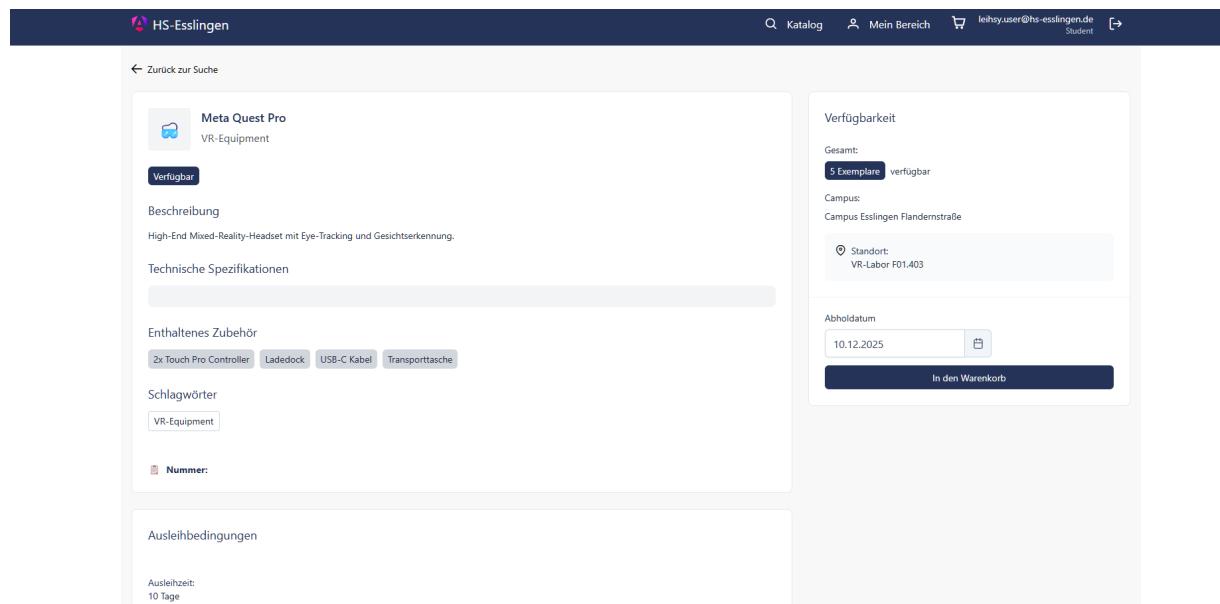


Abbildung 37: Produkt-Detailseite

Dabei wird auch das gewählte Datum in den Warenkorb übertragen. Im Header der Seite zeigt das Warenkorb-Icon ein Badge mit der Anzahl der Produkte die sich im Warenkorb befinden und der Button SZum Warenkorb hinzufügen" wird grün und ändert den Text in SZum Warenkorb hinzugefügt".

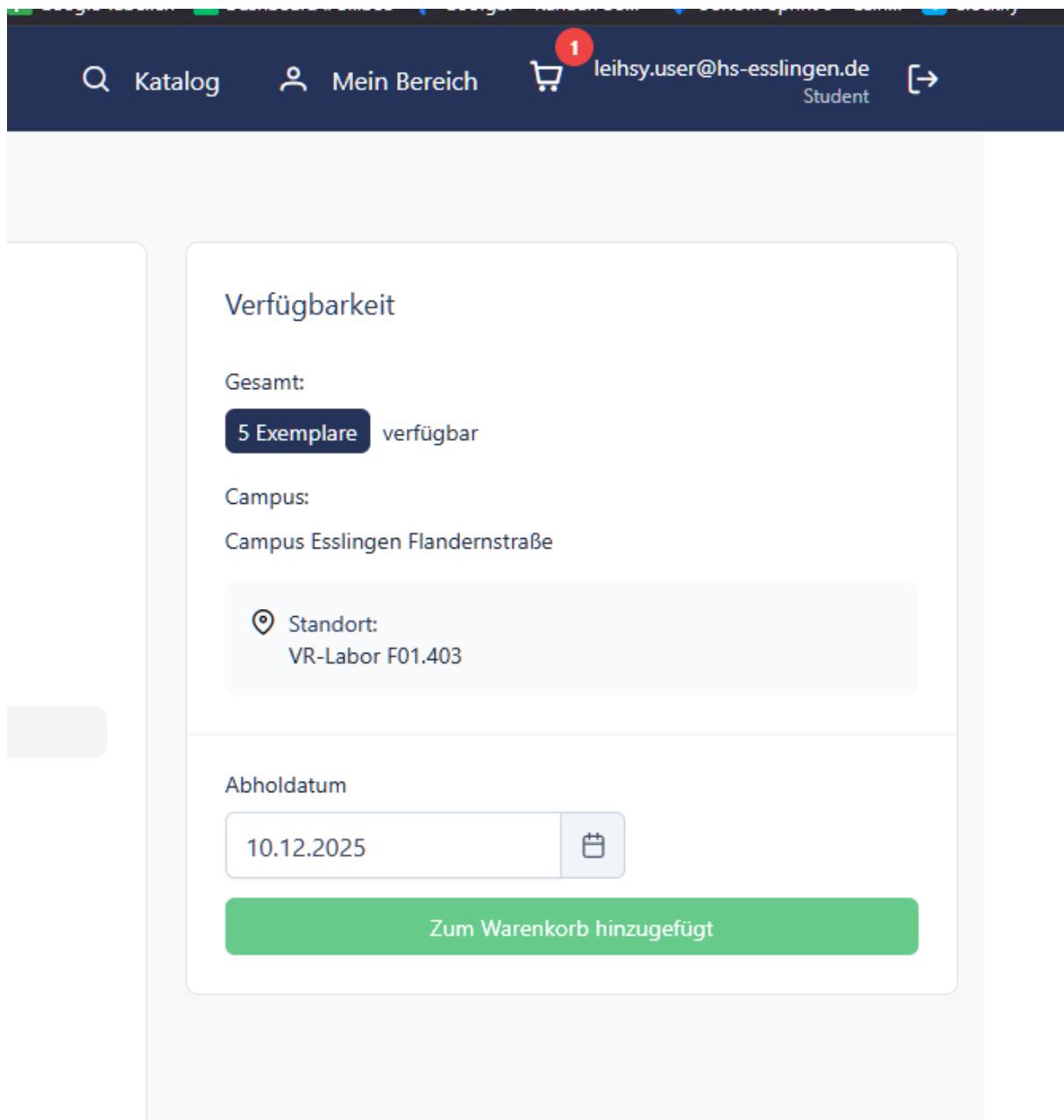


Abbildung 38: Zum Warenkorb Hinzugefügt

Aus technischer Sicht wird bei dem Click auf SZum Warenkorb hinzufügen" die Funktion addItem des Cart-Services aufgerufen. Dieser speichert die Produkt-ID, das Abholdatum und die neu generierte Warenkorb-ID des Items (die dazu dient Einträge im Warenkorb zu unterscheiden) im lokalen Speicher des Browsers. Dies sorgt für eine verbesserte Latenz und verringert den Verarbeitungsaufwand im Backend. Außerdem wird die Observable itemCount aktualisiert die beispielsweise vom Header genutzt wird um das Badge mit der Anzahl der Produkte im Warenkorb immer aktuell und automatisch auch über mehrere Tabs hinweg synchronisiert anzuzeigen. Mit einem Click auf das Warenkorb-Icon im Header gelangt man auf die Warenkorb-Seite. Hier werden alle Produkte angezeigt die sich

aktuell im Warenkorb befinden sowie das zuvor ausgewählte Abholdatum.

The screenshot shows the HS-Esslingen library website's shopping cart page. At the top, there is a dark header bar with the HS-Esslingen logo, a search icon, a user menu with the email 'leihsysuser@hs-esslingen.de' and status 'Student', and a sign-out icon. Below the header, the page title 'Warenkorb' is displayed with a count of '1'. The main content area contains a product card for 'Meta Quest Pro'. The card includes the product name, inventory number '1', a category dropdown, and a red trash can icon in the top right corner. Below the card, there are two sections: 'Abholort:' with the address 'Campus Esslingen Flandernstraße VR-Labor F01.403' and 'Abholdatum:' with a date input field set to '10.12.2025' and a calendar icon. A box labeled 'Ausleihbedingungen:' contains the text 'Ausleihfrist: 10 Tage' and 'Verlängerungen: Nach Verfügbarkeit'. At the bottom of the page, there is a summary section titled 'Gesamtanzahl:' showing '1 Produkt(e)'. It includes a checkbox for accepting terms and conditions, a 'Weiter suchen' button, and a large blue 'Ausleihanfrage abschicken' button.

Abbildung 39: Warenkorb-Seite

In der rechten oberen Ecke jedes Produkts befindet sich ein roter Mülleimer, mit dessen Anklicken das entsprechende Produkt aus dem Warenkorb gelöscht wird. Dazu wird im Cart-Service die Funktion `removeItem` aufgerufen die den entsprechenden Eintrag anhand seiner Warenkorb-ID aus dem lokalen Browser-Speicher löscht und den `itemCount` aktualisiert. Es werden weitere Informationen zu dem jeweiligen Produkt angezeigt, beispielsweise die Inventarnummer, der Abholort an dem sich das Produkt befindet und die maximale Ausleihdauer. Unter den Produkten befindet sich eine Checkbox mit der der Entleiher versichert, das Produkt ünbeschädigt, vollständig und rechtzeitig zurückzugeben". Nur wenn diese Zusicherung erteilt wurde kann der Button zum Absenden der Ausleihanfrage(n) betätigt werden.

This screenshot shows a simplified version of the shopping cart page, likely a confirmation or summary step. It features a summary box with the text 'Gesamtanzahl: 1 Produkt(e)'. Below it is a checkbox with the text 'Ich akzeptiere die Ausleihbedingungen und verpflichte mich, den/die Artikel unbeschädigt, vollständig und rechtzeitig zurückzugeben.' followed by a checked checkbox icon. At the bottom, there are two buttons: a white 'Weiter suchen' button and a dark blue 'Ausleihanfrage abschicken' button.

Abbildung 40: Warenkorb-Checkout

Dieser sendet für jedes Produkt im Warenkorb einzeln eine Ausleihanfrage an das Backend. Somit vereinfacht sich die Behandlung der Ausleihanfragen im Backend verglichen mit Sammelaufträgen oder Ähnlichem. Nach dem Absenden der Ausleihanfragen wird der Warenkorb mit der Funktion clearCart des Cart-Services im lokalen Browser-Speicher gelöscht.

10.5.1 Benutzeroberfläche

Abbildung 41 zeigt die Benutzeroberfläche der Buchungsübersicht. Hier werden alle bestehenden, ausstehenden oder stornierten Buchungen eines Benutzers tabellarisch dargestellt. Über ein zentrales Suchfeld können Buchungen nach Produktnamen, Inventarnummer, Verleiher oder Status gefiltert werden.

Die Tabelle enthält folgende Spalten:

- **Produkt:** Anzeigename des gebuchten Geräts.
- **Inventarnummer:** Eindeutige Kennung des physischen Geräts.
- **Verleiher:** Verantwortlicher Nutzer bzw. Administrator.
- **Status:** Buchungsstatus wie „Ausstehend“ oder „Storniert“.
- **Abholung / Rückgabe:** Geplante Zeiträume der Leihe.
- **Erstellt am:** Erstellungszeitpunkt der Buchung.

Am unteren Rand befindet sich eine Pagination, über die zwischen mehreren Buchungsseiten navigiert werden kann.

PRODUKT	INVENTARNUMMER	VERLEIHER	STATUS	ABHOLUNG	RÜCKGABE	ERSTELLT AM
Meta Quest Pro	VR-PRO-001	admin@hs-esslingen.de	Ausstehend	10.12.2025	15.12.2025	04.12.2025, 13:05
Meta Quest Pro	VR-PRO-001	admin@hs-esslingen.de	Storniert	10.12.2025	15.12.2025	04.12.2025, 16:48
Meta Quest Pro	VR-PRO-001	admin@hs-esslingen.de	Storniert	10.12.2025	15.12.2025	04.12.2025, 16:48

Abbildung 41: Buchungsverlauf eines Users

10.5.2 Backend-Implementierung

Dieses Kapitel dokumentiert die Implementierung des Buchungssystems im LeihSy-Backend. Der Fokus liegt auf der Umstellung von Entity-basierten Responses auf das DTO-Pattern, der korrekten Status-Verwaltung und der Integration mit dem Security-System.

10.5.2.1 DTO-Pattern und Circular Reference

Ausgangssituation:

Der BookingController gab Booking-Entities direkt als Response zurück. Dies führte zu einem Circular Reference Problem: Jackson folgte allen bidirektionalen JPA-Beziehungen (Booking → Item → Product → Items → Bookings) und generierte JSON-Responses mit über 8000 Zeilen.

Lösungsansätze:

Zwei Ansätze wurden evaluiert:

Option A - JsonIgnoreProperties: Annotations direkt an Entity-Feldern würden die Serialisierung steuern. Nachteil: Vermischt Persistenz-Logik mit Präsentations-Logik.

Option B - DTO-Pattern (gewählt): Entities bleiben sauber, DTOs enthalten nur benötigte Felder als flache Struktur. MapStruct übernimmt automatisches Mapping zwischen Entity und DTO.

Durchgeführte Änderungen:

- BookingDTO mit Lombok-Annotations optimiert (@Data, @Builder, @NoArgsConstructor, @AllArgsConstructor)
- Code-Reduktion von ca. 150 auf 30 Zeilen durch Wegfall manueller Getter/Setter
- BookingService: Alle public Methoden geben nun BookingDTO statt Booking zurück
- BookingMapper (MapStruct) bildet verschachtelte Entity-Beziehungen auf flache DTO-Felder ab
- BookingController: Alle Endpoints verwenden nun DTOs in Request/Response

Ergebnis:

JSON-Response wurde von 8000+ auf ca. 50 Zeilen reduziert. Entities bleiben frei von Serialisierungs-Logik. Klare Trennung zwischen Datenbank-Modell (Entity) und API-Modell (DTO).

10.5.2.2 Status-Verwaltung über Timestamps

Ausgangssituation:

Initial wurde versucht, den Booking-Status als Enum zu implementieren und explizit zu setzen. Dies führte zu Compile-Fehlern, da die Booking-Entity den Status als berechnetes Transient-Feld implementiert hatte.

Konzept der berechneten Status:

Der Status wird nicht in der Datenbank gespeichert, sondern zur Laufzeit aus den vorhandenen Timestamp-Feldern berechnet. Die Logik folgt einer Prioritäts-Kaskade: deletedAt → CANCELLED, returnDate → RETURNED, distributionDate → PICKED_UP, confirmedPickup → CONFIRMED, ansonsten → PENDING.

Statt einen Status zu setzen, werden die entsprechenden Timestamp-Felder manipuliert:

Gewünschter Status	Aktion
PENDING	Alle Timestamps auf NULL
CONFIRMED	confirmedPickup setzen
PICKED_UP	distributionDate setzen
RETURNED	returnDate setzen
CANCELLED	deletedAt setzen

Tabelle 5: Status-Steuerung über Timestamps

Vorteile dieses Ansatzes:

- Single Source of Truth: Status ergibt sich eindeutig aus Timestamps
- Automatischer Audit-Trail für alle Status-Wechsel
- Keine Inkonsistenzen durch manuelle Status-Pflege möglich
- Vereinfachte Datenbank-Queries

10.5.2.3 Architektur-Entscheidungen

DTO-Pattern als Standard:

Die konsequente Verwendung von DTOs für alle API-Responses verhindert nicht nur Circular Reference Probleme, sondern bietet weitere Vorteile: Trennung von Datenbank- und API-Struktur, gezielte Datenauswahl pro Endpoint, Schutz interner Strukturen vor Breaking Changes und bessere Performance durch reduzierte Datenmenge.

Berechnete Status statt gespeicherter Enum:

Die Entscheidung für berechnete Status basierend auf Timestamps statt eines gespeicherten Enum-Wertes eliminiert mögliche Inkonsistenzen. Jeder Status-Wechsel ist automatisch mit einem Zeitstempel dokumentiert (Audit-Trail) und die Logik ist zentralisiert in der Entity-Klasse.

MapStruct für DTO-Mapping:

MapStruct wurde gewählt, da es zur Compile-Zeit Code generiert (keine Reflection zur Laufzeit) und typsicher ist. Die Annotation-basierte Konfiguration ist übersichtlich und ermöglicht komplexe Mappings wie verschachtelte Entity-Beziehungen auf flache DTO-Felder.

Lombok zur Reduktion von Boilerplate:

Unter Einsatz von Lombok (siehe Abschnitt 5.2.2) wurden die DTOs und Services mit Annotations wie `@Data`, `@Builder` und `@RequiredArgsConstructor` versehen. Dies reduzierte den Code in den Booking-Komponenten um ca. 80% und verbesserte die Lesbarkeit erheblich.

API-Dokumentation mit Swagger:

Der BookingController wurde mit vollständigen OpenAPI-Annotations versehen: `@Tag` für Gruppierung, `@Operation` für Endpoint-Beschreibungen, `@ApiResponses` für Status-Codes und `@Schema` für Request-DTOs. Die Dokumentation ist unter `/swagger-ui.html` verfügbar.

10.5.2.4 Zusammenfassung

Komponente	Vorher	Nachher
BookingDTO	Manuelle Getter/Setter (ca. 150 Zeilen)	Lombok Annotations (ca. 30 Zeilen)
BookingService	Return Type: Booking Entity	Return Type: BookingDTO
BookingController	Entities in Response führten zu Circular Reference	DTOs liefern sauberes, flaches JSON
Status-Verwaltung	Versuch Enum zu setzen führte zu Compile-Fehler	Timestamps setzen, Status wird berechnet
UserService	getCurrentUser() Methode fehlte	Implementiert mit Auto-User-Creation
API-Dokumentation	Keine Swagger-Annotations	Vollständige OpenAPI-Dokumentation

Tabelle 6: Übersicht aller Änderungen am Booking-System

10.6 Authentifizierung und Benutzerverwaltung

10.6.1 Überblick & Architektur

Das LeihSy-System implementiert eine moderne OAuth2-basierte Authentifizierung mit JSON Web Tokens (JWT). Die Authentifizierung erfolgt über den zentralen Keycloak-Server der Hochschule Esslingen, was eine nahtlose Integration mit bestehenden RZ-Accounts ermöglicht.

10.6.1.1 Komponenten

Die Security-Architektur besteht aus drei Hauptkomponenten:

- **Keycloak Identity Provider:** Zentrale Authentifizierungsstelle der Hochschule Esslingen unter `auth.insy.hs-esslingen.com`. Verwaltet Benutzerkonten, Rollen und erstellt JWT-Tokens nach erfolgreichem Login.
- **Angular Frontend:** Übernimmt die Token-Verwaltung im Browser. Nutzt die `keycloak-js` Bibliothek für den OAuth2 Authorization Code Flow mit PKCE.
- **Spring Boot Backend:** Validiert eingehende JWT-Tokens und prüft Berechtigungen. Konfiguriert als OAuth2 Resource Server.

10.6.1.2 Authentifizierungsablauf

Der Login-Prozess folgt dem OpenID Connect (OIDC) Standard:

1. Benutzer ruft Frontend auf (`http://localhost:4200`)
2. Angular leitet zur Keycloak Login-Seite weiter
3. Benutzer authentifiziert sich mit RZ-Account
4. Keycloak erstellt JWT-Token und sendet es an Frontend
5. Frontend speichert Token im Browser (SessionStorage)
6. Bei jedem Backend-Request wird Token im Authorization-Header mitgesendet
7. Backend validiert Token und extrahiert Benutzerinformationen

10.6.2 JWT-Token und Security-Konfiguration

Ein JSON Web Token besteht aus drei Base64-kodierten Teilen (Header, Payload, Signature):

```
{  
  "sub": "a980c70e-...",           // Keycloak User-ID  
  "preferred_username": "admin",    // Username  
  "email": "admin@hs-esslingen.de",  
  "realm_access": {  
    "roles": ["admin", "user"]      // Zugewiesene Rollen  
  },  
  "iss": "https://auth.insy.hs-esslingen.com/realms/insy",  
  "exp": 1733331200                // Ablaufzeit (Unix Timestamp)  
}
```

Die Token-Signatur wird mit RSA-256 erstellt und kann vom Backend mit dem Public Key von Keycloak validiert werden.

10.6.2.1 SecurityConfig Klasse

Die zentrale Security-Konfiguration erfolgt in `SecurityConfig.java`:

- **CORS-Konfiguration:** Erlaubt Cross-Origin Requests vom Frontend (`localhost:4200`)
- **CSRF-Schutz deaktiviert:** Da JWT-Tokens im Authorization-Header übertragen werden, ist der Cookie-basierte CSRF-Schutz nicht erforderlich
- **OAuth2 Resource Server:** Konfiguriert JWT-Validierung mit Keycloak als Issuer
- **Rollen-Mapping:** Konvertiert Keycloak-Rollen (`admin`) zu Spring Security Authorities (`ROLE_admin`)

10.6.2.2 Autorisierungsregeln

Die URL-Muster werden mit spezifischen Berechtigungen versehen:

Endpoint-Muster	Berechtigung
<code>/api/products/**</code>	Öffentlich (permitAll)
<code>/api/categories/**</code>	Öffentlich (permitAll)
<code>/api/admin/**</code>	Nur Admin-Rolle
<code>/api/bookings/lenders/**</code>	Admin oder Lender-Rolle
Alle anderen	Authentifizierung erforderlich

Tabelle 7: Zugriffskontrolle nach Endpoint-Mustern

10.6.3 Rollensystem

Das System unterscheidet drei Benutzerrollen:

- **user:** Standard-Rolle für Studierende. Kann Gegenstände durchsuchen und Buchungen erstellen.
- **lender:** Rolle für Verleiher. Kann zusätzlich Buchungsanfragen bearbeiten und Ausgabe/Rückgabe dokumentieren.
- **admin:** Administrator-Rolle. Vollzugriff auf alle Funktionen inklusive Gegenstandsverwaltung und Statistiken.

Die Rollenzuweisung erfolgt zentral in Keycloak über Realm Roles. Ein Benutzer kann mehrere Rollen gleichzeitig haben.

10.6.4 Frontend-Integration

10.6.4.1 AuthService

Der AuthService kapselt die Keycloak-Integration:

- Initialisiert Keycloak-Client mit `keycloak-js` Bibliothek
- Verwaltet Token-Lifecycle (Abruf, Refresh, Ablauf)
- Stellt Methoden zur Rollen-Prüfung bereit (`hasRole()`)
- Extrahiert Benutzerinformationen aus Token-Claims

10.6.4.2 Route Guards

Angular Route Guards schützen Frontend-Routes vor unberechtigtem Zugriff. Der `AuthGuard` prüft den Login-Status, während der `RoleGuard` die erforderlichen Rollen validiert. Bei fehlender Berechtigung erfolgt eine Umleitung zur Login- oder Fehlerseite. Die grundlegende Funktionsweise von Guards im Angular-Framework wird in Abschnitt 9.2 beschrieben.

10.6.4.3 Token-Validierung

Bei jedem Backend-Request durchläuft der JWT-Token folgende Validierungsschritte:

1. **Signatur-Prüfung:** Validierung mit Keycloak Public Key (RSA-256)
2. **Issuer-Prüfung:** Token muss von konfiguriertem Keycloak stammen
3. **Ablauf-Prüfung:** `exp`-Claim darf nicht überschritten sein
4. **Rollen-Extraktion:** `realm_access.roles` wird in Spring Security Authorities konvertiert

Bei fehlgeschlagener Validierung antwortet das Backend mit HTTP 401 Unauthorized.

10.6.4.4 Automatische Benutzererstellung

Beim ersten Login eines Benutzers wird automatisch ein User-Datensatz in der lokalen Datenbank angelegt:

- `unique_id`: Keycloak Subject (UUID)
- `name`: Aus Token-Claim `preferred_username`
- `email`: Aus Token-Claim `email`

Die Methode `UserService.getCurrentUser()` extrahiert die Keycloak-ID aus dem Security Context und holt oder erstellt den entsprechenden User.

10.6.4.5 Vorteile der JWT-basierten Authentifizierung

- **Stateless:** Backend muss keine Sessions speichern
- **Skalierbar:** Jeder Backend-Server kann Token validieren
- **Single Sign-On:** Nutzer ist automatisch in allen HS-Anwendungen eingeloggt
- **Standardisiert:** OAuth2 und OIDC sind etablierte Standards

10.6.4.6 Implementierte Schutzmaßnahmen

- Token-Signaturprüfung verhindert Token-Manipulation
- Kurze Token-Laufzeit (Standard: 5 Minuten) minimiert Missbrauchsrisiko
- HTTPS erzwingt verschlüsselte Übertragung (in Produktion)
- CORS-Whitelist beschränkt erlaubte Origins

10.6.4.7 Keycloak-Konfiguration

Das System nutzt den gemeinsamen Keycloak-Server mit dem InSy-Inventarsystem:

- **Realm:** insy
- **Backend Client-ID:** insy-backend (temporär, leihsy-backend-dev geplant)
- **Frontend Client-ID:** leihsy-frontend-dev
- **Token Endpoint:** /realms/insy/protocol/openid-connect/token

Die Keycloak-Verbindungsparameter sind in `application.yml` hinterlegt und können pro Umgebung (dev/prod) angepasst werden.

10.6.5 Automatische Benutzersynchronisation

10.6.5.1 Übersicht

Dieses Kapitel dokumentiert die Implementierung der automatischen Benutzersynchronisation zwischen dem Keycloak Identity Provider und der LeihSy-Datenbank. Die Synchronisation stellt sicher, dass Benutzer automatisch in der lokalen Datenbank angelegt werden, sobald sie sich erstmals über Keycloak authentifizieren.

10.6.5.2 Ausgangssituation

Vor der Implementierung mussten Benutzer manuell in der LeihSy-Datenbank angelegt werden, bevor sie das System nutzen konnten. Dies führte zu einem Medienbruch: Obwohl die Authentifizierung über Keycloak erfolgte, existierte der Benutzer nicht in der lokalen Datenbank und konnte daher keine Buchungen durchführen.

10.6.5.3 Lösungsansatz

Die implementierte Lösung synchronisiert Benutzerdaten automatisch beim ersten Login. Dabei werden die relevanten Informationen aus dem JWT-Token extrahiert und in der lokalen Datenbankpersistiert.

10.6.5.4 Backend-Komponenten

Im Backend wurde ein neuer REST-Endpoint `GET /api/users/me` implementiert. Dieser Endpoint gibt die Daten des aktuell authentifizierten Benutzers zurück. Falls der Benutzer noch nicht in der Datenbank existiert, wird er automatisch angelegt.

Die Benutzeridentifikation erfolgt über die eindeutige Keycloak-ID (Subject Claim), die im JWT-Token enthalten ist. Zusätzlich werden der Benutzername und die E-Mail-Adresse aus dem Token extrahiert.

Ein Security-Filter wurde hinzugefügt, der bei jedem authentifizierten API-Request prüft, ob der Benutzer bereits existiert. Falls nicht, wird ein neuer Datensatz angelegt. Existiert der Benutzer bereits, werden geänderte Stammdaten wie der Name aktualisiert.

10.6.5.5 Frontend-Komponenten

Im Frontend wurde der AuthService erweitert. Dieser verwaltet nun zusätzlich zum Keycloak-Authentifizierungsstatus auch die Benutzerdaten aus der lokalen Datenbank.

Ein HTTP-Interceptor wurde implementiert, der automatisch das JWT-Token als Bearer-Token an alle API-Requests anhängt. Dies ermöglicht die serverseitige Authentifizierung ohne manuelle Token-Verwaltung in den einzelnen Services.

Die Initialisierungsreihenfolge beim Anwendungsstart wurde angepasst: Zuerst wird Keycloak initialisiert, danach erfolgt der Aufruf des `/api/users/me` Endpoints zur Benutzersynchronisation.

10.6.5.6 Race Condition bei der Initialisierung

Eine zentrale Herausforderung war die korrekte Reihenfolge der Initialisierung. Der AuthService im Frontend wurde ursprünglich vor Abschluss der Keycloak-Initialisierung instanziert, wodurch der Authentifizierungsstatus fälschlicherweise als `false` erkannt wurde.

Die Lösung besteht darin, die Initialisierungslogik aus dem Konstruktor zu entfernen und stattdessen eine explizite `initialize()`-Methode zu verwenden. Diese wird erst aufgerufen, nachdem Keycloak vollständig initialisiert wurde.

10.6.5.7 Injection Context in asynchronen Funktionen

Angular's `inject()`-Funktion kann nur synchron innerhalb eines Injection Context aufgerufen werden. In asynchronen Funktionen geht dieser Kontext nach dem ersten `await` verloren.

Die Lösung besteht darin, alle benötigten Dependencies am Anfang der Funktion zu injizieren, bevor asynchrone Operationen ausgeführt werden.

10.6.5.8 Rollenverwaltung

Die Benutzerrollen werden ausschließlich in Keycloak verwaltet und bei jedem Request aus dem JWT-Token extrahiert. Eine lokale Speicherung der Rollen in der LeihSy-Datenbank ist nicht erforderlich.

Die Extraktion erfolgt aus zwei Token-Bereichen:

- **Realm-Rollen:** Globale Berechtigungen aus dem Claim `realm_access.roles`
- **Client-Rollen:** Anwendungsspezifische Berechtigungen aus `resource_access`

Die Bedeutung und Berechtigungen der einzelnen Rollen (`admin`, `lender`, `user`) sind in Abschnitt 10.6.3 dokumentiert.

10.6.5.9 Ergebnis

Nach der Implementierung werden Benutzer vollautomatisch beim ersten Login angelegt. Der Prozess ist für den Endbenutzer transparent und erfordert keine zusätzlichen Registrierungsschritte. Die Datenbank-ID des Benutzers steht nach erfolgreicher Synchronisation für alle weiteren API-Aufrufe zur Verfügung.

10.7 Bildverwaltung

Dieses Kapitel dokumentiert die Implementierung des Bild-Upload-Systems für Produktbilder im LeihSy-System. Das System ermöglicht Administratoren und Verleiher, Bilder für Produkte hochzuladen, die anschließend im Katalog angezeigt werden.

10.7.1 Anforderungen

Gemäß User Story US-009 soll das System Bilder speichern und ausliefern können. Die Anforderungen umfassen:

- Bildupload mit Validierung (max. 5MB, Formate: JPG, PNG, WebP)
- Speicherung im Filesystem des Servers
- REST-Endpoints für Upload, Download und Löschen
- Integration in das Admin-Dashboard zur Produkterstellung

- Anzeige der Bilder im Produktkatalog

10.7.2 Architektur-Entscheidung: Filesystem vs. Datenbank

Für die Speicherung von Bildern wurden zwei Optionen evaluiert:

Option A – Datenbank (BLOB): Bilder werden als Binary Large Object direkt in der Datenbank gespeichert. Vorteile sind die einfache Datensicherung und transaktionale Konsistenz. Nachteile sind jedoch die erhöhte Datenbankgröße, langsamere Queries und höherer Speicherverbrauch.

Option B – Filesystem (gewählt): Bilder werden im Dateisystem gespeichert, die Datenbank enthält nur den URL-Pfad als String. Diese Lösung bietet bessere Performance beim Ausliefern von Bildern, einfachere Skalierbarkeit und die Möglichkeit, einen CDN vorzuschalten. Die Datenbank bleibt schlank und performant.

Die Entscheidung fiel auf das Filesystem, da für ein Verleihsystem mit Produktbildern die Performance beim Laden der Katalogseite wichtiger ist als transaktionale Konsistenz der Bilddaten.

10.7.3 Backend-Implementierung

10.7.3.1 Konfiguration

Die Upload-Konfiguration erfolgt in der `application.properties`:

```
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=5MB
spring.servlet.multipart.max-request-size=5MB
app.upload.dir=uploads/images/
```

Die Klasse `FileStorageConfig` erstellt beim Anwendungsstart automatisch das Upload-Verzeichnis, falls es nicht existiert. Dies verhindert Laufzeitfehler beim ersten Upload.

10.7.3.2 ImageService

Der `ImageService` kapselt die gesamte Logik für Dateispeicherung und -verwaltung:

- `saveImage(MultipartFile, String productName)`: Validiert die Datei und speichert sie mit einem aus dem Produktnamen generierten Dateinamen
- `loadImage(String filename)`: Lädt ein Bild aus dem Filesystem
- `deleteImage(String filename)`: Löscht ein Bild
- `validateImage(MultipartFile)`: Prüft Dateigröße und Dateityp

Die Validierung stellt sicher, dass nur erlaubte Dateitypen (JPG, PNG, WebP) mit maximal 5MB akzeptiert werden. Bei Verstößen wird eine `FileNotFoundException` geworfen.

10.7.3.3 Dateinamenskonvention

Ursprünglich wurden UUIDs als Dateinamen verwendet (z.B. 550e8400-e29b-41d4-a716-44665544). Dies wurde geändert zu produktnamenbasierten Dateinamen für bessere Nachvollziehbarkeit:

```
"Valve Index"      → valve-index.jpg  
"Meta Quest Pro"  → meta-quest-pro.jpg  
"Canon EOS R6 II" → canon-eos-r6-ii.jpg
```

Der Produktname wird dabei in Kleinbuchstaben konvertiert und Sonderzeichen werden durch Bindestriche ersetzt. Da jedes Produkt genau ein Bild hat, sind keine UUIDs für Eindeutigkeit erforderlich.

10.7.3.4 ImageController

Der REST-Controller stellt drei Endpoints bereit:

Methode	Endpoint	Beschreibung
POST	/api/images/upload	Lädt ein Bild hoch und gibt die URL zurück
GET	/api/images/{filename}	Liefert das Bild als Binärdaten aus
DELETE	/api/images/{filename}	Löscht das angegebene Bild

Tabelle 8: REST-Endpoints der Bildverwaltung

Der GET-Endpoint setzt den korrekten `Content-Type` Header basierend auf der Dateiendung, sodass Browser das Bild direkt anzeigen können.

10.7.3.5 Integration mit ProductService

Beim Erstellen oder Aktualisieren eines Produkts wird das Bild automatisch verarbeitet:

1. Frontend sendet `FormData` mit Produktdaten und Bilddatei
2. `ProductController` empfängt Multipart-Request
3. `ProductService` ruft `ImageService.saveImage()` auf

4. Rückgabewert (Dateiname) wird als `imageUrl` im Product gespeichert
5. Datenbank enthält nur den Pfad: `/api/images/valve-index.jpg`

10.7.4 Frontend-Integration

10.7.4.1 ProductService-Erweiterung

Der Angular `ProductService` wurde um Multipart-Upload-Support erweitert:

```
createProduct(product: ProductDTO, image: File | null): Observable<Product> {
  const formData = new FormData();
  formData.append('product', new Blob([JSON.stringify(product)], {
    type: 'application/json'
}));
  if (image) {
    formData.append('image', image);
  }
  return this.http.post<Product>(`.${this.apiUrl}/products`, formData);
}
```

Das Produkt wird als JSON-Blob gesendet, das Bild als separater File-Teil im `FormData`-Objekt.

10.7.4.2 Admin-Dashboard

Im Admin-Product-Dashboard wurde ein Bild-Upload-Bereich hinzugefügt:

- File-Input mit `accept="image/*"` für Dateiauswahl
- Bildvorschau vor dem Speichern mittels `FileReader`
- Validierung von Dateityp und -größe im Frontend
- Signal-basierte Zustandsverwaltung: `selectedFile`, `imagePreview`

Bei der Bearbeitung eines bestehenden Produkts wird das aktuelle Bild als Vorschau angezeigt. Ein neues Bild ersetzt das alte automatisch.

10.7.4.3 Katalog-Anzeige

Die Katalogseite lädt Produktbilder über eine `getImageUrl()`-Methode: Die Methode behandelt verschiedene Fälle: fehlende Bilder, absolute URLs (externe Bilder) und relative API-Pfade (interne Bilder).

10.7.5 Deployment-Überlegungen

Für den Produktivbetrieb auf dem Server müssen Docker Volumes konfiguriert werden, damit Bilder Container-Neustarts überleben:

```
# docker-compose.yml
services:
  backend:
    volumes:
      - ./uploads:/app/uploads
```

Ohne Volume-Mapping würden alle hochgeladenen Bilder beim Neustart des Containers verloren gehen, da Docker Container standardmäßig zustandslos sind.

10.7.6 Zusammenfassung

Das implementierte Bild-Upload-System ermöglicht eine vollständige Verwaltung von Produktbildern. Die Architekturentscheidung für Filesystem-Speicherung bietet optimale Performance für den Anwendungsfall eines Produktkatalogs. Die klare Trennung zwischen ImageService (Dateiverwaltung) und ProductService (Geschäftslogik) ermöglicht einfache Wartung und spätere Erweiterungen wie Thumbnail-Generierung oder CDN-Integration.

10.8 Automatische Buchungsstornierung

Das System storniert Buchungsanfragen automatisch, wenn diese nicht innerhalb eines definierten Zeitraums bearbeitet werden. Dies verhindert, dass Gegenstände durch unbearbeitete Anfragen dauerhaft blockiert werden.

10.8.1 Anforderungen

Gemäß User Story US-015 sollen unbestätigte Anfragen nach 24 Stunden automatisch storniert werden. Zusätzlich werden bestätigte Buchungen, die nicht abgeholt wurden, nach 24 Stunden als abgelaufen markiert.

10.8.2 Implementierung

Wir haben einen Spring Scheduler implementiert, der periodisch die Datenbank prüft und veraltete Buchungen per Soft-Delete storniert.

10.8.2.1 Konfiguration

Die Aktivierung erfolgt über `@EnableScheduling` in der Hauptklasse sowie drei Properties in der `application.properties`:

- `leihsy.scheduler.enabled` – Aktiviert/deaktiviert den Scheduler
- `leihsy.booking.auto-cancel-hours` – Frist für unbestätigte Anfragen (Standard: 24h)
- `leihsy.booking.auto-expire-hours` – Frist für nicht abgeholt Buchungen (Standard: 24h)

10.8.2.2 Scheduler-Komponente

Die Klasse `BookingScheduler` im Package `scheduler` enthält zwei zeitgesteuerte Methoden:

Methode	Ausführung	Funktion
<code>autoCancelPendingBookings</code>	Jede volle Stunde	Storniert PENDING-Buchungen ohne Termin
<code>autoExpireConfirmedBookings</code>	Jede halbe Stunde	Markiert nicht abgeholt Buchungen als EXPIRED

Tabelle 9: Scheduler-Methoden

Die Stornierung erfolgt durch Setzen des `deletedAt`-Feldes (Soft-Delete). Die bestehenden Repository-Queries `findPendingOlderThan` und `findConfirmedNotPickedUpOlderThan` werden wiederverwendet.

10.8.3 Zusammenfassung

Der Scheduler läuft automatisch im Hintergrund und erfordert keine Benutzerinteraktion. Er kann über die Konfiguration deaktiviert werden, beispielsweise für die lokale Entwicklung.

10.9 Studentengruppen für gemeinsame Ausleihen

Das Studentengruppen-Feature ermöglicht es Studierenden, sich zu Gruppen zusammenzuschließen und gemeinsam Gegenstände auszuleihen. Verleiher sehen bei Gruppenanfragen alle Mitglieder und können die Anfrage entsprechend bearbeiten.

10.9.1 Anforderungen

Das Feature basiert auf User Story US-044 und erfüllt folgende Anforderungen:

- Studierende können Gruppen erstellen und verwalten
- Gruppenmitglieder können hinzugefügt und entfernt werden
- Buchungen können optional einer Gruppe zugeordnet werden
- Verleiher sehen bei Gruppenbuchungen alle Mitgliedernamen
- Gruppen können nur gelöscht werden, wenn keine aktiven Buchungen existieren

10.9.2 Architektur-Entscheidung: Gruppen vs. Sammel-Buchungen

Bei der Konzeption standen zwei Ansätze zur Diskussion:

Option A – Sammel-Buchungen: Eine Buchung enthält mehrere Items und mehrere Nutzer. Vorteil: Alles in einem Request. Nachteil: Komplexe Validierung, alle Items müssen gemeinsam bestätigt/abgelehnt werden.

Option B – Separate Gruppen-Entity (gewählt): Gruppen existieren unabhängig von Buchungen. Jede Buchung referenziert optional eine Gruppe. Vorteil: Gruppen sind wiederverwendbar, einzelne Buchungen können unabhängig bearbeitet werden. Nachteil: Zusätzliche Entity und Tabellen.

Wir haben uns für Option B entschieden, da diese Architektur flexibler ist und das spätere Hinzufügen von Gruppen-Budgets ermöglicht.

10.9.3 Datenmodell

Das Feature führt zwei neue Datenbanktabellen ein:

Tabelle	Spalten	Beschreibung
student_groups	id, name, description, created_by_id, timestamps	Haupttabelle für Gruppen
student_group_members	group_id, user_id	Join-Tabelle (M:N)

Tabelle 10: Neue Datenbanktabellen für Studentengruppen

Die bookings-Tabelle wurde um die optionale Spalte `student_group_id` erweitert, die als Fremdschlüssel auf `student_groups` verweist.

10.9.4 Backend-Implementierung

Die Implementierung folgt dem etablierten Schichtenmuster mit Entity, Repository, Service und Controller.

10.9.4.1 Entity und DTOs

Die `StudentGroup`-Entity erbt von `BaseEntity` und enthält neben den Stammdaten eine Many-to-Many-Beziehung zu `User` für die Mitgliederverwaltung. Der Ersteller wird automatisch als erstes Mitglied hinzugefügt.

Für die API-Kommunikation wurden drei DTOs erstellt:

- `StudentGroupDTO` – Response mit allen Gruppeninformationen inkl. Mitgliederliste
- `CreateStudentGroupDTO` – Request zum Erstellen (Name, Beschreibung)
- `UpdateStudentGroupDTO` – Request zum Aktualisieren

10.9.4.2 Service-Logik

Der `StudentGroupService` implementiert folgende Geschäftslogik:

- Nur der Ersteller kann die Gruppe bearbeiten oder löschen
- Mitglieder können sich selbst aus einer Gruppe entfernen
- Gruppen mit aktiven Buchungen (PENDING, CONFIRMED, PICKED_UP) können nicht gelöscht werden
- Bei Gruppenerstellung wird der Ersteller automatisch als Mitglied hinzugefügt

10.9.4.3 REST-API Endpoints

Der `StudentGroupController` stellt folgende Endpoints bereit:

Methode	Endpoint	Beschreibung
GET	/api/groups	Alle Gruppen (Admin)
GET	/api/groups/me	Eigene Gruppen des Users
GET	/api/groups/{id}	Gruppe per ID
GET	/api/groups/search?q=	Suche nach Gruppenname
POST	/api/groups	Neue Gruppe erstellen
PATCH	/api/groups/{id}	Gruppe aktualisieren
DELETE	/api/groups/{id}	Gruppe löschen
POST	/api/groups/{id}/members/{userId}	Mitglied hinzufügen
DELETE	/api/groups/{id}/members/{userId}	Mitglied entfernen

Tabelle 11: REST-Endpoints für Studentengruppen

10.9.4.4 Integration in Booking-System

Das bestehende Booking-System wurde um die Gruppenunterstützung erweitert:

- `Booking`-Entity: Neues optionales Feld `studentGroup`
- `BookingDTO`: Neue Felder `groupId`, `groupName`, `groupMemberNames`
- `BookingMapper`: Mapping der Gruppeninformationen inkl. Mitgliedernamen
- `BookingService`: Validierung der Gruppenmitgliedschaft bei Buchungserstellung
- `BookingController`: Optionales `groupId`-Feld im `CreateBookingRequest`
- Neuer Endpoint GET `/api/bookings/groups/{groupId}` für alle Buchungen einer Gruppe

Bei der Buchungserstellung mit Gruppen-ID wird geprüft, ob der anfragende User Mitglied der angegebenen Gruppe ist. Ist dies nicht der Fall, wird die Anfrage mit HTTP 400 abgelehnt.

10.9.5 Zusammenfassung

Das Studentengruppen-Feature ermöglicht kollaborative Ausleihen für Projektgruppen. Die Implementierung als separate Entity bietet Flexibilität für zukünftige Erweiterungen wie Gruppen-Budgets. Die Integration in das bestehende Booking-System erfolgte durch optionale Felder, wodurch Einzelbuchungen weiterhin ohne Gruppenangabe möglich sind.

10.10 InSy-Integration

Die InSy-Integration ermöglicht den automatisierten Import von Gegenständen aus dem bestehenden Inventarisierungssystem (InSy) der Hochschule Esslingen in das LeihSy-Verleihsystem. Administratoren können eingehende Import-Anfragen prüfen, genehmigen oder ablehnen und dabei entscheiden, ob ein neues Produkt erstellt oder ein bestehendes erweitert werden soll.

10.10.1 Anforderungen

Gemäß User Story US-037 soll das System Daten aus InSy importieren können. Die konkreten Anforderungen umfassen:

- REST-API Endpoint für eingehende Import-Anfragen aus InSy
- Authentifizierung über API-Key (oder später OAuth2)
- Import von Inventarnummer, Name, Beschreibung, Kategorie, Lagerort und Besitzer
- Validierung der eingehenden Daten
- Bei bestehender Inventarnummer: Update statt Duplikat
- Admin-UI zur Prüfung und Genehmigung von Import-Anfragen
- Protokollierung aller Import-Vorgänge

Zusätzlich wurde im Kundengespräch geklärt, dass der Admin beim Import entscheiden können soll:

- **Neues Product erstellen:** Wenn das Gerät noch nicht im System existiert
- **Zu bestehendem Product hinzufügen:** Wenn es sich um ein weiteres Exemplar eines vorhandenen Gerätetyps handelt (z.B. eine weitere Meta Quest 3)

10.10.2 Architektur-Entscheidung: Staging-Tabelle vs. Direktimport

Für die Verarbeitung eingehender InSy-Daten haben wir zwei Ansätze evaluiert:

10.10.2.1 Option A – Direktimport Eingehende Daten werden sofort als Product und Item in die Haupttabellen geschrieben. Der Vorteil wäre die Einfachheit, jedoch fehlt dabei die Möglichkeit zur manuellen Prüfung und es könnten fehlerhafte oder unvollständige Daten ins System gelangen.

10.10.2.2 Option B – Staging-Tabelle (gewählt) Eingehende Daten werden zunächst in einer separaten `insy_import_items`-Tabelle gespeichert. Administratoren können diese Einträge prüfen und dann gezielt importieren oder ablehnen. Dieser Ansatz bietet:

- Qualitätskontrolle durch manuelle Prüfung
- Flexibilität bei der Zuordnung zu Products
- Nachvollziehbarkeit durch Status-Tracking
- Schutz vor fehlerhaften oder doppelten Einträgen

Wir haben uns für Option B entschieden, da die manuelle Kontrolle über importierte Gegenstände ein wichtiges Qualitätsmerkmal darstellt und der Kunde explizit eine Admin-Oberfläche zur Prüfung gewünscht hat.

10.10.3 Backend-Implementierung

10.10.3.1 Entity: InsyImportItem

Die Staging-Entity erweitert `BaseEntity` und speichert alle relevanten Felder aus InSy:

Feld	Typ	Beschreibung
insyId	Long	Eindeutige ID aus InSy
invNumber	String	Inventarnummer
name	String	Gerätename
description	String	Beschreibung
owner	String	Besitzer
location	String	Standort/Raumnummer
categoryName	String	Kategorienname
status	InsyImportStatus	PENDING, IMPORTED, REJECTED
rejectionReason	String	Grund bei Ablehnung

Tabelle 12: Felder der InsyImportItem-Entity

10.10.3.2 Enum: InsyImportStatus

Der Status eines Import-Eintrags durchläuft folgende Zustände:

- **PENDING:** Neu eingegangen, wartet auf Admin-Entscheidung
- **IMPORTED:** Erfolgreich als Product/Item importiert
- **REJECTED:** Vom Admin abgelehnt (mit Begründung)

Methode	Endpoint	Beschreibung
POST	/api/insy/push	Empfängt Daten von InSy (oder Mock)
GET	/api/insy/imports	Alle Import-Einträge (mit Filtern)
GET	/api/insy/imports/{id}	Einzelnen Eintrag abrufen
GET	/api/insy/imports/count	Statistiken (pending, imported, etc.)
PATCH	/api/insy/imports/{id}	Status ändern (IMPORT/REJECT)
PATCH	/api/insy/imports/batch	Mehrere Einträge gleichzeitig
POST	/api/insy/mock/imports	Mock-Daten generieren (Development)

Tabelle 13: InSy-Import REST-Endpoints

10.10.3.3 REST-Endpoints

10.10.3.4 Service-Logik: InsyImportService

Der Service implementiert die zentrale Import-Logik mit folgenden Kernfunktionen:

Duplikat-Erkennung: Bei eingehenden Daten wird geprüft, ob bereits ein Eintrag mit gleicher insyId existiert. Falls ja, werden die Daten aktualisiert statt ein Duplikat zu erstellen.

Product-Matching: Beim Abrufen von Import-Einträgen wird automatisch geprüft, ob ein Product mit ähnlichem Namen existiert. Diese Information wird im DTO angereichert:

```
private InsyImportItemDTO enrichWithMatchingProduct(
    InsyImportItemDTO dto, InsyImportItem item) {
    List<Product> matches = productRepository.searchByName(item.getName());
    if (!matches.isEmpty()) {
        dto.setHasMatchingProduct(true);
        dto.setMatchingProductId(matches.get(0).getId());
        dto.setMatchingProductName(matches.get(0).getName());
    }
    return dto;
}
```

Import-Aktionen: Der PATCH-Endpoint akzeptiert ein DTO mit einer action und optionalen Parametern:

Action	Parameter	Ergebnis
IMPORT_AS_NEW	categoryId, locationId, lenderId	Neues Product + Item erstellen
IMPORT_TO_EXISTING	productId, lenderId	Item zu bestehendem Product
REJECT	reason	Eintrag als abgelehnt markieren

Tabelle 14: Import-Aktionen und ihre Parameter

10.10.4 Entwicklungsgeschichte und Iterationen

Die Implementierung erfolgte in mehreren Phasen, wobei wir auf Basis von Tests und Feedback iterativ verbessert haben.

10.10.4.1 Phase 1: Initiale Backend-Implementierung

Wir haben zunächst die Entity, Repository und grundlegenden CRUD-Endpoints erstellt. Der erste Entwurf verwendete APPROVE und DELETE als Aktionen, analog zu einem einfachen Genehmigungs-Workflow.

10.10.4.2 Phase 2: Erweiterung um Import-Logik

Nach etwas ausprobieren wurde klar, dass der Admin mehr Kontrolle benötigt. Wir haben die Aktionen auf IMPORT_AS_NEW, IMPORT_TO_EXISTING und REJECT umgestellt:

Vorher	Nachher	Begründung
APPROVE	IMPORT_AS_NEW	Klarere Semantik: Was passiert genau?
-	IMPORT_TO_EXISTING	Neue Anforderung: Zu Product hinzufügen
DELETE	REJECT	Soft-Delete mit Begründung statt Löschen

Tabelle 15: Evolution der Import-Aktionen

10.10.4.3 Phase 3: Frontend-Backend-Synchronisation

Bei der Frontend-Implementierung stellten wir fest, dass die ursprünglichen Endpoint-Pfade nicht mit dem Backend übereinstimmten. Das Frontend erwartete POST-Endpoints wie /approve und /reject, während das Backend PATCH mit Action-Parameter verwendete.

Wir haben uns entschieden, das Frontend an das Backend anzupassen, da das PATCH-Pattern mit Action-Parameter flexibler und REST-konformer ist:

```
// Vorher (Frontend)
POST /api/insy/imports/{id}/approve
POST /api/insy/imports/{id}/reject

// Nachher (angepasst an Backend)
PATCH /api/insy/imports/{id} { action: "IMPORT_AS_NEW", ... }
PATCH /api/insy/imports/{id} { action: "REJECT", reason: "..." }
```

10.10.5 Frontend-Integration

10.10.5.1 Komponenten-Struktur

Die Admin-Oberfläche für den InSy-Import besteht aus folgenden Komponenten:

- `AdminInsyImportComponent`: Hauptkomponente mit Tabelle und Dialogen
- `AdminInsyImportService`: Page-Service für State Management
- `InsyImportService`: HTTP-Service für API-Kommunikation

10.10.5.2 Import-Dialog

Der Import-Dialog ermöglicht dem Admin die Entscheidung über den Import-Typ. Die Benutzeroberfläche zeigt:

1. Radio-Buttons für die Auswahl: “Als neues Produkt” oder “Zu bestehendem Produkt”
2. Falls ein passendes Product gefunden wurde, wird dies hervorgehoben
3. Dropdown-Menüs für Kategorie, Standort und Verleiher (bei neuem Product)
4. Dropdown für Product-Auswahl (bei bestehendem Product)

10.10.5.3 UX-Iterationen

Während der Frontend-Entwicklung haben wir mehrere UX-Probleme identifiziert und behoben:

Problem 1 – Scroll-Jumping: Beim Öffnen des Dialogs sprang die Seite zum Anfang. Ursache war das automatische Fokussieren des Dialogs. Lösung: `[focusOnShow]="false"` am Dialog.

Problem 2 – Dropdown-Overflow: Die PrimeNG-Dropdowns wurden am Dialog-Rand abgeschnitten. Lösung: `appendTo="body"` an allen Dropdowns, damit sie im Body-Kontext gerendert werden.

Problem 3 – Nicht scrollbarer Dialog: Bei vielen Formularfeldern war der Dialog-Content nicht scrollbar. Lösung: `[contentStyle] = "{'max-height': '70vh', 'overflow-y': 'auto'}"`.

10.10.5.4 Service-Methoden

Der HTTP-Service stellt folgende Methoden bereit:

10.10.6 Mock-Daten für Entwicklung

Für die Entwicklung ohne echte InSy-Anbindung haben wir einen Mock-Endpoint implementiert:

Methode	Beschreibung
getImportRequests(filter)	Import-Einträge mit optionalen Filtern laden
getStatistics()	Dashboard-Zahlen (pending, imported, rejected)
importAsNewProduct(id, data)	Als neues Product importieren
importToExistingProduct(id, data)	Zu bestehendem Product hinzufügen
rejectImport(id, reason)	Import ablehnen mit Begründung
bulkImportAsNew(ids, data)	Batch-Import mehrerer Einträge

Tabelle 16: Frontend-Service-Methoden

`POST /api/insy/mock/imports`

Dieser generiert realistische Testdaten mit verschiedenen Gerätetypen (VR-Brillen, Kameras, Audio-Equipment) und zufälligen Inventarnummern. Der Endpoint ist nur im Development-Profil verfügbar.

10.10.7 Zusammenfassung

Die InSy-Integration ermöglicht einen kontrollierten Import von Gegenständen aus dem bestehenden Inventarsystem. Durch die Staging-Tabelle und den Admin-Workflow ist sichergestellt, dass nur geprüfte Daten ins LeihSy-System gelangen. Das intelligente Product-Matching erleichtert die Zuordnung zu bestehenden Produkten und verhindert Duplikate.

Die Implementierung umfasst:

- 1 Entity mit 3 Status-Zuständen
- 7 REST-Endpoints inkl. Batch-Verarbeitung
- Vollständige Admin-UI mit Import-Dialog
- Mock-Endpoint für Entwicklung und Tests
- Automatisches Product-Matching

Die Integration in das bestehende InSy-System kann durch Anpassung des Push-Endpoints erfolgen, sobald die Schnittstelle auf InSy-Seite bereitsteht.

10.11 Sicheres Token-basiertes QR-Code-System

Wir haben das ursprüngliche, statische QR-Code-System durch ein sicheres, Token-basiertes Transaktionssystem ersetzt. Dies ermöglicht eine sichere Authentifizierung bei der Ausgabe und Rückgabe von Equipment, indem temporäre, einmalige Codes verwendet werden.

10.11.1 Motivation und Anforderungen

Das vorherige System kodierte lediglich die Booking-ID in einem permanent gültigen QR-Code. Dies stellte ein Sicherheitsrisiko dar, da der Code theoretisch kopiert oder erraten werden konnte. Zudem gab es keine zeitliche Begrenzung für die Gültigkeit.

Unsere Ziele für das neue System waren:

- **Sicherheit:** Der QR-Code darf keine direkten Rückschlüsse auf die Buchung zulassen und muss schwer zu erraten sein.
- **Zeitbegrenzung:** Ein Token soll nur 15 Minuten gültig sein.
- **Einmaligkeit:** Nach erfolgreichem Scan muss der Token ungültig werden (Single-Use).
- **Workflow-Steuerung:** Das System soll automatisch erkennen, ob es sich um eine Abholung (PICKUP) oder Rückgabe (RETURN) handelt.

10.11.2 Architektur-Entscheidung: Token-Transaktionen

Wir haben uns für eine Entkopplung von Buchungs-ID und QR-Code entschieden. Anstatt die ID zu scannen, generieren wir eine `BookingTransaction`-Entity.

Ablauf:

1. Der **Student** fordert einen Token an (Generierung).
2. Der **Verleiher** scannt den Token (Einlösung).
3. Das System führt den Statuswechsel der Buchung durch.

Diese Architektur erlaubt uns, Metadaten wie `expires_at` (Ablaufzeit) und `used_at` (Nutzungszeitpunkt) pro Transaktion zu speichern und historisch nachvollziehbar zu machen.

10.11.3 Backend-Implementierung

Die Kernlogik befindet sich im `BookingTransactionService`. Wir haben eine neue Entity `BookingTransaction` eingeführt, die einen 8-stelligen alphanumerischen Code speichert.

10.11.3.1 Automatische Typerkennung

Um die API für das Frontend zu vereinfachen, bestimmt das Backend den Transaktionstyp automatisch anhand des aktuellen Buchungsstatus:

- Status `CONFIRMED` → Transaktionstyp PICKUP
- Status `PICKED_UP` → Transaktionstyp RETURN

10.11.3.2 Idempotenz und Cleanup

Um Datenbank-Ressourcen zu schonen und die User Experience zu verbessern, haben wir eine Idempotenz-Prüfung implementiert: Fragt ein User einen Token an, während noch ein gültiger (nicht abgelaufener) Token für diese Buchung existiert, geben wir den bestehenden Token zurück, anstatt einen neuen zu generieren. Alte, ungenutzte Tokens werden vor einer Neu-Generierung invalidiert.

Methoden	Endpoint	Funktion
POST	/api/bookings/{id}/transactions	Token generieren (Student)
PATCH	/api/transactions/{token}	Token einlösen (Verleiher)

Tabelle 17: API-Endpoints für das Transaktionssystem

10.11.4 Frontend-Integration

Die Integration im Angular-Frontend erfolgt rollenspezifisch getrennt.

10.11.4.1 Studenten-Sicht (User Dashboard)

In der BookingQrComponent wird der QR-Code angezeigt. Wir haben hier einen Live-Timer implementiert, der dem Studenten anzeigt, wie lange der Code noch gültig ist (Countdown von 15 Minuten). Ist der Code abgelaufen, wird der Student visuell benachrichtigt und kann einen neuen Code anfordern.

10.11.4.2 Verleiher-Sicht (Lender Dashboard)

Im Dashboard des Verleiher haben wir die QrScannerComponent integriert. Diese bietet zwei Eingabemethoden:

- **Kamera-Scan:** Nutzung der Browser-API (`BarcodeDetector`) zum Scannen des QR-Codes.
- **Manuelle Eingabe:** Ein Textfeld zur Eingabe des 8-stelligen Tokens, falls keine Kamera verfügbar ist oder der Scan fehlschlägt.

Das Dashboard gibt dem Verleiher unmittelbares Feedback über Toast-Nachrichten (z.B. „Artikel VR-001 erfolgreich ausgegeben“).

10.11.5 Zusammenfassung

Durch die Implementierung des Token-Systems haben wir die Sicherheit bei der Geräteausgabe signifikant erhöht. Der Prozess ist nun robuster gegen Missbrauch und bietet durch die Timer-Anzeige und die flexible Eingabe (Scan oder manuell) eine verbesserte User Experience für beide Parteien.

11 Ausblick

Für zukünftige Versionen der Anwendung sind mehrere funktionale sowie technische Erweiterungen vorgesehen, die aus zeitlichen Gründen nicht umgesetzt werden konnten. Im Folgenden werden zentrale Aspekte betrachtet und mögliche Weiterentwicklungen beschrieben, die perspektivisch sinnvoll erscheinen.

11.1 Datenmanagement

Ein wesentlicher Schwerpunkt zukünftiger Arbeiten liegt in der Verbesserung des Datenmanagements innerhalb der Datenbank. Insbesondere sollen regelmäßige Aufräumroutinen für als *soft deleted* markierte Datensätze implementiert werden, um die Datenkonsistenz zu erhöhen und die langfristige Performance des Systems sicherzustellen. Der Fokus liegt dabei auf Daten, die entweder ausschließlich temporär gespeichert wurden oder keinen Einfluss auf die Nachvollziehbarkeit und Integrität anderer Datensätze besitzen.

Ein Beispiel hierfür sind Tokens, die zur Generierung zeitlich begrenzter QR-Codes verwendet werden. Nach dem erfolgreichen Scannen erfüllen diese keinen weiteren Zweck, da sämtliche relevanten Informationen dauerhaft in den Buchungsdatensätzen gespeichert sind.

Ebenso sammeln sich bei Buchungen, die von Nutzern erstellt und anschließend storniert oder abgelehnt wurden, Datensätze an, die keinen zusätzlichen Mehrwert für die Dokumentation des Ausleihverlaufs bieten. Diese können nach einer definierten Frist ohne Informationsverlust entfernt werden.

11.1.1 Performance und Wartbarkeit

Weitere zentrale Verbesserungspotenziale bestehen im Bereich der Performance sowie der Wartbarkeit (*Maintainability*) der Anwendung. Im aktuellen Entwicklungsstand werden beim erneuten Laden einer Seite sämtliche benötigten Daten direkt aus der Datenbank abgerufen, was insbesondere bei wachsendem Datenbestand zu längeren Ladezeiten führen kann.

Zur Optimierung wäre der Einsatz geeigneter Caching-Strategien sinnvoll. Mögliche Ansätze sind beispielsweise ein Browser-Cache, der häufig benötigte Daten lokal beim Nutzer speichert und diese nach Ablauf der Gültigkeit erneut aus der Datenbank abruft. Alternativ oder ergänzend könnten Proxy- oder Reverse-Proxy-Lösungen eingesetzt werden, um wiederkehrende Anfragen frühzeitig abzufangen und häufig genutzte Daten effizient bereitzustellen. Diese Maßnahmen würden die Systemperformance nachhaltig verbessern und die Benutzererfahrung positiv beeinflussen.

Zur Steigerung der Wartbarkeit empfiehlt sich langfristig die Einführung einer Microservices-

Architektur. Dabei würden funktional zusammengehörige Komponenten in eigenständige Services ausgelagert. Dies führt zu einer losen Kopplung der einzelnen Module, einer besseren Skalierbarkeit sowie übersichtlicheren und leichter wartbaren Codebasen. Eine sinnvolle funktionale Aufteilung könnte beispielsweise wie folgt aussehen:

- **Gegenstandsverwaltung** (Anlegen und Verwaltung von Gegenständen sowie deren Attributen)
- **Ausleihprozess** (Warenkorb, Buchungsübersichten sowie Abhol- und Rückgabe- prozesse)
- **Projektgruppenverwaltung** (Erstellung und Verwaltung von Projektgruppen im Vorlesungskontext)

11.2 Funktionale Erweiterungen

11.2.1 Erweiterung der Projektgruppen

Darüber hinaus ist eine Erweiterung des bestehenden **Projektgruppen-Features** geplant. Die grundlegende Funktionalität zur Erstellung von Projektgruppen sowie zur Verwaltung eines Budgets wurde bereits implementiert. Es fehlen jedoch weiterführende Funktionen, wie etwa die Einrichtung eines dedizierten Gegenstandspools, aus dem Projektgruppen ihre Ausleihe planen können. Dabei soll das verfügbare Budget automatisch mit den jeweiligen Tagespreisen der Gegenstände verrechnet werden.

Für die Umsetzung dieser Funktionalität ist zudem die Einführung einer Dozentenrolle erforderlich, welche den Gegenstandspool erstellt der für das Projekt relevant ist.

11.2.2 Verleihergruppen

Ein weiteres geplantes Feature ist die Einführung von **Verleihergruppen**, innerhalb derer mehrere Verleiher dieselben Gegenstände anbieten können. Dadurch wird es möglich, dass ein Gegenstand von unterschiedlichen Personen bereitgestellt wird, was die Verfügbarkeit und Flexibilität des Systems deutlich erhöht. Auch die Komplexität erhöht zum Beispiel bei den Fragen werden die Nachrichten an alle Verleiher in der Gruppe zugestellt, wie geht man um mit parallelen Ausleihbestätigungen.

11.2.3 In-App Nachrichten

Bisher ist eine Kommunikation per E-Mail, in Zukunft soll parallel dazu auch eine Nachrichtensystem implementiert werden, welches den Usern erlaubt Nachrichten auch in der App zu lesen und zu verfassen.

11.2.4 Verlängerung der Ausleihen

Die Möglichkeit zur Verlängerung bestehender Ausleihen stellt ein weiteres zentrales Feature dar, das aufgrund der hohen Komplexität und begrenzter Entwicklungszeit nicht umgesetzt werden konnte. Ziel ist es, Nutzern zu ermöglichen, aktive Ausleihen unter Berücksichtigung bereits geplanter zukünftiger Buchungen für denselben Gegenstand zu verlängern, ohne diesen zunächst zurückgeben und erneut ausleihen zu müssen.

Hierfür wäre die Implementierung einer entsprechenden Logik erforderlich, die sämtliche Buchungen eines Gegenstands analysiert und prüft, ob eine Verlängerung möglich ist oder gegebenenfalls Umbuchungen notwendig werden. Insbesondere bei zeitlichen Überschneidungen mit anderen Buchungen müsste das System entsprechende Entscheidungen automatisiert treffen.

11.2.5 Herausforderungen bei Ausleihen anhand verschiedener Use-Cases

Darüber hinaus existieren weitere Anwendungsfälle, die für einen vollständig abgebildeten und robusten Ausleihprozess essenziell sind, im Rahmen dieser Arbeit jedoch nicht vertieft wurden. Diese müssen zukünftig berücksichtigt werden:

- **Verspätete Rückgaben** (Beeinträchtigung oder Verhinderung geplanter Ausleihen)
- **Rückgabe beschädigter Gegenstände** (Sperrung weiterer Ausleihen und Verpflichtung zum Ersatz durch den Entleiher)