

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
Kunde: Christian Haas
Abgabedatum: 21. 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	Links zu allen verwendeten Tools	10
2	Erklärung der Aufgabe	10
2.1	Ist-Zustand	10
2.2	Soll-Zustand	11
2.3	Zielgruppe	11
2.4	Stakeholder	11
3	Anforderungsanalyse	12
3.1	Must-Have Features	12
3.1.1	Authentifizierung & Benutzerverwaltung	13
3.1.2	Gegenstandsverwaltung	13
3.1.3	Ausleihprozess	13
3.1.4	Email-System	14
3.1.5	Administration & Übersicht	14
3.1.6	Protokollierung & Datenschutz	14
3.1.7	Technische Anforderungen	14
3.2	Nice-To-Have Features	15
3.2.1	Hohe Priorität (++)	15
3.2.2	Mittlere Priorität (+)	15
3.2.3	Niedrige Priorität (-)	16
3.3	Funktionale Anforderungen	16
3.4	Nicht-funktionale Anforderungen	17
3.4.1	Performance und Skalierbarkeit	17
3.4.2	Gebrauchstauglichkeit (Usability)	18
3.4.3	Sicherheit	18
3.4.4	Zuverlässigkeit und Verfügbarkeit	19
3.4.5	Wartbarkeit und Erweiterbarkeit	19
3.4.6	Datenschutz und Compliance	20
3.4.7	Kompatibilität	20
3.4.8	Dokumentation	20
3.5	Qualitätsziele	21
3.6	Constraints	21
3.7	User-Stories	21
3.7.1	Epic 1: Authentifizierung & Benutzerverwaltung	22
3.7.2	Epic 2: Gegenstandsverwaltung	22
3.7.3	Epic 3: Ausleihprozess – Warenkorb	24
3.7.4	Epic 4: Ausleihprozess – Terminverwaltung	26
3.7.5	Epic 5: Ausgabe/Rückgabe vor Ort	27
3.7.6	Epic 6: E-Mail-Benachrichtigungen	28
3.7.7	Epic 7: Protokollierung & Datenschutz	29
3.7.8	Epic 8: Administration & Reporting	30
3.7.9	Epic 9: Integration & Schnittstellen	30
3.7.10	Epic 10: Technische Anforderungen	31
3.7.11	Epic 11: Nice-to-Have Features	31
4	Zeitmanagement	34

5 Projektmanagement	35
5.1 Methode	35
5.2 Organisation	35
5.3 Eingesetzte Tools	36
6 Entwicklungsumgebung	36
6.1 Eingesetzte IDE	36
6.2 Eingesetzte Frameworks	36
6.2.1 Backend Framework	36
6.2.2 Weitere Backend-Bibliotheken	37
6.2.3 Frontend Framework	37
6.2.4 Weitere Frontend-Bibliotheken	37
6.2.5 Lizenzen der eingesetzten Frameworks und Bibliotheken	38
6.2.6 Integration in bereits bestehende Infrastruktur	38
6.3 Server	38
6.4 Datenbank	38
6.4.1 Entwicklungsumgebung mit H2	39
6.4.2 Spring Profiles	39
6.4.3 Datenbankverbindung	40
6.4.4 Automatische Testdaten	40
6.5 Development-Server	40
6.6 Version Control management System	42
6.7 API-Testing mit Postman	44
6.7.1 Automatisches Token-Management	44
6.7.2 Collection Variables	45
6.7.3 Test-Requests	45
6.7.4 Vorteile	46
7 User Research	46
7.1 Proto-Personas	46
7.1.1 Persona A – Lena Schmid (Studierende)	46
7.1.2 Persona B – Max Schmidt (IT-Admin / Systemverantwortlicher) .	47
7.1.3 Persona C – Prof. Tom Fischer (Lehrender/Verleiher)	48
7.2 Interview Auswertung	49
7.3 Auswertung (Online-Umfrage, n = 28)	50
7.3.1 Stichprobe & Nutzung	50
7.3.2 Zufriedenheit (Ist-Prozess)	50
7.3.3 Größte Pain Points (Top-Nennungen)	51
7.3.4 Adoptionsbereitschaft für ein digitales System	51
7.3.5 Top-Features (max. 3 Stimmen)	52
7.3.6 Unverzichtbare Filter	53
7.3.7 Verfügbarkeit je Campus	53
7.3.8 Storno-Fenster (Abhol-/Rückgabe-Slots)	54
8 UI-Prototyp	54
8.1 Farbpalette	54
8.1.1 Version 1 (Erstkonzept)	54
8.1.2 Version 2 (Wireframe)	55
8.1.3 Finale, farbige Version	55

8.2	Erste Prototypen	55
8.2.1	Version 1 – Grundidee	55
8.2.2	Version 2 – Änderungen gegenüber Version 1	60
8.3	Visuelle Umsetzung der Funktionen	65
8.3.1	Login-Seite	66
8.3.2	Geräte-Details	66
8.3.3	Warenkorb	67
9	Softwarearchitektur	69
9.1	Keycloak	69
9.2	Frontend	69
9.3	Backend	70
9.4	Entity-Relationship-Diagramme	71
9.4.1	Erster Entwurf	71
9.4.2	Änderungen vom ersten Entwurf	72
9.4.3	Aktualisierung des Entity-Relationship-Diagramm	74
9.5	Logische Sichten	76
9.5.1	Verteilungssicht	76
9.5.2	Struktursicht	78
9.5.3	Verhaltenssicht	80
9.6	API-Dokumentation	83
10	Features	93
10.1	Admin Dashboard	93
10.1.1	Admin-Menü	93
10.1.2	Admin-Menü: Alle Gegenstände (Übersicht)	94
10.1.3	Admin-Menü: Produktgruppen verwalten	96
10.1.4	Admin-Menü: Einzelne Gegenstände verwalten	98
10.1.5	Admin-Menü: Buchungen verwalten	102
10.1.6	Admin-Menü: Standorte Verwalten	105
10.1.7	Admin-Menü: Gruppen verwalten	106
10.1.8	Admin-Feature: Kategorienverwaltung	107
10.1.9	Admin-Feature: Verleiher-Zuordnung	110
10.2	Verleiher Dashboard	112
10.2.1	Verleiher-Menü	112
10.2.2	Verleiher-Menü: Meine Gegenstände	113
10.2.3	Verleiher-Menü: Ausleihe Übersicht	115
10.2.4	Verleiher-Menü: Anfragen	117
10.2.5	Verleiher-Menü: Private Gegenstände verleihen	119
10.2.6	Verleiher-Dashboard: Backend-Implementierung	120
10.2.7	Offene Anfragen für Verleiher: Backend-Implementierung	121
10.3	Nutzer Dashboard	123
10.3.1	Nutzer-Menü	123
10.3.2	Nutzer-Menü: Meine Buchungen	124
10.3.3	Nutzer-Menü: Mein Gruppen	126
10.4	E-Mails für den Ausleihprozess	128
10.4.1	Bestätigung der Abholung	128
10.4.2	Dokumentation der Rückgabe	128
10.4.3	Benachrichtigung bei Statusänderungen	128

10.4.4	Automatisierte Erinnerungen und Mahnwesen	129
10.5	Warenkorb	129
10.5.1	Erste Version des Warenkorbs	130
10.5.2	Verbesserungen des Warenkorbs	133
10.5.3	Backend-Änderungen für Warenkorb	137
10.6	Backend-Implementierung der Buchungsverwaltung	139
10.6.1	DTO-Pattern und Circular Reference	139
10.6.2	Status-Verwaltung über Timestamps	140
10.6.3	Architektur-Entscheidungen	140
10.6.4	Zusammenfassung	141
10.7	Authentifizierung und Benutzerverwaltung	142
10.7.1	Überblick & Architektur	142
10.7.2	JWT-Token und Security-Konfiguration	143
10.7.3	Rollensystem	144
10.7.4	Frontend-Integration	144
10.7.5	Automatische Benutzersynchronisation	146
10.8	Bildverwaltung	148
10.8.1	Anforderungen	148
10.8.2	Architektur-Entscheidung: Filesystem vs. Datenbank	148
10.8.3	Backend-Implementierung	148
10.8.4	Frontend-Integration	150
10.8.5	Deployment-Überlegungen	151
10.8.6	Zusammenfassung	152
10.9	Automatische Buchungsstornierung	152
10.9.1	Anforderungen	152
10.9.2	Implementierung	152
10.9.3	Zusammenfassung	153
10.10	Studentengruppen für gemeinsame Ausleihen	153
10.10.1	Anforderungen	153
10.10.2	Architektur-Entscheidung: Gruppen vs. Sammel-Buchungen	153
10.10.3	Datenmodell	154
10.10.4	Backend-Implementierung	154
10.10.5	Zusammenfassung	155
10.11	InSy-Integration	155
10.11.1	Anforderungen	156
10.11.2	Architektur-Entscheidung: Staging-Tabelle vs. Direktimport	156
10.11.3	Backend-Implementierung	157
10.11.4	Entwicklungsgeschichte und Iterationen	158
10.11.5	Mock-Daten für Entwicklung	161
10.11.6	Zusammenfassung	161
10.12	Sicheres Token-basiertes QR-Code-System	161
10.12.1	Motivation und Anforderungen	162
10.12.2	Architektur-Entscheidung: Token-Transaktionen	162
10.12.3	Backend-Implementierung	162
10.12.4	Frontend-Integration	163
10.12.5	Zusammenfassung	163
10.13	Qualitätssicherung und Backend-Hardening	164
10.13.1	Test-Architektur	164

10.13.2 Validierung der Geschäftslogik	164
10.13.3 Controller-Tests und Security-Integration	165
10.13.4 Sicherheitsoptimierungen (Hardening)	165
10.13.5 Fazit	166
10.14 Deployment	166
10.14.1 Frontend	167
10.14.2 Backend	167
11 Aufgabenverteilung	168
11.1 Frontend	168
11.2 Backend	170
11.3 Projektübergreifende Aufgaben	173
12 Reflektion	173
12.1 Umsetzung der User Stories	174
12.1.1 Umgesetzte Must-Have Features	174
12.1.2 Umgesetzte Should-Have Features	174
12.1.3 Umgesetzte Nice-to-Have Features	175
12.1.4 Zusätzlich umgesetzte Features	175
12.1.5 Nicht umgesetzte Nice-to-Have Features	176
12.1.6 Zusammenfassung	177
12.2 Reflektion zum Projektmanagement	177
12.3 Reflektion zur Zeitplanung	178
12.4 Reflektion zum Lernfortschritt	179
13 Ausblick	180
13.1 Datenmanagement	180
13.2 Performance und Wartbarkeit	180
13.3 Funktionale Erweiterungen	181
13.3.1 Erweiterung der Projektgruppen	181
13.3.2 Verleihergruppen	181
13.3.3 In-App Nachrichten	181
13.3.4 Verlängerung der Ausleihen	182
13.3.5 Herausforderungen bei Ausleihen anhand verschiedener Use-Cases .	182

Abbildungsverzeichnis

1	Netzwerkübersicht Development-Server	41
2	Alter Entwurf der CI/CD Pipeline Architektur in GitHub	42
3	Finale CI/CD Pipeline in GitHub	43
4	Persona Lena Schmid	47
5	Persona Max Schmidt	48
6	Persona Tom Fischer	49
7	Anteil Studenten mit kürzlichen Ausleihen	50
8	Zufriedenheit mit aktuellem Prozess	50
9	aktuelle Pain Points	51
10	Adoptionsbereitschaft für digitales System	52
11	Wichtigste Features	52
12	Unverzichtbare Filter	53
13	Wichtigkeit Verfügbarkeit nach Campus	53
14	Länge Stornierbarkeit	54
15	Login screen	56
16	Inventarkatalog	57
17	Geräte-Details Seite	58
18	Persönlicher Bereich	59
19	Aktualisierter Inventarkatalog	60
20	Kalender in Produktdetails	61
21	Abholungszeitfenster	62
22	Warenkorb	63
23	Standort des Geräts	64
24	Aktualisierter persönlicher Bereich	65
25	Aktualisierter Login screen	66
26	Geräte-Details Seite mit aktualisierten Design	67
27	Verbesserter Warenkorb	68
28	Erster Entwurf Entity-Relationship-Diagramm	72
29	Entity-Relationship-Diagramm	76
30	Diagramm zur Verteilungssicht	77
31	Komponentendiagramm	78
32	Sequenzdiagramm zum Ausleihprozess	81
33	Sequenzdiagramm zum Rückgabeprozess	82
34	Sequenzdiagramm zum Login	83
35	Übersicht über Gegenstände einer Produktgruppe	94
36	Übersicht über alle Produktgruppen	96
37	Übersicht über alle Buchungen	102
38	Statistiken zu allen Buchungen	104
39	Standortmenü für den Admin	105
40	Detailansicht der Standorte und der dort gelagerten Produkte	106
41	Kategorienverwaltung im Admin-Dashboard	108
42	Dialog zur Verleiher-Zuordnung in der Geräteverwaltung	110
43	Verleiher-Ansicht aller zugewiesenen Items	114
44	Detailansicht des Gegenstandes für den Verleiher	115
45	Ausleihe-Übersicht im Verleiher-Menü	116
46	Anfragen-Übersicht	118

47	Buchungsansicht aus Sicht des Users	125
48	Ansicht des Nutzers im Gruppenmenü	126
49	Produkt-Detailseite	130
50	Zum Warenkorb Hinzugefügt	131
51	Warenkorb-Seite	132
52	Warenkorb-Checkout	132
53	Add To Cart Neu	133
54	Add To Cart Eingabefehler	134
55	Verbesserter Warenkorb	135
56	Warenkorb Eingabefehler	136
57	Warenkorb Buchungen erfolgreich erstellt	137

Tabellenverzeichnis

1	Qualitätsziele der App gemäß arc42	21
2	Projekt-Constraints gemäß arc42	21
3	Lizenzen der eingesetzten Frameworks und Bibliotheken	38
4	Übersicht der Konfigurationsdateien	39
5	Status-Steuerung über Timestamps	140
6	Übersicht aller Änderungen am Booking-System	141
7	Zugriffskontrolle nach Endpoint-Mustern	143
8	REST-Endpoints der Bildverwaltung	150
9	Scheduler-Methoden	153
10	Neue Datenbanktabellen für Studentengruppen	154
11	REST-Endpoints für Studentengruppen	155
12	Felder der InsyImportItem-Entity	157
13	InSy-Import REST-Endpoints	158
14	Import-Aktionen und ihre Parameter	159
15	Evolution der Import-Aktionen	159
16	Frontend-Service-Methoden	161
17	API-Endpoints für das Transaktionssystem	163
18	Übersicht der implementierten Test-Ebenen	164
19	Aufgabenverteilung im Frontend	168
20	Aufgabenverteilung im Backend	170
21	Aufgabenverteilung für projektübergreifende Tätigkeiten	173

1 Links zu allen verwendeten Tools

Der folgende Link ruft die zentrale GitHub Seite der LeihSy Organisation auf.

- **GitHub Organisation:** <https://github.com/LeihSy>

In der Organisation, sind drei Repositories enthalten:

- **Backend:** Enthält das Backend Repository, mit der Lizenz der Applikation sowie dem Installations- und Administrationshandbuch
- **Frontend:** Enthält das Backend Repository, mit der Lizenz der Applikation sowie dem Installations- und Administrationshandbuch
- **Documentation:** Enthält die Meilensteinpräsentationen sowie die finalen Zeiteinträge aus Clockify

2 Erklärung der Aufgabe

In diesem Kapitel werden die Ausgangslage sowie die Motivation für das Projekt LeihSy dargestellt. Der bestehende manuelle Prozess der Geräteausleihe an der Hochschule wird analysiert, seine Defizite werden aufgezeigt, und daraus werden die konkreten Ziele für die Entwicklung der neuen Softwarelösung abgeleitet.

2.1 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 gestaltet sich der Ausleihprozess unübersichtlich, ineffizient und wenig nachhaltig. Ziel der neuen Lösung ist es, diese Defizite zu beheben und den Prozess zu optimieren.

2.2 Soll-Zustand

Ziel ist es, den bisherigen papierbasierten Ablauf durch eine gebrauchstaugliche und nachhaltige Website zu ersetzen und den Ausleihprozess von Anfang bis Ende digital nachvollziehbar zu gestalten.

Studierende, Dozierende und Mitarbeitende können sich über ihren Hochschul-Account anmelden, verfügbare Geräte einsehen sowie Reservierungen und Verlängerungen durchführen.

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 wird darauf geachtet, dass der Code strukturiert, effizient und funktionsfähig gestaltet ist.

2.3 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.

2.4 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 2.3 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 .

3 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.

3.1 Must-Have Features

In diesem Abschnitt werden die essenziellen funktionalen Anforderungen definiert, die für den produktiven Betrieb von LeihSy unerlässlich sind. Diese Funktionen bilden den Kern des Minimum Viable Products (MVP) und haben bei der Implementierung höchste Priorität, um die grundlegenden Abläufe der Ausleihe und Rückgabe sicherzustellen.

3.1.1 Authentifizierung & Benutzerverwaltung

Die Authentifizierung soll über Single Sign-On (SSO) mittels Keycloak erfolgen, um eine nahtlose Integration mit dem bestehenden Hochschul-RZ-Account zu ermöglichen. Studierende und Mitarbeiter sollen sich mit ihren gewohnten Zugangsdaten anmelden können, ohne separate Credentials für LeihSy verwalten zu müssen.

Das System benötigt eine rollenbasierte Zugriffskontrolle mit drei Benutzerrollen: Administrator, Verleiher und Student/Entleiher. Jede Rolle erhält spezifische Berechtigungen für verschiedene Funktionen des Systems. Die Rollenvergabe soll über Keycloak-Benutzergruppen erfolgen, sodass die Zuordnung zentral verwaltet werden kann.

3.1.2 Gegenstandsverwaltung

Für die Verwaltung von Verleihgegenständen sind vollständige CRUD-Operationen erforderlich, die Administratoren und Verleiher das Erstellen, Anzeigen, Bearbeiten und Löschen von Gegenständen ermöglichen. Jeder Gegenstand muss mit Pflichtattributen wie Inventarnummer, Name, Beschreibung, Kategorie, Foto, Zubehör, Lagerort und Status erfasst werden können.

Eine wichtige Anforderung ist die Verwaltung von Sets gleichartiger Gegenstände. Wenn beispielsweise zehn identische Meta Quest VR-Brillen vorhanden sind, sollen diese als Set mit eindeutigen Nummern angelegt werden können. Dies soll die Verwaltung großer Bestände erleichtern und dennoch die individuelle Nachverfolgung einzelner Geräte ermöglichen.

Das System muss eine Kategorisierung von Gegenständen sowie Filtermöglichkeiten nach Kategorie, Verfügbarkeit und Lagerort bieten. Eine Volltext-Suche über Name und Beschreibung ist ebenfalls erforderlich. Verleiher sollen ausschließlich die ihnen zugeordneten Gegenstände sehen können.

Bei der Auswahl eines Gegenstandes soll das System automatisch benötigte Zusatzgegenstände anzeigen. Wenn beispielsweise eine Kamera ausgewählt wird, soll auf die Notwendigkeit einer Speicherkarte hingewiesen und diese zur Mitausleihe vorgeschlagen werden.

3.1.3 Ausleihprozess

Der Ausleihprozess beginnt mit einer Warenkorb-Funktionalität, über die Studierende mehrere Gegenstände sammeln können, bevor sie eine gemeinsame Ausleiheanfrage stellen. Die Auswahl des gewünschten Ausleihzeitraums erfolgt über einen Kalender, der die Verfügbarkeit der Gegenstände in Echtzeit anzeigt. Auf der Detailseite jedes Gegenstandes soll ein Verfügbarkeitskalender bereits gebuchte Zeiträume visuell darstellen.

Nach dem Absenden einer Anfrage erhält der zuständige Verleiher eine Benachrichtigung und kann die Ausleihe bestätigen oder ablehnen. Nicht bestätigte Anfragen sollen nach

24 Stunden automatisch storniert werden, um das System von veralteten Anfragen freizuhalten und unnötige Blockierungen zu vermeiden.

Die digitale Ausgabe und Rückgabe erfolgt mit Email-Bestätigung. Vor Ort müssen sich Studierende authentifizieren, um den Erhalt beziehungsweise die Rückgabe des Gegenstandes zu bestätigen. Jeder Student soll zudem seinen persönlichen Ausleihverlauf mit aktuellen, offenen und vergangenen Ausleihen einsehen können.

3.1.4 Email-System

Das Email-System muss automatische Benachrichtigungen bei verschiedenen Ereignissen im Ausleihprozess versenden. Dies umfasst Bestätigungen bei Erstellung, Bestätigung oder Ablehnung einer Anfrage sowie bei Ausgabe und Rückgabe von Gegenständen. Zwei Tage vor Fälligkeit sollen Studierende automatisch eine Erinnerungsmail erhalten. Bei nicht fristgerechter Rückgabe müssen sowohl der Student als auch der Verleiher benachrichtigt werden.

3.1.5 Administration & Übersicht

Das Admin-Dashboard soll eine zentrale Übersicht über alle wichtigen Kennzahlen und Vorgänge im System bieten. Administratoren benötigen Einsicht in offene Anfragen, bestätigte aber noch nicht abgeholt Ausleihen sowie aktuelle und zukünftige Ausleihen. Zusätzlich sollen Statistiken zur Auslastung und Nutzung bereitgestellt werden. Über eine Systemkonfiguration müssen Administratoren zentrale Einstellungen verwalten können.

3.1.6 Protokollierung & Datenschutz

Für die Nachvollziehbarkeit aller Vorgänge ist ein umfassendes Audit-Log erforderlich. Dieses muss alle relevanten Aktionen wie Ausleihen, Rückgaben, Änderungen an Gegenständen und Login-Versuche protokollieren. Die Protokollierung erfolgt datenschutzkonform und ermöglicht es, im Problemfall nachzuvollziehen, wer wann welche Aktion durchgeführt hat.

Die DSGVO-Konformität wird durch konsequente Datensparsamkeit und Zweckbindung sichergestellt. Es werden nur die für den Ausleihprozess notwendigen Daten gespeichert und definierte Aufbewahrungsfristen festgelegt. Nutzer erhalten das Recht auf Datenexport und Löschung ihrer personenbezogenen Daten. Session-Daten müssen verschlüsselt übertragen und temporäre Daten wie Warenkorb-Inhalte automatisch nach einer definierten Zeitspanne gelöscht werden.

3.1.7 Technische Anforderungen

Alle Komponenten des Systems sollen mit Docker containerisiert werden. Frontend, Backend, Datenbank und Keycloak müssen über Docker-Compose mit einem einzigen Befehl

startbar sein. Als Datenbank ist PostgreSQL für die persistente Datenspeicherung vorgesehen. Die Benutzeroberfläche muss responsiv gestaltet und für Desktop sowie Mobile optimiert sein.

3.2 Nice-To-Have Features

Ergänzend zu den Must-Have-Anforderungen wurden eine Reihe von Funktionen definiert, die den Funktionsumfang von LeihSy erweitern und die User Experience verbessern sollen, jedoch für den initialen Betrieb nicht zwingend erforderlich sind. Diese Anforderungen wurden nach Priorität gestaffelt, um eine flexible Umsetzung je nach verfügbaren Zeitressourcen zu ermöglichen.

3.2.1 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

3.2.2 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)

3.2.3 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

3.3 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.

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.

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.

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 storniert werden.

Benachrichtigungssystem:

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

Übersichten und Verlauf:

Das System muss allen Benutzergruppen rollenbezogene Übersichten bereitstellen: Stu-

dierende sehen ihren persönlichen Ausleihverlauf, Verleiher sehen Anfragen und Ausleihen ihrer zugeordneten Gegenstände, Administratoren erhalten ein zentrales Dashboard mit Statistiken und Gesamtübersicht.

Integration:

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

Protokollierung:

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

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.

3.4 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.

3.4.1 Performance und Skalierbarkeit

Neben den funktionalen Aspekten spielen Qualitätsanforderungen eine entscheidende Rolle für die Akzeptanz und den langfristigen Betrieb von LeihSy. In den folgenden Abschnitten werden die technischen Rahmenbedingungen wie Leistungsfähigkeit, Sicherheit und Wartbarkeit spezifiziert, die das System erfüllen muss.

- **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.

3.4.2 Gebrauchstauglichkeit (Usability)

Da das System von einer breiten Nutzerbasis verwendet wird, wird besonderer Wert auf eine intuitive Benutzeroberfläche gelegt, die eine effiziente Bedienung ohne lange Einarbeitungszeit ermöglicht.

- **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.

3.4.3 Sicherheit

Da das System sensible Nutzerdaten verarbeitet und wertvolles Inventar verwaltet, müssen unberechtigte Zugriffe und Manipulationen konsequent unterbunden werden. Hierfür kommen moderne Authentifizierungsstandards (OAuth2/OIDC) sowie ein striktes, rollenbasiertes Zugriffskonzept zum Einsatz.

- **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.

3.4.4 Zuverlässigkeit und Verfügbarkeit

Zur Sicherstellung des reibungslosen Ausleihbetriebs an der Hochschule muss das System insbesondere zu Stoßzeiten (z.B. Semesterbeginn) stabil arbeiten. Eine hohe Verfügbarkeit während der Geschäftszeiten minimiert Warteschlangen und den Einsatz von Papier-Rückfallebenen.

- **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.

3.4.5 Wartbarkeit und Erweiterbarkeit

Da LeihSy als studentisches Projekt langfristig von wechselnden Teams weiterentwickelt werden soll, sind eine saubere Code-Architektur und umfassende Dokumentation essenziell. Die Software ist modular aufgebaut, sodass zukünftige Features ohne aufwendiges Refactoring integriert werden können.

- **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.

3.4.6 Datenschutz und Compliance

Der Umgang mit personenbezogenen Daten von Studierenden und Mitarbeitenden erfordert die strikte Einhaltung der DSGVO sowie der internen Hochschulrichtlinien. Mechanismen zur Datensparsamkeit werden implementiert, und historische Daten (z.B. alte Ausleihen) werden nur so lange gespeichert, wie es erforderlich ist.

- **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.

3.4.7 Kompatibilität

Die Webanwendung muss auf den gängigen Endgeräten und Browsern der Studierenden fehlerfrei funktionieren, um eine hürdenfreie Nutzung zu gewährleisten. Zudem muss die Schnittstellenkompatibilität zu bestehenden Hochschulsystemen (insbesondere dem Inventarsystem InSy) dauerhaft sichergestellt sein.

- **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.

3.4.8 Dokumentation

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

3.5 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

3.6 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

3.7 User-Stories

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

3.7.1 Epic 1: Authentifizierung & Benutzerverwaltung

Dieses Epic bildet das fundamentale Sicherheitsgerüst der Anwendung. Es umfasst die Integration des zentralen Identitätsmanagements der Hochschule sowie die Implementierung einer rollenbasierten Zugriffskontrolle, um sicherzustellen, dass Nutzer nur auf die für sie freigegebenen Funktionen zugreifen können.

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

3.7.2 Epic 2: Gegenstandsverwaltung

Dieses Epic bildet das Herzstück des Inventarsystems. Es bündelt alle Funktionen, die es den Verleihern ermöglichen, Produkte, Kategorien und Lagerorte effizient zu erfassen und zu pflegen, um so die Basis für den Ausleihbetrieb bereitzustellen.

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

3.7.2.1 Hinzugekommene User Stories

- Medienkatalog filtern & Zeitraum prüfen
 - Als Student 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 Student 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.
- Gerätedetails und Ausleihbarkeit prüfen
 - Als Student 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.

3.7.3 Epic 3: Ausleihprozess – Warenkorb

Dieses Epic beschreibt den Kern des studentischen Workflows: die Auswahl und Reservierung von Equipment. Es umfasst alle Funktionen, die notwendig sind, um Gegenstände in einem virtuellen Warenkorb zu sammeln, Verfügbarkeiten für einen gewählten Zeitraum

zu prüfen und die Buchungsanfrage gebündelt abzusenden.

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

3.7.3.1 Hinzugekommene User Stories

- Gerätedetails und Ausleihbarkeit prüfen
 - Als Student 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.

3.7.4 Epic 4: Ausleihprozess – Terminverwaltung

Dieses Epic fokussiert sich auf die zeitliche Koordination der Ausleihen und umfasst die Verwaltung eingehender Buchungsanfragen sowie die Abstimmung verbindlicher Abhol- und Rückgabetermine zwischen Verleihern und Studierenden.

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

3.7.5 Epic 5: Ausgabe/Rückgabe vor Ort

Dieses Epic behandelt den physischen Prozess der Geräteübergabe. Es umfasst alle Funktionen, die für die Protokollierung der Ausgabe sowie die Prüfung und Erfassung der Rückgabe vor Ort im Leihraum notwendig sind.

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

3.7.6 Epic 6: E-Mail-Benachrichtigungen

Dieses Epic automatisiert die Kommunikation zwischen System und Nutzern, um den manuellen Verwaltungsaufwand zu minimieren. Es umfasst den trigger-basierten Versand von E-Mails bei Statusänderungen, Fristerinnerungen für bald fällige Rückgaben sowie Mahnungen bei Überfälligkeiten.

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

3.7.7 Epic 7: Protokollierung & Datenschutz

Dieses Epic widmet sich der Rechtssicherheit und Transparenz des Systems. Es umfasst die Implementierung von Audit-Logs zur Nachvollziehbarkeit kritischer Aktionen (z. B. Statusänderungen) sowie technische Mechanismen zur datenschutzkonformen Löschung und Anonymisierung personenbezogener Daten gemäß DSGVO.

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

3.7.8 Epic 8: Administration & Reporting

Dieses Epic bündelt alle administrativen Funktionen, die für die Systempflege und das übergeordnete Management notwendig sind. Dazu gehören die Verwaltung von Nutzerrollen, globale Konfigurationen sowie die Generierung von Reports und Bestandslisten für Inventuren.

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 (Asinas)

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

3.7.9 Epic 9: Integration & Schnittstellen

Dieses Epic fokussiert sich auf die nahtlose Einbettung von LeihSy in die bestehende IT-Landschaft der Hochschule. Im Mittelpunkt steht die Schnittstelle zur Datensynchronisation mit dem führenden Inventarsystem (InSy).

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

3.7.10 Epic 10: Technische Anforderungen

Dieses Epic definiert das fundamentale technische Rückgrat der Anwendung. Es umfasst die Einrichtung der Entwicklungsumgebung, die Konfiguration der CI/CD-Pipelines für automatisierte Tests und Deployments sowie die Bereitstellung der Container-Infrastruktur (Docker) und Datenbanken.

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

3.7.11 Epic 11: Nice-to-Have Features

Dieses Epic sammelt funktionale Erweiterungen, die den Nutzungskomfort erhöhen, aber für den initialen Betrieb nicht kritisch sind (Scope Management). Diese Anforderungen dienen als Backlog für verbleibende Projektzeit oder zukünftige Weiterentwicklungen.

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 Verleiher 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 Gruppenausleihen

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

4 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.

4.0.0.1 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.

5 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.

5.1 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.

5.2 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.

5.3 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.

6 Entwicklungsumgebung

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

6.1 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.

6.2 Eingesetzte Frameworks

Um das Projekt umzusetzen wurden unterschiedliche Frameworks und Bibliotheken eingesetzt. Die Frameworks und Bibliotheken werden in den folgenden Abschnitten genauer ausgeführt und mit Erklärungen für die Auswahl der Technologien begründet.

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.2.4 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.

6.2.5 Lizenzen der eingesetzten Frameworks und Bibliotheken

Die verwendete Software muss, wie in Abschnitt 3.6 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

6.2.6 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.

6.3 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.

6.4 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.

6.4.1 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.

6.4.2 Spring Profiles

Um flexibel zwischen verschiedenen Datenbankumgebungen wechseln zu können, wurden Spring Profiles implementiert. Die Konfiguration ist auf die drei Dateien aus Tabelle 4 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.

6.4.3 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.

6.4.4 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 Produktionsdaten strikt getrennt bleiben.

6.5 Development-Server

Der Development-Server wird in Abbildung 1, dargestellt im folgenden soll dieser näher beschrieben werden.

Für die Entwicklung wurde ein eigener 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 mithilfe von WireGuard realisiert wurde. WireGuard wurde ausgewählt, da es einfach einzurichten ist, zuverlässig funktioniert und eine sichere Verbindung gewährleistet.

Die Verbindung zur Datenbank erfolgt über das VPN mithilfe von WireGuard. Sobald die Verbindung hergestellt ist, können die Entwickler auf die Datenbank zugreifen und mit den vorbereiteten Mock-Daten arbeiten. Dadurch wird eine gemeinsame, einheitliche Grundlage geschaffen, die die Entwicklung deutlich erleichtert.

Damit ist die Basis gelegt, eine Datenbank mit einheitlichen Testdaten und ein VPN für den sicheren Zugang zu verwenden. Für das Projekt sind die folgenden Konfigurationen ausreichend, um stabil und konsistent arbeiten zu 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.

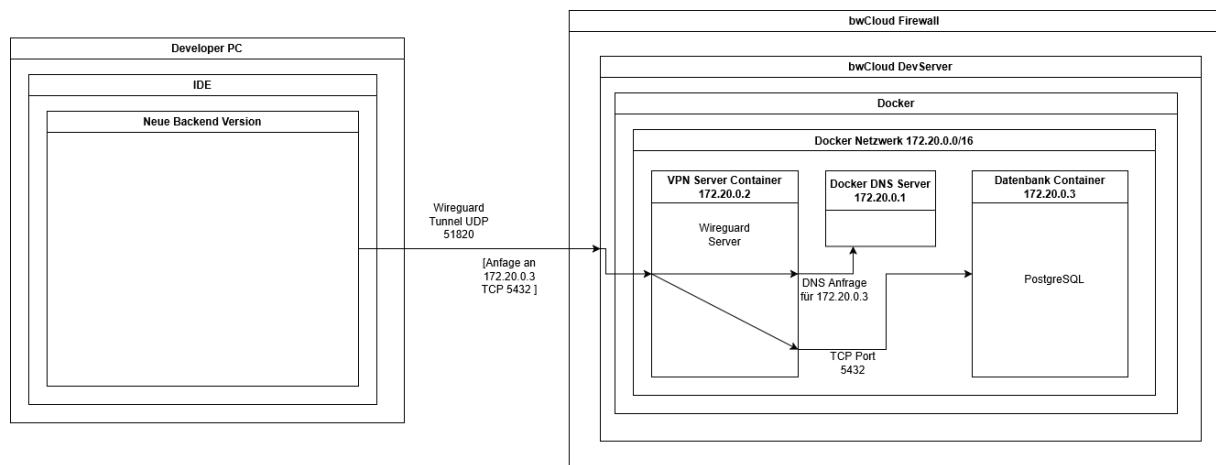


Abbildung 1: Netzwerkübersicht Development-Server

6.6 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.

CI/CD Pipeline

Zur Automatisierung des Entwicklungsprozesses wurde eine CI/CD Pipeline entworfen. In Abbildung 2 ist der erste Entwurf der CI/CD-Pipeline zu sehen. Die 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 unsauberer 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 deployt wird, in welchem Integrations-Tests durchgeführt werden , um die App als ganzes zu testen
6. Falls alle Tests positiv durchlaufen wird das Image zur Registry hinzugefügt und ist bereit auf dem Production-Server zu laufen

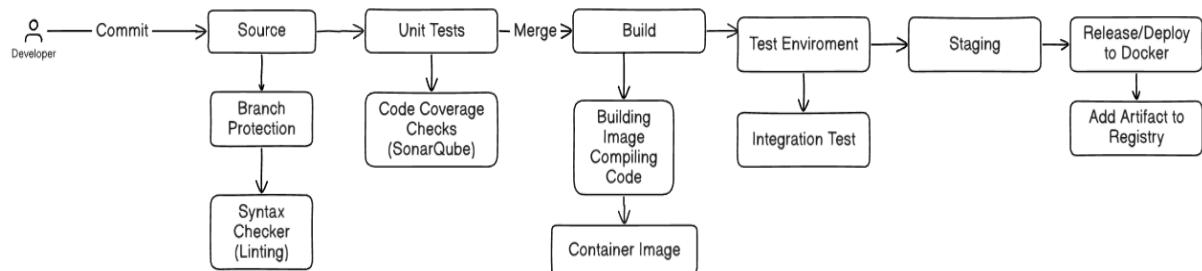


Abbildung 2: Alter Entwurf der CI/CD Pipeline Architektur in GitHub

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 Abbildung 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 6.7 erläutert wird.

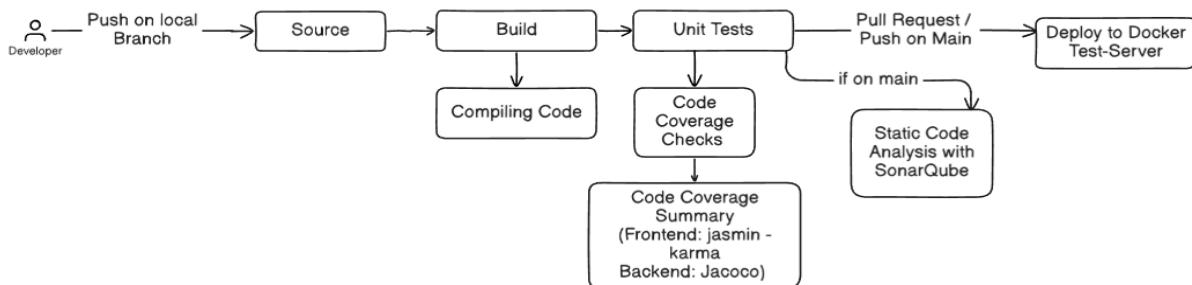


Abbildung 3: Finale CI/CD Pipeline in GitHub

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.

6.7 API-Testing mit Postman

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.

6.7.1 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`.

6.7.2 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_expiry`: Unix-Timestamp des Token-Ablaufs

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

6.7.3 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

6.7.4 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.

7 User Research

Die Entwicklung einer benutzerzentrierten Software erfordert ein tiefes Verständnis der Zielgruppe. In diesem Kapitel wird die Analyse der potenziellen Nutzer, ihrer Arbeitsweisen und der Probleme im aktuellen analogen Prozess dokumentiert, um sicherzustellen, dass LeihSy reale Bedürfnisse adressiert.

7.1 Proto-Personas

Zur Greifbarmachung der Anforderungen wurden exemplarische Nutzerprofile erstellt. Diese sogenannten Proto-Personas repräsentieren die wichtigsten Anwendergruppen – Studierende und Verleiher – und veranschaulichen deren unterschiedliche Ziele, Motivationen und Frustrationen.

7.1.1 Persona A – Lena Schmid (Studierende)

Die Abbildung 4 zeigt eine Persona, die die Nutzergruppe der Studenten visualisieren soll.

- **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.

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.

ALTE	26
BERUF	Medieninformatik Studentin

ZIELE

- Schnell sehen, was verfügbar ist & einfach anfragen
- Zeitfenster planen & bei Konflikten klare Hinweise
- Transparenz über Status & Historie

BEDÜRFNISSE

- 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.
- Automatische Bestätigungen und Erinnerungen, damit keine Anfragen vergessen oder überfällig werden.
- Transparente Nachverfolgung des eigenen Status und früherer Anfragen, ohne extra nachfragen zu müssen.

PAIN POINTS

- Unklare Verfügbarkeit/Konflikte
- Lange Wartezeiten/keine Antwort
→ Auto-Storno hilft
- Fehlende Erinnerungen → Überfälligkeit

Abbildung 4: Persona Lena Schmid

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

Die Abbildung 5 zeigt eine Persona, die die Nutzergruppe der Administratoren visualisieren soll.

- **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-Leisystem

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 5: Persona Max Schmidt

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

Die Abbildung 6 zeigt eine Persona, die die Nutzergruppe der Verleiher visualisieren soll.

- **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 <ul style="list-style-type: none"> • Schnell prüfen & entscheiden: Anfragen annehmen/ablehnen/Termine abstimmen • Eigene Gegenstände im Blick behalten • Vor Ort: Ausgabe/Rückgabe sauber dokumentieren • Nachhaltigkeit fördern: Ressourcenverbrauch senken BEDÜRFNISSE <ul style="list-style-type: none"> • Dashboard offene Anfragen + Filter/Sortierung • Alternativtermine vorschlagen • E-Mail-Trigger für Abholung, Rückgabe • Zentrale Plattform: z. B. Integration in die Hochschul-App PAIN POINTS <ul style="list-style-type: none"> • 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 6: Persona Tom Fischer

7.2 Interview Auswertung

Für das Projekt „Leihsy“ wurden Studierende der Hochschule interviewt. Dabei wurde der 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 konnten alle Studierenden den UI-Prototyp problemlos bedienen und alle wichtigen Funktionen ohne Schwierigkeiten finden. Die Farbgestaltung wurde als passend bewertet, und die Elemente waren auf den ersten Blick klar erkennbar. Im entsprechenden Ranking wurden 4,5 von 5 Punkten erreicht.

Der halbe Punkt Abzug resultiert aus der Darstellung des Raums für die Geräteabholung. Da über dem Raum der Campus mit einem Standort-Emoji angezeigt wird, war für die Studierenden nicht sofort ersichtlich, wo sich der Raum genau befindet. Das Emoji lenkte die Aufmerksamkeit leicht vom Raum ab.

7.3 Auswertung (Online-Umfrage, n = 28)

Ergänzend zu den qualitativen Proto-Personas wurde eine quantitative Online-Umfrage durchgeführt, um die Annahmen auf eine breitere Datenbasis zu stützen. Im Folgenden werden die Rückmeldungen der 28 Teilnehmenden ausgewertet, um konkrete Muster im Nutzungsverhalten und die Dringlichkeit bestimmter Features zu identifizieren.

7.3.1 Stichprobe & Nutzung

In der Abbildung 7 wird die Nutzung der Ausleihen von der Hochschule thematisiert.

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

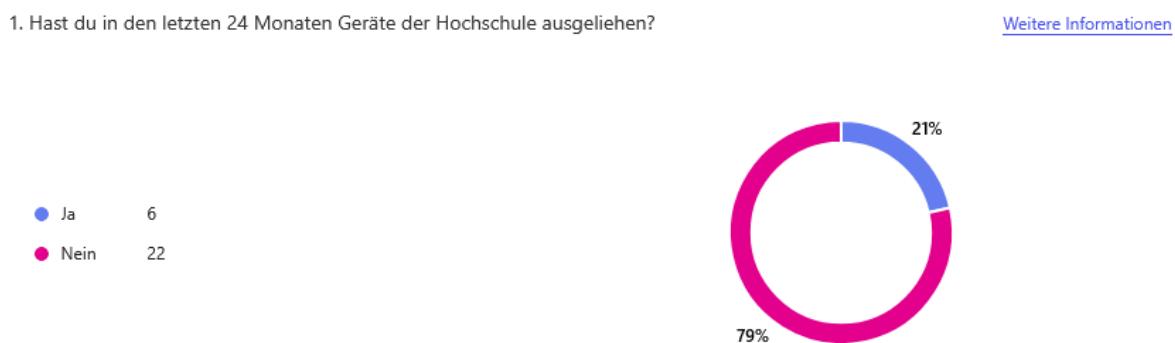


Abbildung 7: Anteil Studenten mit kürzlichen Ausleihen

7.3.2 Zufriedenheit (Ist-Prozess)

In der Abbildung 8 wird die Zufriedenheit des Ausleihprozess an der Hochschule thematisiert.

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

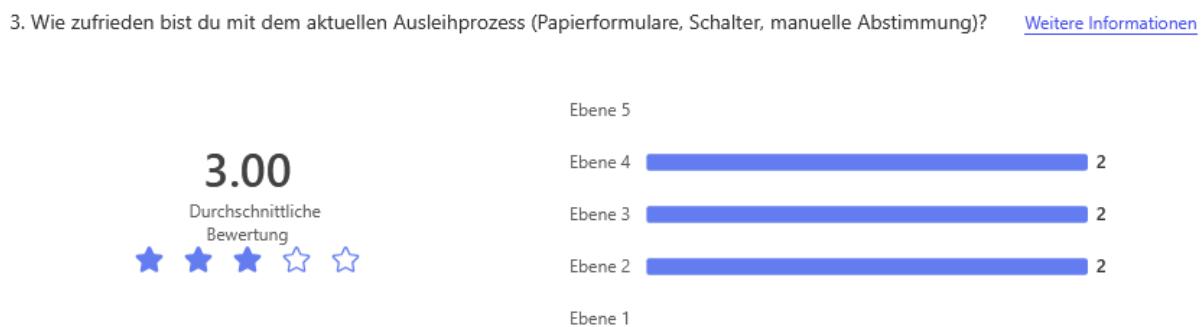


Abbildung 8: Zufriedenheit mit aktuellem Prozess

7.3.3 Größte Pain Points (Top-Nennungen)

In der Abbildung 9 werden die Painpoints des Ausleihprozess an der Hochschule thematisiert.

- Formulare/Unterschriften 5
- Unklare Verfügbarkeit 3
- Rückgabe unflexibel 3
- Abholung nicht planbar 2

⇒ Papier & manuelle Abstimmung sind die Hauptbremsen; 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

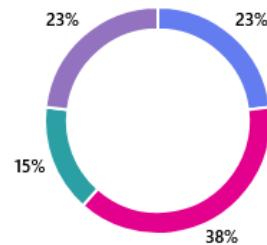


Abbildung 9: aktuelle Pain Points

7.3.4 Adoptionsbereitschaft für ein digitales System

In der Abbildung 10 wird die Adoptionsbereitschaft für ein digitales Ausleihsystem an der Hochschule thematisiert.

- 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

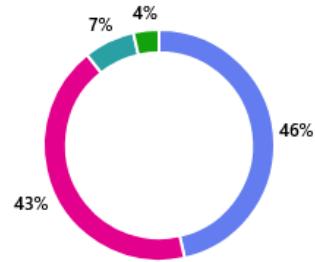


Abbildung 10: Adoptionsbereitschaft für digitales System

7.3.5 Top-Features (max. 3 Stimmen)

In der Abbildung 11 werden die Features, die sich die potenziellen Nutzer am meisten wünschen thematisiert.

- 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](#)

● Live-Verfügbarkeit	16
● Standort-Filter (Campus)	11
● klares Zubehör-Listing	7
● maximale Ausleihdauer sichtbar	8
● Online-Verlängerung	12
● Erinnerungen vor Rückgabe	17
● transparente Gebühren	7

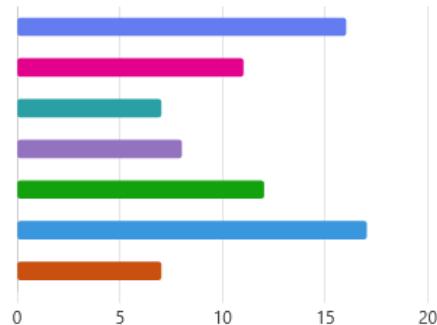


Abbildung 11: Wichtigste Features

7.3.6 Unverzichtbare Filter

In der Abbildung 12 werden die Filter, die im Katalog vorhanden sein sollen thematisiert.

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

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

[Weitere Informationen](#)



Abbildung 12: Unverzichtbare Filter

7.3.7 Verfügbarkeit je Campus

In der Abbildung 13 wird die Nützlichkeit einer Filterung nach dem Campus thematisiert.

- 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](#)

● gar nicht hilfreich ● eher nicht hilfreich ● eher hilfreich ● sehr hilfreich

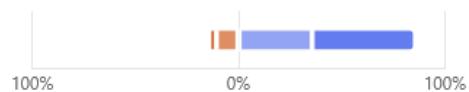


Abbildung 13: Wichtigkeit Verfügbarkeit nach Campus

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

In der Abbildung 14 wird die Länge der Möglichkeit die Buchung zu stornieren thematisiert.

- 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 14: Länge Stornierbarkeit

8 UI-Prototyp

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

8.1 Farbpalette

Die visuelle Identität von LeihSy orientiert sich am Corporate Design der Hochschule, um einen hohen Wiedererkennungswert und Vertrauen zu schaffen. Zusätzlich zur Markenidentität erfüllt die Farbwahl funktionale Zwecke: Sie lenkt die Aufmerksamkeit des Nutzers, kennzeichnet den Status von Ausleihen (z. B. rot für überfällig) und gewährleistet durch ausreichende Kontraste die Barrierefreiheit der Benutzeroberfläche.

8.1.1 Version 1 (Erstkonzept)

Schon in der ersten Version wurden 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.

8.1.2 Version 2 (Wireframe)

Die Farben aus Version 1 wurden übernommen und leicht verfeinert, um mehr Klarheit und Konsistenz zu schaffen. Dabei wurde auf neutrale UI-Farben gesetzt:

- Primär: #0A0AOA
- Sekundär: #717182
- Ausgewählt: #E9EBEF
- Buttons: #000000
- Dazu die Statusfarben: #00A63E (verfügbar) und #C10007 (ausgeliehen).

8.1.3 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).

8.2 Erste Prototypen

Um die Anforderungen frühzeitig zu visualisieren und das Bedienkonzept zu validieren, wurden erste visuelle Entwürfe der Benutzeroberfläche erstellt. Diese Prototypen dienten als Diskussionsgrundlage mit den Stakeholdern und halfen dabei, logische Abläufe und das Layout vor der technischen Umsetzung zu schärfen.

8.2.1 Version 1 – Grundidee

- **Anmeldung & Rollen:** Eine einfache Anmeldemaske mit klarer Rollenauswahl (Studierende, Dozierende, Personal). (Siehe Abbildung 15)

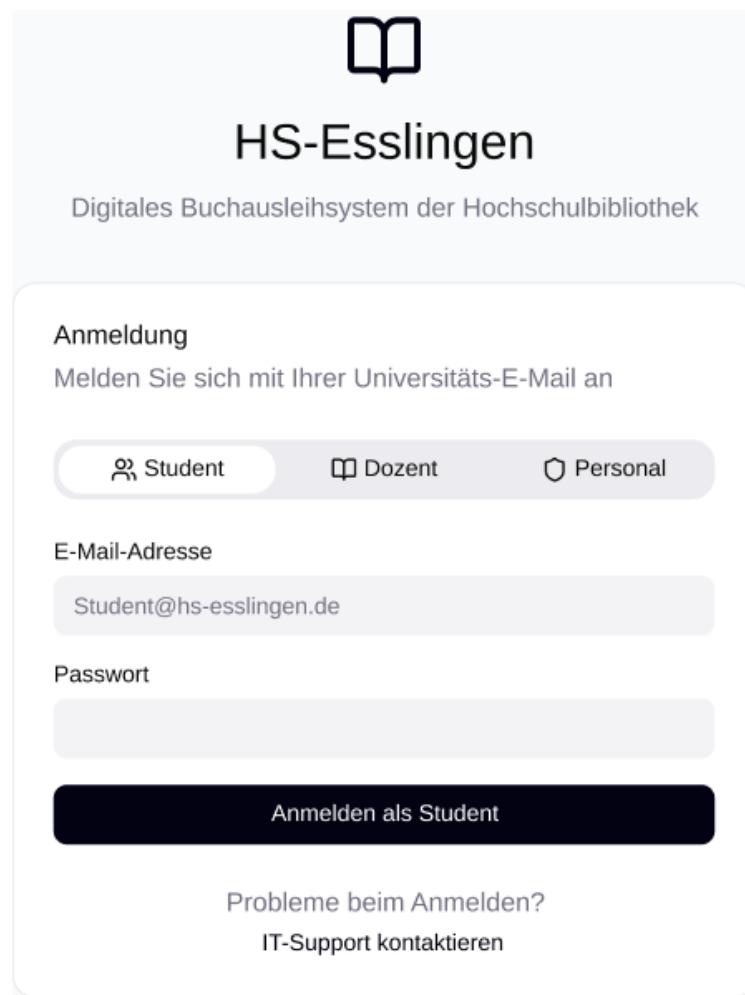
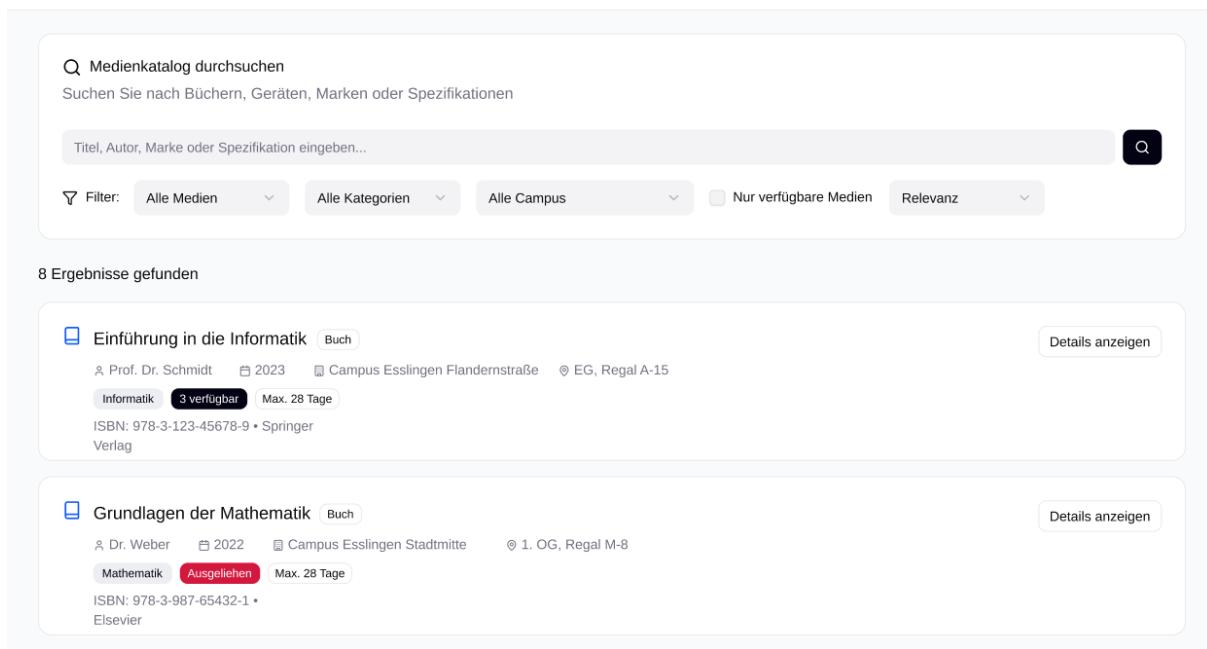


Abbildung 15: Login screen

- **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 16)



The screenshot shows the library catalog search interface. At the top, there is a search bar labeled "Q Medienkatalog durchsuchen" with the 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". There are two search results 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 16: 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 17)

The screenshot displays the HS-Esslingen library website's product detail page for the Meta Quest 3 VR-Brille. At the top, there are navigation links for 'HS-Esslingen' (with a logo), 'Katalog', 'Mein Bereich', and an email address 'asesit00@hs-esslingen.de' for 'Student'. Below the header, a backlink '← Zurück zur Suche' is visible.

Product Information:

- Name:** Meta Quest 3 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
- Included 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 Options: 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 of the page, a footer note reads: © 2025 Hochschule Esslingen - University of Applied Sciences. Alle Rechte vorbehalten. Datenschutz • Impressum • Barrierefreiheit.

Abbildung 17: Geräte-Details Seite

- **Mein Bereich:** Übersichtsseite mit aktuellen Ausleihen, Fälligkeiten, Gebühren und Verlängerungsmöglichkeit. (Siehe Abbildung 18)

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 18: Persönlicher Bereich

8.2.2 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 19)

The screenshot shows the HS-Esslingen inventory catalog interface. At the top, there is a header with the logo, navigation links (Katalog, Mein Bereich, Abmelden), and a search bar. Below the header, the title "Medienkatalog durchsuchen" is displayed, followed by a subtitle "Nutzen Sie nach Stichworten, Geräten, Verleih- oder Downloadmedien". A search input field and a "Suchen" button are present. The main content area shows search filters: "Alle Kategorien", "Alle Campus", and "Alle Medien". It displays 14 results found for "Meta Quest 3". Each result card includes the item name, category (VR-215 - VR-Geräte), location (Campus Esslingen Flandernstraße, Campus Esslingen Stadtmitte, Campus Göppingen), availability status (Verfügbar: 2/2 or Ausgeliehen), and a "Details ansehen" link. Below the first result, a note states: "Hochmoderne VR-Brille für immersive virtuelle Erfahrungen. Ideal für Forschungszwecke, Entwicklung und Lehrzwecke im Bereich Game Design und Informatik." Another section shows results for "Canon EOS R6 Mark II", with similar details and notes about its professional use for photo and video projects.

Abbildung 19: 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 20 und 21)

[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 20: Kalender in Produktdetails

61

[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:

Zeitfenster:
Zeitfenster wählen

Kategorisierung

VR-Geräte

Support

geraeieverleihs@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

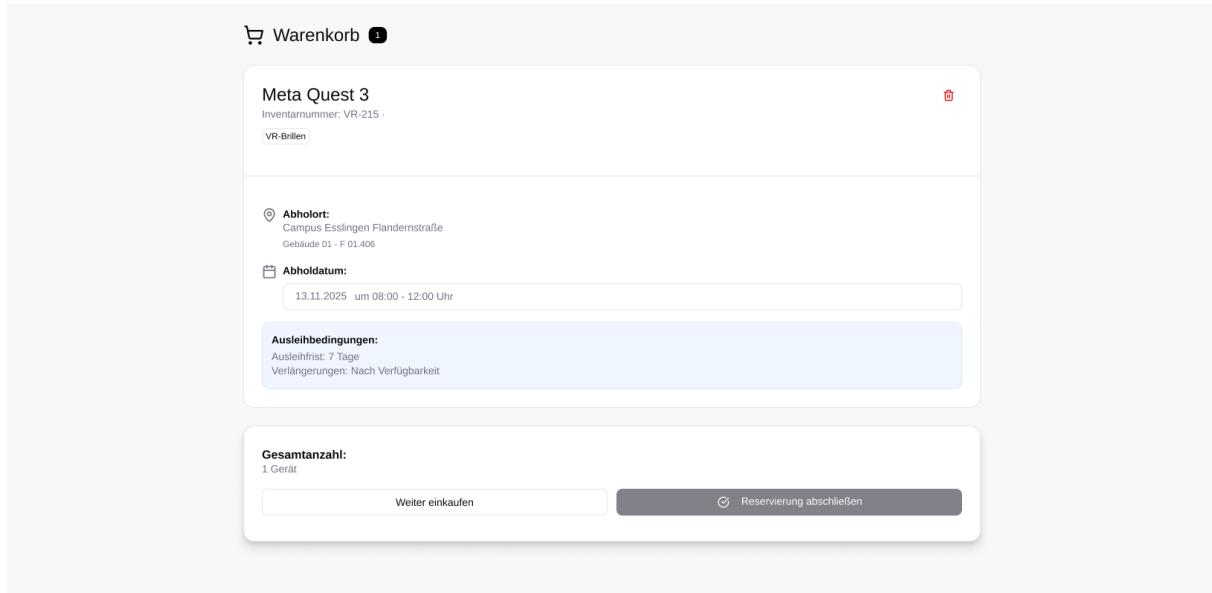
Professionelles System für Studierende und Lehrende der Hochschule Esslingen.

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 21: Abholungszeitfenster

- Auf Wunsch des Kunden wurde ein Warenkorb integriert. Darin können Nutzer ihre gewünschten Geräte direkt hinzufügen, einsehen und verwalten. (Siehe Abbildung 22)

**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 22: 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 23)

[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

geraeieverleih@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 23: 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 24)

Willkommen zurück

Meine Ausleihen

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

Sony A7 IV Inventory-Nr.: K-512 Campus Esslingen Flandernstraße	Ausgeliehen am: 10.09.2025	Rückgabe bis: 17.09.2025	0 Tage
Ausleihen: 0/1 Verlängerungen genutzt	Verlängern Details		
Meta Quest 2 Inventory-Nr.: VR-225 Campus Esslingen Flandernstraße	Ausgeliehen am: 10.10.2025	Rückgabe bis: 17.10.2025	0 Tage
Ausleihen: 0/1 Verlängerungen genutzt	Verlängern Details		
Lichtset Aputure 300d II Inventory-Nr.: L-450 Campus Esslingen Flandernstraße	Ausgeliehen am: 18.10.2025	Rückgabe bis: 25.10.2025	5 Tage
Ausleihen: 0/1 Verlängerungen genutzt	Verlängern Details		

Schnelle Verlängerungen

Verlängern Sie mehrere Medien gleichzeitig

Sony A7 IV K-512	0d
Meta Quest 2 VR-225	0d
Lichtset Aputure 300d II L-450	0d

Ausgewählte verlängern

Zukünftige Abholungen

Ihre reservierten Geräte zur Abholung

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

Support

geraeiterverleih@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

Abbildung 24: Aktualisierter persönlicher Bereich

8.3 Visuelle Umsetzung der Funktionen

In diesem Abschnitt wird das finale User Interface der Anwendung präsentiert. Basierend auf den zuvor definierten Design-Richtlinien und Prototypen wurden die funktionalen Anforderungen in konkrete Views übersetzt. Die folgenden Darstellungen zeigen die Um-

setzung der zentralen Workflows aus der Perspektive der verschiedenen Benutzerrollen.

8.3.1 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 25)

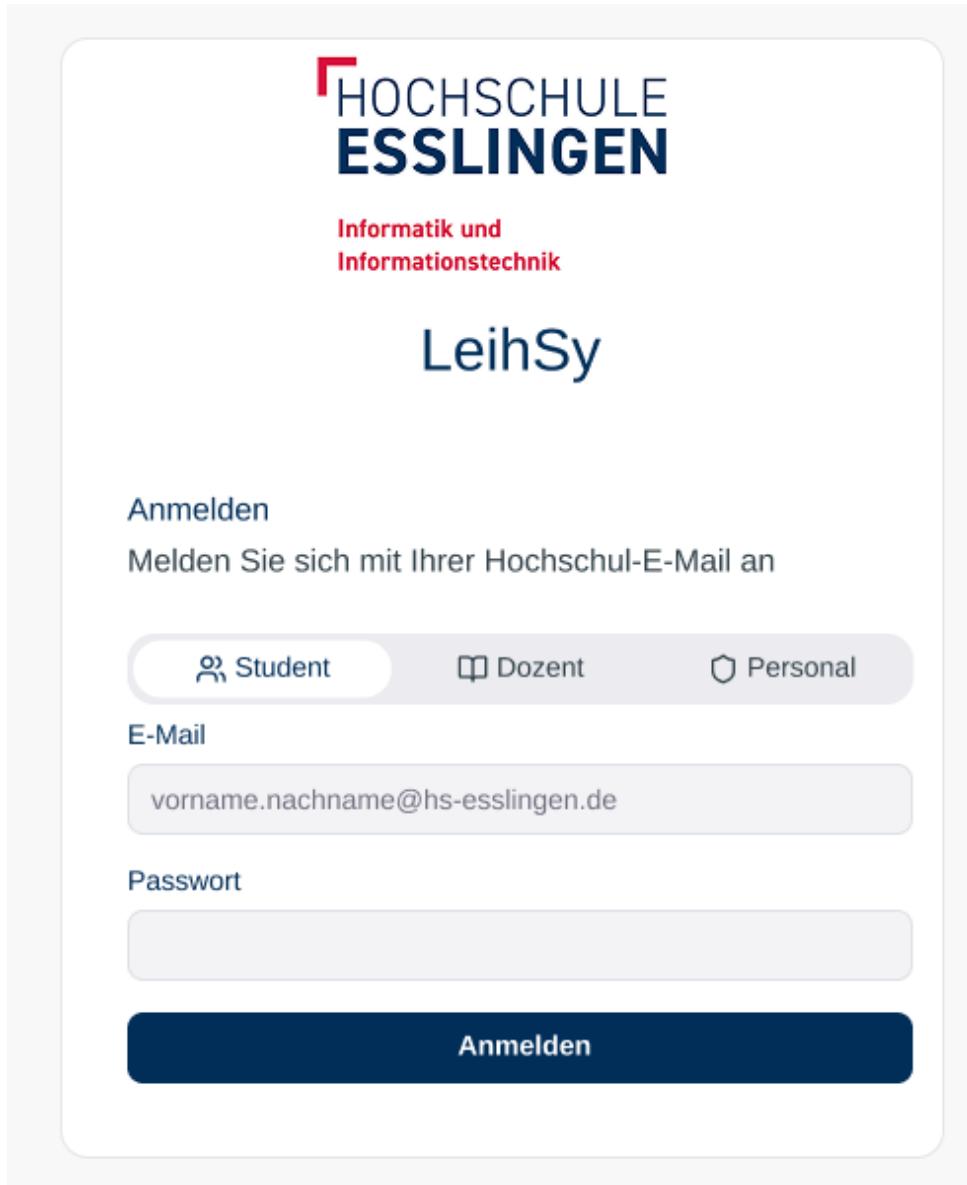


Abbildung 25: Aktualisierter Login screen

8.3.2 Geräte-Details

Oben links wurde das Logo der Hochschule eingefügt, und die Farben wurden einheitlich angepasst. Die finale Version entspricht genau unserer Vorstellung. (Siehe Abbildung 26)

The screenshot shows the Hochschulbibliothek website's device details page for a "Meta Quest 3".

Device Details:

- 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
- Number:** VR-215 - **Model:** Meta Quest 3

Borrowing 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

Reservation:

Abholtermin wählen

Abholdatum: Datum wählen

Zeitfenster: Zeitfenster wählen

Termin reservieren

Categorization:

VR-Geräte

Support	Ausgabestellen	LeihSy
<p>✉ geraeteverleihe@hs-esslingen.de 📞 +49 711 397-3456 Für technische Probleme oder Fragen zur Ausleihe</p>	<p>⌚ Mediennausgabe 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 ⌚ Kamerás: 15+ Profi-Kamerás ⌚ Lichtsets & Equipment: 15+ Sets</p>

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

Abbildung 26: Geräte-Details Seite mit aktualisierten Design

8.3.3 Warenkorb

Das Warenkorb Logo wurde so gestaltet, dass oben die Anzahl der hinzugefügten Geräte in Nummern dargestellt wird. Außerdem wird der Bereich für Terminreservierungen in Rot dargestellt, damit dieser sofort ins Auge fällt und die Aufmerksamkeit des Betrachters auf sich zieht. (Siehe Abbildung 27)



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 27: Verbesserter Warenkorb

9 Softwarearchitektur

Das Kapitel Softwarearchitektur befasst sich mit der Planung und Visualisierung zentraler Kernkonzepte der Web-Anwendung und soll der Übersicht dienen.

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

Das Frontend bildet die sichtbare Schnittstelle für die Anwender und wurde als responsive Webanwendung realisiert. In diesem Abschnitt wird die technische Struktur der Client-Seite sowie die Umsetzung der Interaktionslogik erläutert, um eine intuitive Bedienung auf verschiedenen Endgeräten sicherzustellen.

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.

9.3 Backend

Das Backend stellt die zentrale Verarbeitungseinheit der Anwendung dar und ist für die Geschäftslogik, Datenhaltung sowie die Bereitstellung der Schnittstellen für das Frontend verantwortlich. Ziel der Architektur ist eine klare Trennung von Verantwortlichkeiten, eine hohe Wartbarkeit sowie eine gute Erweiterbarkeit des Systems.

Struktursicht

Die Backend-Architektur orientiert sich am *Model-View-Controller*-Prinzip (MVC) und setzt ergänzend auf das Repository Pattern sowie eine konsequente *Separation of Concerns*. Die Struktur gliedert sich in mehrere logisch getrennte Schichten:

1. **Controller:** Die Controller bilden den Einstiegspunkt für externe Anfragen (z. B. REST-Endpunkte). Sie nehmen Client-Anfragen entgegen, validieren die Eingabedaten und koordinieren den Ablauf, ohne selbst Geschäftslogik zu enthalten. Die Weiterverarbeitung erfolgt durch delegierte Service-Komponenten.
2. **Service:** Die Service-Schicht kapselt die fachliche Geschäftslogik der Anwendung. Sie verarbeitet die vom Controller übergebenen Daten, orchestriert Abläufe und greift bei Bedarf auf mehrere Repositories zu. Dadurch bleibt die Logik unabhängig von der Art der Datenpersistenz.
3. **Repository:** Repositories stellen den Zugriff auf die Persistenzschicht bereit. Sie abstrahieren Datenbankoperationen wie Lesen, Schreiben oder Aktualisieren von Entitäten und entkoppeln die Geschäftslogik von der konkreten Datenbanktechnologie.

4. **Model:** Die Model-Schicht repräsentiert die Domänenobjekte der Anwendung. Sie bildet die Struktur der gespeicherten Daten ab und entspricht in der Regel den Datenbankentitäten.
5. **DTO (Data Transfer Objects):** DTOs dienen dem kontrollierten Datenaustausch zwischen den einzelnen Schichten sowie zwischen Backend und Frontend. Sie verhindern eine direkte Exposition der internen Model-Strukturen und ermöglichen eine gezielte Anpassung der übertragenen Daten.
6. **Mapper:** Mapper übernehmen die Transformation zwischen Model-Objekten und DTOs. Dadurch wird eine saubere Trennung zwischen interner Datenrepräsentation und externen Schnittstellen gewährleistet.

Durch die Kombination aus MVC, Repository Pattern und klar getrennten Schichten wird eine modulare, testbare und wartungsfreundliche Backend-Architektur erreicht, die eine stabile Grundlage für zukünftige Erweiterungen bildet.

9.4 Entity-Relationship-Diagramme

Die Datenbank ist eine elementare Komponente für die Web-Anwendung und dient dazu Daten die zum Betrieb der Anwendung erforderlich sind konsistent abzuspeichern. In den folgenden Abschnitten wird der Aufbau der Datenbank und die Relationen der einzelnen Tabellen behandelt.

9.4.1 Erster Entwurf

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

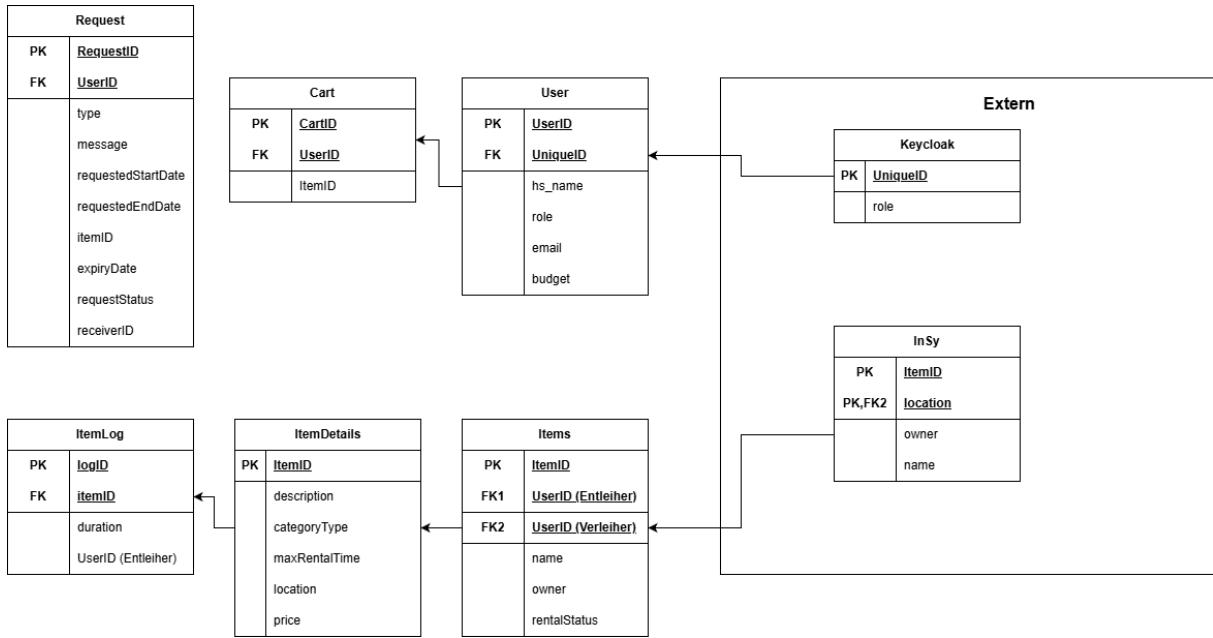


Abbildung 28: Erster Entwurf Entity-Relationship-Diagramm

9.4.2 Änderungen vom ersten Entwurf

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:

9.4.2.1 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.

9.4.2.2 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.

9.4.2.3 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.
- `Booking_Transactions`: Eine neue Entität zur Erstellung des QR-Codes während dem Abhol- und Rückgabeprozess.
- `Student_Groups`: Eine neue Entität zur Erstellung von Studentengruppen für Projekte.
- `Student_Groups_Members`: Eine neue Entität zur Verwaltung von Nutzern in den Studentengruppen.
- `Insy_Import_Items`: Eine neue Entität zur temporären Speicherung und Verwaltung von Gegenständen, die über die Schnittstelle zur InSy Applikation bereitgestellt werden.

9.4.2.4 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.

9.4.2.5 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.

9.4.2.6 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.

9.4.3 Aktualisierung des Entity-Relationship-Diagramm

Das in Abbildung 29 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.

9.4.3.1 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.

9.4.3.2 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.

9.4.3.3 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).

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

9.4.3.5 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.

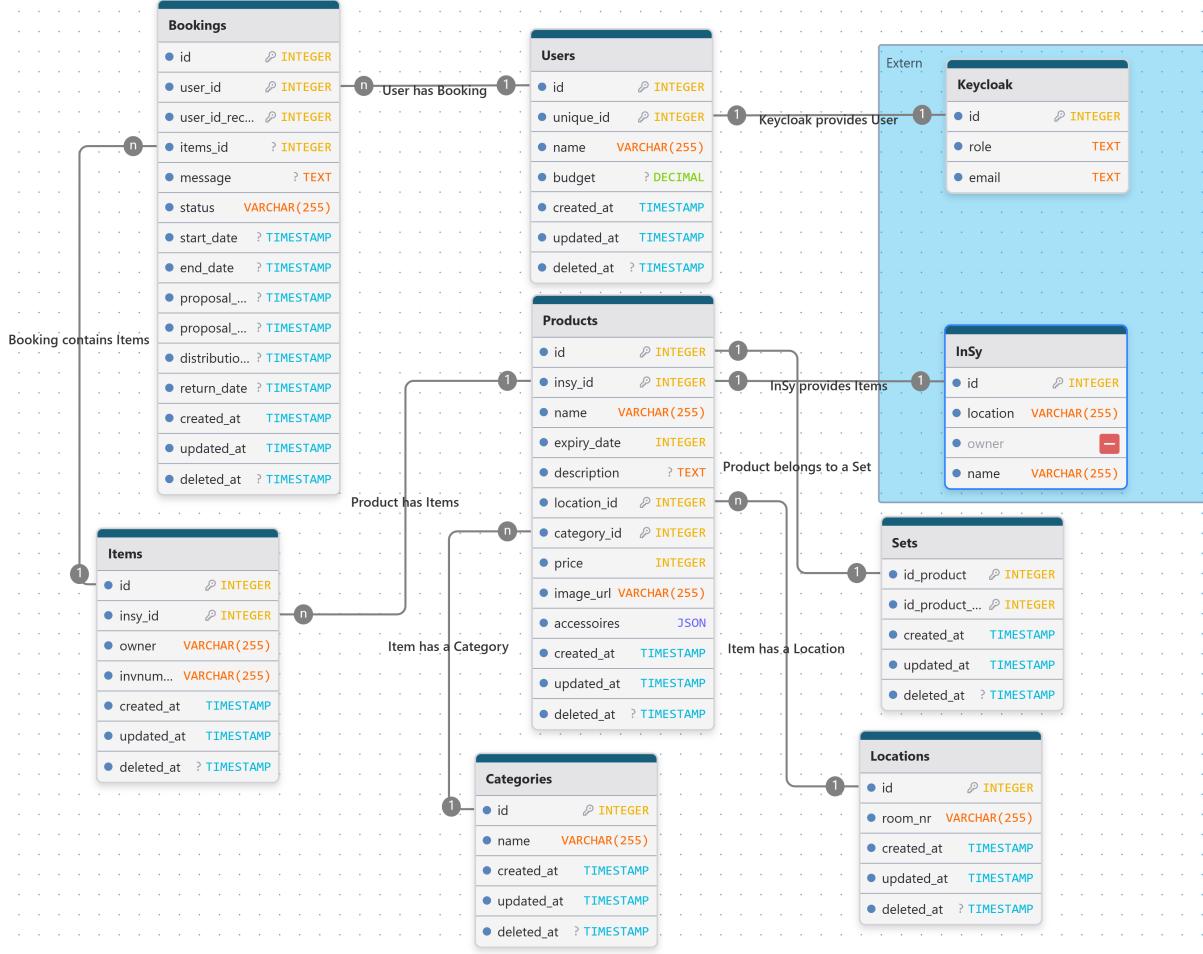


Abbildung 29: Entity-Relationship-Diagramm

9.5 Logische Sichten

9.5.1 Verteilungssicht

Die Anwendung folgt einem klassischen Client-Server-Architekturmmodell, wie in Abbildung 30 dargestellt. Der Client ist der Webbrower, der auf dem Endgerät des Benutzers ausgeführt wird. Möchte der Benutzer die Anwendung aufrufen, sendet sein Brower 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.

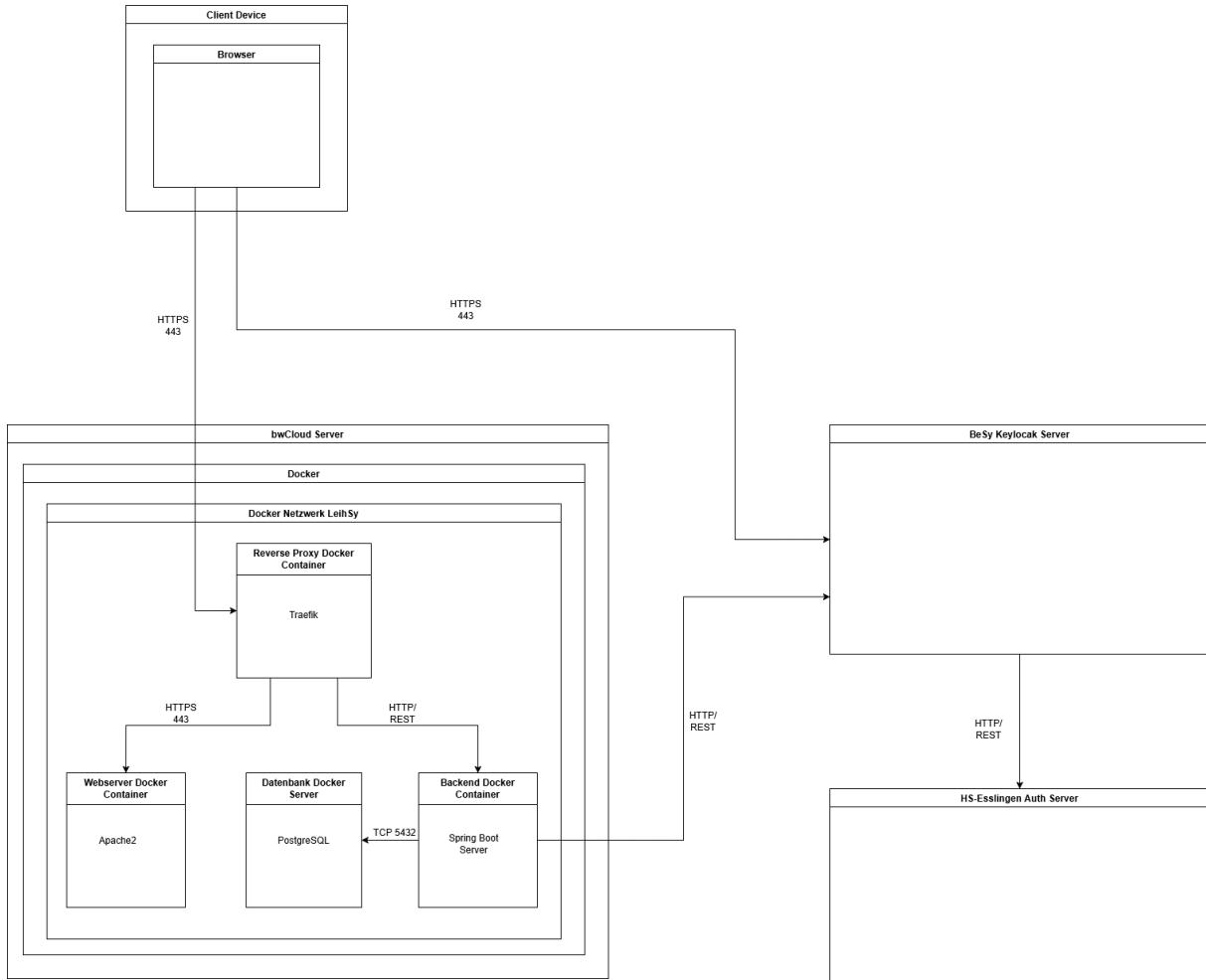


Abbildung 30: Diagramm zur Verteilungssicht

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 Co-

de 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.

9.5.2 Struktursicht

Das Komponentendiagramm in Abbildung 31 beschreibt die Architektur des LeihSys-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.

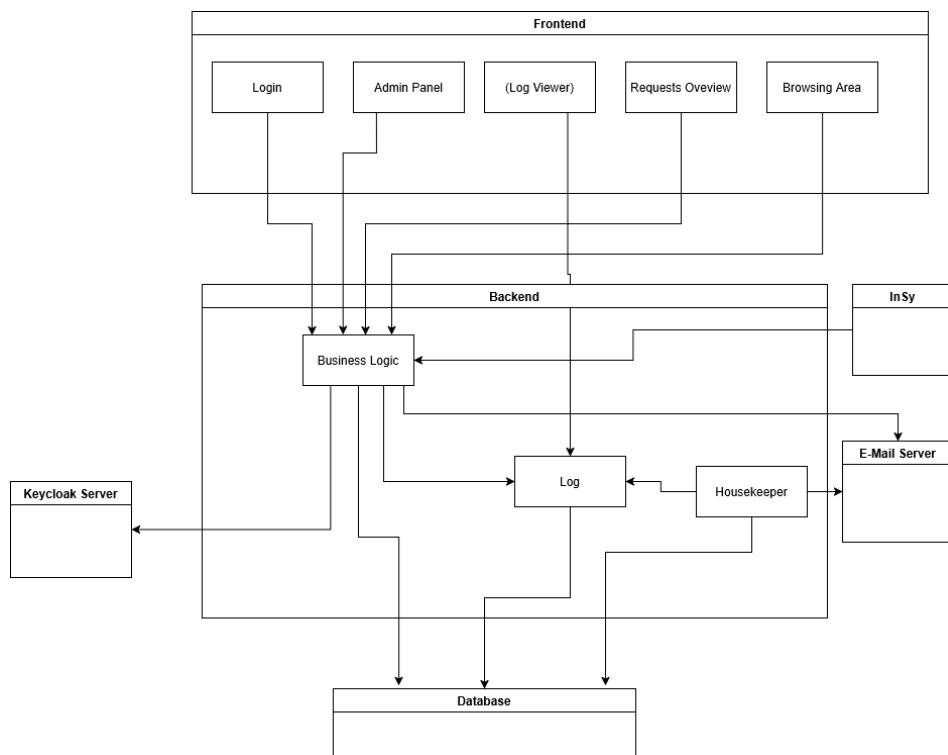


Abbildung 31: Komponentendiagramm

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.

9.5.3 Verhaltenssicht

Ergänzend zur statischen Struktur beschreibt die Verhaltenssicht die Dynamik des Systems zur Laufzeit. In diesem Abschnitt wird der Lebenszyklus zentraler Geschäftsobjekte visualisiert – insbesondere der Statuswechsel von Ausleihen – um die komplexe Business-Logik und die möglichen Interaktionspfade transparent zu machen.

9.5.3.1 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 zu stimmen. Bei Letzterem schickt der Verleiher seine Termine an welchen er Zeit hat. Falls keine Reaktion des Verleiher erfolgt wird die Anfrage storniert.

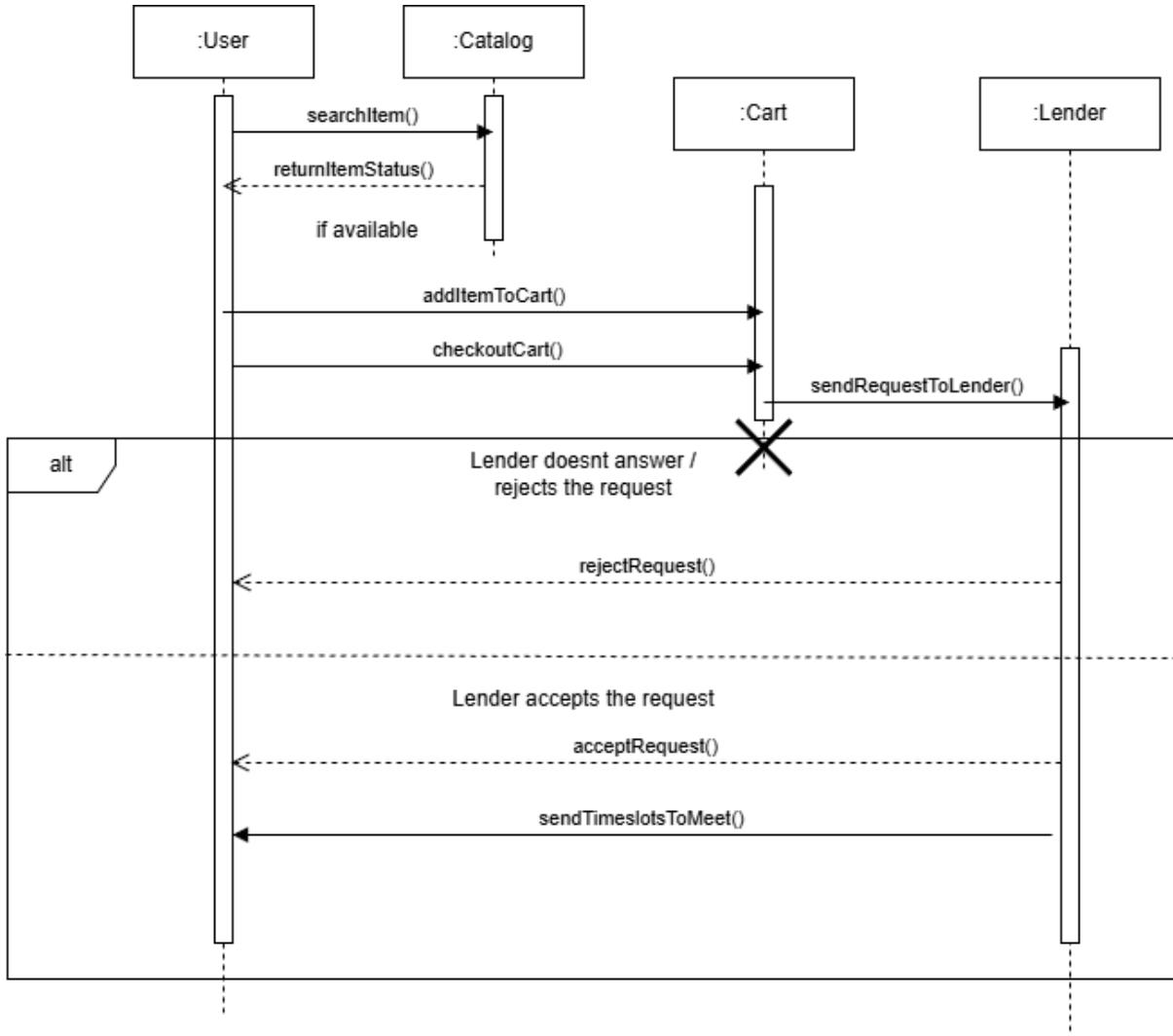


Abbildung 32: Sequenzdiagramm zum Ausleihprozess

Bei der Rückgabe bekommt der Nutzer eine Nachricht des Verwaltungssystem, 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.

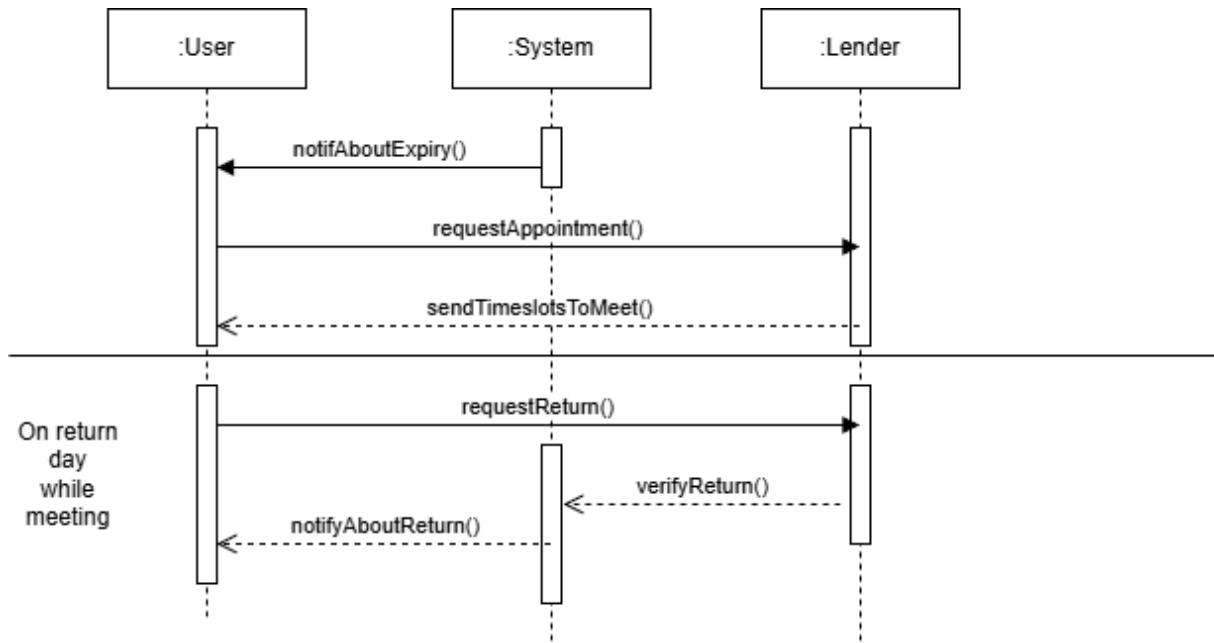


Abbildung 33: Sequenzdiagramm zum Rückgabeprozess

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.

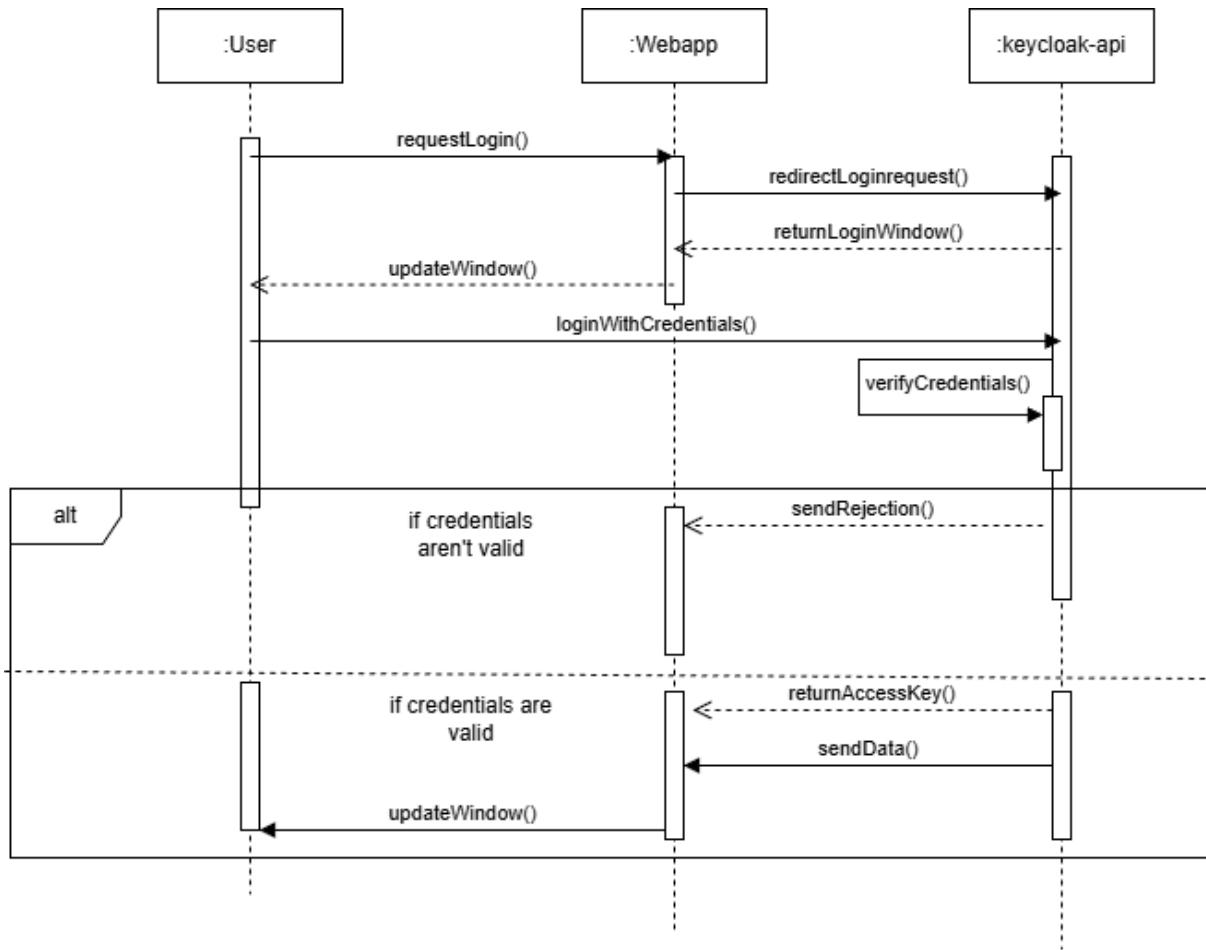


Abbildung 34: Sequenzdiagramm zum Login

9.6 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 Beispiele ausgaben 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.

API-Umstrukturierung nach REST Best Practices:

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.

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.

Phase 1: CRUD-Vervollständigung und Verb-Entfernung

Location-Controller 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).

Category-Controller 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`.

Image-Controller 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.

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.

Phase 2: Ressourcen-Hierarchie und `Product.isActive`

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.

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.

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.

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.

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.

Phase 3: Query-Parameter statt Sub-Routes

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.

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.

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.

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.

Phase 4: PATCH statt PUT und DELETE für Reject

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.

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.

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.

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

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.

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

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.

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.

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).

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.

10 Features

Im folgenden Abschnitt sollen die Funktionalitäten der Applikation näher betrachtet werden.

10.1 Admin Dashboard

Im folgenden werden die Funktionen, die sich im Admin-Dashboard befinden ausführlich beschrieben und erklärt, sowie durch Bilder visualisiert.

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.

10.1.1.1 Beschreibung des Menüs

Die Menüpunkte sind listenartig angeordnet und jeder Menüpunkt enthält:

- Icon
- Titel,
- Beschreibung des Menüs.

10.1.1.2 Änderungshistorie

- **Menü-Komponente** Erstellung einer Menü-Komponente zur einheitlichen Darstellung der Menüpunkte.

Begründung: Verbesserung der Übersichtlichkeit und Vereinheitlichung der Darstellung aller Administrationsfunktionen.

Auswirkungen: Schnellere Produktpflege und konsistentes UI-Design.

- **Änderung des Menü-Designs** Umstellung des Designs von quadratischen Kacheln auf flache, schmale Kacheln.

Begründung: Es passen mehr kleinere Kacheln auf den Bildschirm.

Auswirkungen: Schnellere Orientierung.

- **Änderung des Menü-Layouts** Anpassung des Layouts von zwei quadratischen Menükacheln pro 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)“ bietet eine vollständige Übersicht über sämtliche Gegenstände im System, wie in Abbildung 35 zu sehen. Sie dient Administratoren dazu, alle Produktinstanzen unabhängig vom übergeordneten Produkt zentral zu verwalten.

Einzelne Gegenstände verwalten

Erstelle, bearbeite oder lösche einzelne physische Gegenstände zu jedem Produkt

Produkte und ihre Gegenstände					
<input type="text"/> Suche nach Produktnamen, Kategorie...					
Meta Quest Pro Kategorie: VR-Equipment Standort: VR-Labor F01.403					8 / 11 verfügbar
High-End Mixed-Reality-Headset mit Eye-Tracking und Gesichtserkennung.					
ITEM ID ↑↓	INVENTARNUMMER ↑↓	BESITZER ↑↓	VERLEIHER ↑↓	STATUS ↑↓	AKTIONEN
31	VR-PRO-003	Christian Haas	admin@hs-esslingen.de	Ausgeliehen	
40	VR-PRO-008	christian.haas@hs-esslingen.de	christian.haas@hs-esslingen.de	Ausgeliehen	
39	VR-PRO-007	christian.haas@hs-esslingen.de	Anna Schmidt	Verfügbar	
41	VR-PRO-009	Christian Haas	Christian Haas	Verfügbar	
1	VR-001	Christian Haas	Andreas Heinrich	Ausgeliehen	

Abbildung 35: Übersicht über Gegenstände einer Produktgruppe

10.1.2.1 Beschreibung des Menüs

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:

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

10.1.2.2 Änderungshistorie

- **Verwendung der Tabellen-Komponente** 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.

- **Bearbeitungsverlinkungen** 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 Nutzererfahrung.

10.1.3 Admin-Menü: Produktgruppen verwalten

Das Menü "Produktgruppen verwalten" zeigt eine tabellarische Übersicht aller im System vorhandenen Produkte, vgl. Abbildung 36. Sie dient der zentralen Verwaltung des Produktbestands und ermöglicht den schnellen Zugriff auf Bearbeitungs- und Löschfunktionen.

ID ↑↓	NAME ↑↓	KATEGORIE ↑↓	STANDORT ↑↓	PREIS/TAG ↑↓	AKTIONEN
1	Meta Quest Pro	VR-Equipment	VR-Labor F01.403	8,00 €	
2	Valve Index	VR-Equipment	VR-Labor F01.403	11,00 €	
3	Pico 4	VR-Equipment	VR-Labor F01.403	6,00 €	
4	Canon EOS R6 Mark II	Kamera	F01.402 (KEIM)	18,00 €	
5	Nikon Z6 III	Kamera	Bibliothek Flandernstraße	17,00 €	
6	DJI Ronin SC	Kamera-Equipment	F01.402 (KEIM)	10,00 €	
7	Shure SM7B	Audio-Equipment	F01.402 (KEIM)	6,00 €	
8	Zoom H6	Audio-Equipment	Medienlabor S02.201	5,00 €	

Abbildung 36: Übersicht über alle Produktgruppen

10.1.3.1 Beschreibung des Menüs

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.

10.1.3.2 Änderungshistorie

- **Verwendung der Tabellen-Komponente** 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.
- **Verwendung von Button-Komponenten** Einsatz selbst erstellter Button-Komponenten zur einheitlichen Darstellung von Aktionen.
Begründung: Wiederverwendbarkeit, Wartbarkeit sowie einheitliche Farben für Aktionen.
Auswirkungen: Schnellere Komponentenpflege sowie konsistentes und intuitives UI-Design.
- **Dropdownmenüs** Hinzufügen von Dropdownmenüs für die Kategorie- und Standortauswahl.
Begründung: Schnelleres Auffinden von Kategorien und Standorten ohne Kenntnis der ID.
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.

10.1.4.1 Beschreibung des Menüs

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.

Für alle Services die in den Itemseiten verwendet werden, wurden Unit-Tests erstellt, um eine hohe Codequalität zu garantieren.

10.1.4.2 Änderungshistorie

- **Verwendung der Tabellen-Komponente** 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.
- **Verwendung von Button-Komponenten** Einsatz selbst erstellter Button-Komponenten zur einheitlichen Darstellung von Aktionen.
Begründung: Verbesserung der Wiederverwendbarkeit, Wartbarkeit sowie Sicherstellung einer einheitlichen Farbgebung für Aktionen.
Auswirkungen: Schnellere Komponentenpflege sowie konsistentes und intuitives UI-Design.
- **Vorschau bei Eingabe** Hinzufügen einer automatischen Vorschau für Besitzer und Verleiher bei der Eingabe von Namen oder IDs.
Begründung: Schnelleres Auffinden von Personen sowie Reduktion von Fehlern bei der Verleiher- und Besitzerzuweisung.
Auswirkungen: Geringere Fehleranfälligkeit bei Zuweisungen und verbesserte Usability.

10.1.4.3 Zusatzgegenstände

Beim Anlegen bzw. Bearbeiten eines Produkts gibt es im Formular einen eigenen Tab „Zusatzgegenstände“. Dort kann der Admin andere Produkte als Zusatzgegenstand zum Hauptprodukt verknüpfen und festlegen, ob sie „erforderlich“ oder „empfohlen“ sind. Oben steht ein kurzer Hinweistext, darunter gibt es eine Suche, um schnell ein Produkt zu finden.

In der UI werden zwei Bereiche angezeigt:

- **Ausgewählte Zusatzgegenstände:** Sobald etwas gewählt wurde, erscheint es hier mit Name, Kategorie und einem Badge („Erforderlich“/„Empfohlen“). Über Buttons kann man den Typ umschalten oder den Eintrag entfernen.
- **Verfügbare Produkte:** Liste aller Produkte (außer dem aktuell bearbeiteten Produkt), pro Eintrag mit zwei Buttons **Erforderlich** und **Empfohlen**.

10.1.4.4 Technische Umsetzung

- Die Auswahl wird als Liste `selectedRelatedItems` gespeichert (je Eintrag: `productId` + `type`).

- `toggleRelatedItem(productId, type)` fügt hinzu, entfernt (bei erneutem Klick) oder wechselt den Typ.
- Die Suche läuft über `relatedItemsSearch` und filtert die Liste in `availableRelatedProducts()`.
- Beim Speichern wird `relatedItems` mitgesendet, damit das Backend die Verknüpfung dauerhaft speichern kann.

10.1.4.5 Backend Implementierung

Das Feature verknüpft Zusatzgegenstände mit Hauptgegenstände im Backend, um passendes Zusatzgegenstand vorzuschlagen. Es wird unterschieden zwischen `Required` (zwingend nötig) und `Recommended` (empfohlen).

1. Datenmodell (`ProductSet`)

Statt einer Standard-`@ManyToMany`-Beziehung wird die Entity `ProductSet` genutzt. Sie dient als Verbindungstabelle und speichert neben den Produkt-IDs zusätzlich das Attribut `type`.

2. API

- **GET (Abfrage):** Beim Abruf eines Produkts (`GET /api/products/{id}`) nutzt der Mapper eine Hilfsmethode (`mapRelatedItems`). Diese wandelt die komplexen `ProductSet`-Entities aus der Datenbank in eine einfache, flache Liste (`relatedItems`) im JSON-Response um. Das Frontend erhält so direkt die ID, den Namen und den Typ (z. B. "required") des Zubehörs.
- **POST / PUT (Erstellen/Update):** Die Endpoints akzeptieren im JSON-Body eine Liste von Beziehungen. Der Controller reicht diese Liste an den `ProductService` weiter. Dort wird die Speicherung übernommen.

3. Speicherlogik (`ProductService`)

Die Methode `saveProductRelations` nutzt eine „Delete & Re-create“-Strategie:

- **Bereinigung:** Zuerst werden alle alten Beziehungen des Hauptprodukts gelöscht (`deleteByParentProductId`).
- **Neuanlage:** Die neue Liste aus dem POST/PUT-Request wird iteriert und frisch in der Datenbank angelegt.

4. Testdaten

Um das Feature testen zu können, wurde die Datenbank im `DataInitializer` (Dev-Profil) mit exemplarischen Daten befüllt. Hierbei wurden logische Produktkombinationen hardcodiert („geraten“) angelegt, um realistische Szenarien im Frontend zu simulieren.

10.1.5 Admin-Menü: Buchungen verwalten

Das Menü "Buchungen verwalten", siehe Abbildung 37, bietet dem Admin die Möglichkeit alle Buchungen zu sehen.

← Zurück zur Übersicht

Alle Buchungen verwalten

Übersicht über alle Buchungen und Ausleihen im System

Statistiken **Aktualisieren**

7 Aktuelle Ausleihen	1 Überfällige Ausleihen	0 Offene Anfragen	1 Bestätigt, nicht abgeholt
1 Zukünftige Ausleihen	75 Alle Buchungen		

Suche nach Benutzer, Produkt, Inventarnummer...

AUSLEIHER ↑↓	PRODUKT ↑↓	INVENTARNUMMER ↑↓	VERLEIHER ↑↓	STATUS ↑↓	ABHOLUNG ↑↓	RÜCKGABE ↑↓	ERSTELLT AM ↴
admin@hs-esslingen.de	Meta Quest Pro	VR-PRO-007	Anna Schmidt	Abgelehnt	16.01.2026	21.01.2026	17.01.2026 16:11
admin@hs-esslingen.de	Meta Quest Pro	VR-PRO-007	Anna Schmidt	Abgelehnt	23.01.2026	28.01.2026	17.01.2026 01:55

Abbildung 37: Übersicht über alle Buchungen

10.1.5.1 Beschreibung des Menüs

Die Buchungen werden tabellarisch aufgelistet beginnend mit den neusten Buchungen. Die Tabelle hat folgende Spalten:

- **Ausleiher:** 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, welche in Abbildung 38 dargestellt ist. Auf dieser kann ein Zeitraum ausgewählt werden, für welchen die Statistiken erstellt werden sollen, dabei kann man aus vordefinierten Zeiträumen auswählen sowie benutzerdefinierten.

Buchungsstatistiken

Auslastung und Analysen Ihrer Buchungen

Als PDF exportieren

Als HTML exportieren

Aktualisieren

Filter & Zeitraum wählen

Zeitraum

Alle Buchungen

75 von 75 Buchungen



75

GESAMT BUCHUNGEN



8

VERSCHIEDENE PRODUKTE



44

TOP PRODUKT AUSLEIHEN

Buchungen nach Status



Top 10 Produkte

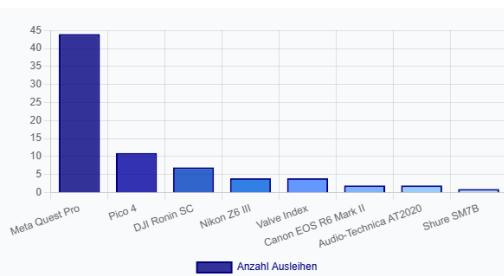


Abbildung 38: Statistiken zu allen Buchungen

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.

Für alle Services die in den Buchungsseiten verwendet werden, wurden Unit-Tests erstellt, um eine hohe Codequalität zu garantieren.

10.1.5.2 Änderungshistorie

- **Tabellen-Komponente** Verwendung einer einheitlichen Tabellenkomponente.
Begründung: Vereinheitlichung der Darstellung aller Tabellen.
Auswirkungen: Schnellere Produktpflege und konsistentes UI-Design.
- **Buchungs-Komponenten** Erstellung von Komponenten für Buchungselemente.
Begründung: Das User-Dashboard benötigt ähnliche Komponenten für die Darstellung der Buchungselemente.
Auswirkungen: Erhöhte Wiederverwendbarkeit, Wartbarkeit und einheitliches UI-Design.

10.1.6 Admin-Menü: Standorte Verwalten

Das Menü „Standorte verwalten“ bietet dem Admin die Funktion Standorte an denen Gegenstände gelagert werden hinzu zufügen, wie in Abbildung 39 dargestellt.

Standorte verwalten

Erstelle und verwalte Räume / Standorte

ID	RAUMNUMMER	ERSTELLT	AKTUALISIERT	AKTIONEN
1	F01.402 (KEIM)	23.11.2025 11:18	23.11.2025 11:18	
2	Bibliothek Flandernstraße	23.11.2025 11:18	23.11.2025 11:18	
3	VR-Labor F01.403	23.11.2025 11:18	23.11.2025 11:18	
4	Medienlabor S02.201	23.11.2025 11:18	23.11.2025 11:18	
7	Privat	01.01.2026 16:12	01.01.2026 16:12	

Abbildung 39: Standortmenü für den Admin

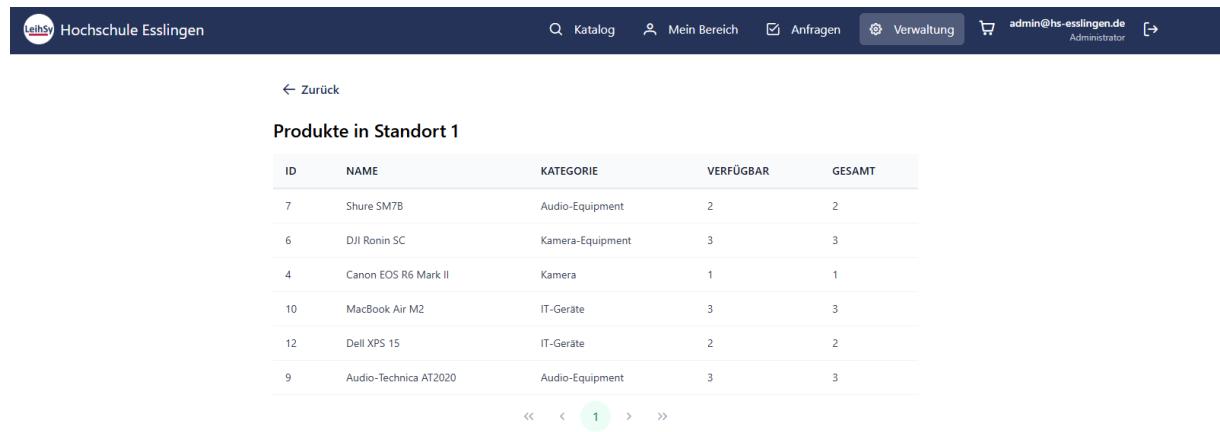
Das Menü listet alle Standorte tabellarisch auf mit folgenden Spalten:

- **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.

Beim Klicken auf einen bestimmten Standort wird die eine Detailseite geöffnet, wie in Abbildung 40 dargestellt. Auf der Seite sieht man alle Produkte, welche an diesem Standort gelagert sind.



The screenshot shows a web interface for managing inventory. At the top, there is a navigation bar with links for Catalog, My Area, Requests, Administration, and Log Out. Below the navigation bar, a heading reads "Produkte in Standort 1". A table lists six products with columns for ID, Name, Category, Available, and Total. The products are: Shure SM7B (Audio-Equipment), DJI Ronin SC (Kamera-Equipment), Canon EOS R6 Mark II (Kamera), MacBook Air M2 (IT-Geräte), Dell XPS 15 (IT-Geräte), and Audio-Technica AT2020 (Audio-Equipment). Below the table is a pagination control with buttons for navigating through multiple pages.

ID	NAME	KATEGORIE	VERFÜGBAR	GESAMT
7	Shure SM7B	Audio-Equipment	2	2
6	DJI Ronin SC	Kamera-Equipment	3	3
4	Canon EOS R6 Mark II	Kamera	1	1
10	MacBook Air M2	IT-Geräte	3	3
12	Dell XPS 15	IT-Geräte	2	2
9	Audio-Technica AT2020	Audio-Equipment	3	3

Abbildung 40: Detailansicht der Standorte und der dort gelagerten Produkte

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 tabellarisch 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

Die Kategorienverwaltung (Abbildung 41) 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.

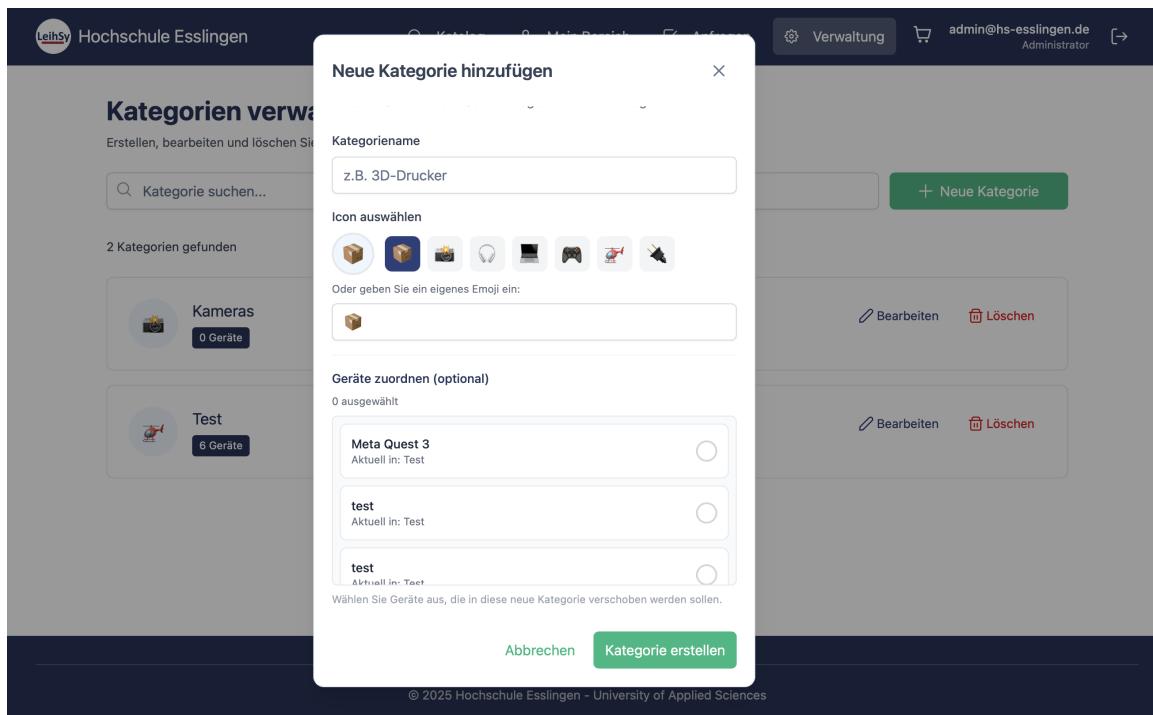


Abbildung 41: Kategorienverwaltung im Admin-Dashboard

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
- Optional beim Erstellen: Geräte einer neuen Kategorie zuordnen oder Produkte werden umgehängt
- Benutzerfeedback über Toast-Meldungen (Erfolg/Fehler)

10.1.8.2 Datenmodell

Die Kategorien sind typisiert über ein Interface mit folgendem Aufbau:

```
interface Category {
  id: string | number;
  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: '[VR]', deviceCount: 12 },  
  { id: '2', name: 'Kameras', icon: '[CAM]', deviceCount: 24 },  
  ...  
];
```

10.1.8.3 Änderungshistorie

In der ersten Version der Kategorienverwaltung war es beim Erstellen einer Kategorie noch nicht möglich, direkt Geräte auszuwählen und zuzuordnen. In der finalen Version wurde diese Funktion erweitert, sodass im Dialog zum Anlegen einer neuen Kategorie (siehe Abbildung 41) nun optional Geräte ausgewählt und sofort der neuen Kategorie zugeordnet werden können.

10.1.9 Admin-Feature: Verleiher-Zuordnung

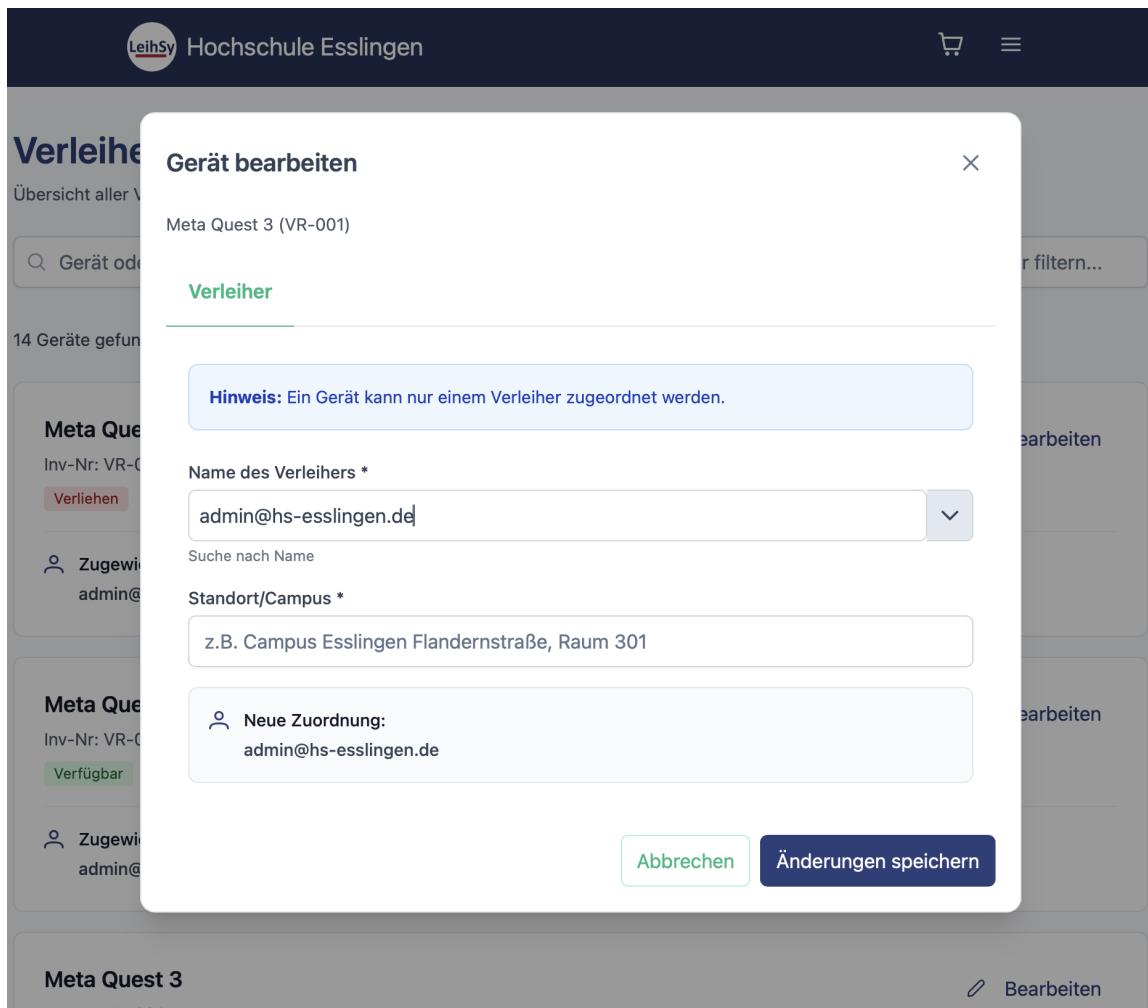


Abbildung 42: Dialog zur Verleiher-Zuordnung in der Geräteverwaltung

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 42), 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 Dashboard

Im folgenden werden die Funktionen, die sich im Verleiher-Dashboard befinden ausführlich beschrieben und erklärt, sowie durch Bilder visualisiert.

10.2.1 Verleiher-Menü

Das Menü für den Verleiher befindet sich oben rechts im Header unter dem Punkt **Anfragen**, 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.

10.2.1.1 Beschreibung des Menüs

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.

10.2.1.2 Änderungshistorie

- **Menü-Komponente** Erstellung einer Menü-Komponente zur einheitlichen Darstellung der Menüpunkte.

Begründung: Verbesserung der Übersichtlichkeit und Vereinheitlichung der Darstellung aller Verleiherfunktionen.

Auswirkungen: Schnellere Produktpflege und konsistentes UI-Design.

- **Änderung des Menü-Designs** Umstellung des Designs von quadratischen Kacheln zu flachen, schmalen Kacheln.

Begründung: Es passen mehr kleinere Kacheln auf den Bildschirm.

Auswirkungen: Schnellere Orientierung.

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

10.2.2 Verleiher-Menü: Meine Gegenstände

Das Menü "Meine Gegenstände" wie in Abbildung 43 dargestellt, zeigt eine vollständige Übersicht über sämtliche Gegenstände die dem Verleiher zugewiesen wurden. Sie dient Verleihern dazu, alle Produktinstanzen, welche ihnen zugewiesen wurden zu überblicken.

10.2.2.1 Beschreibung des Menüs

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),

Meine Gegenstände

Übersicht über alle Ihnen zugewiesenen Gegenstände

👤 Angemeldet als: admin@hs-esslingen.de (Verleiher ID: 5)

Gesamt Gegenstände 27		Verfügbar 25		Ausgeliehen 2	
<input type="text"/> Suche nach Produkt, Kategorie, Inventarnummer...					
Meta Quest Pro ⌚ VR-Equipment ⚓ VR-Labor F01.403				Verfügbar 0 / 1	
INVENTARNUMMER ↑↓	BESITZER ↑↓			STATUS ↑↓	
VR-PRO-003	Christian Haas				
Pico 4 ⌚ VR-Equipment ⚓ VR-Labor F01.403				Verfügbar 1 / 1	
INVENTARNUMMER ↑↓	BESITZER ↑↓			STATUS ↑↓	
VR-PICO-005	Hochschule Esslingen				

Abbildung 43: Verleiher-Ansicht aller zugewiesenen Items

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, siehe Abbildung 44. Auf dieser sieht die Details des Gegenstandes und dem übergeordneten Produkt. Außerdem werden alle Ausleihen des Gegenstandes tabellarisch aufgelistet mit folgenden Spalten:

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

[← Zurück zur Übersicht](#)

VR-PRO-003													
Meta Quest Pro													
Ausgeliehen													
Item Informationen Inventarnummer: VR-PRO-003 Besitzer: Christian Haas Status: Ausgeliehen	Produkt-Info Produkt ID: 1 Preis: €8.00												
Produktgruppe Produkt: Meta Quest Pro Beschreibung: High-End Mixed-Reality-Headset mit Eye-Tracking und Gesichtserkennung. Kategorie: VR-Equipment Standort: VR-Labor F01.403	Ausleihkonditionen Max. Dauer: 10 Tage Preis/Tag: €8.00 Zubehör: ["2x Touch Pro Controller", "Ladedock", "USB-C Kabel", "Transporttasche"]												
Ausleihzeiträume <table><thead><tr><th>AUSLEIHER ↑</th><th>VON ↑</th><th>BIS ↑</th><th>STATUS ↑</th></tr></thead><tbody><tr><td>admin@hs-esslingen.de</td><td>17.01.2026</td><td>18.01.2026</td><td>Bestätigt</td></tr><tr><td>admin@hs-esslingen.de</td><td>10.12.2025</td><td>15.12.2025</td><td>Ausgegeben</td></tr></tbody></table>		AUSLEIHER ↑	VON ↑	BIS ↑	STATUS ↑	admin@hs-esslingen.de	17.01.2026	18.01.2026	Bestätigt	admin@hs-esslingen.de	10.12.2025	15.12.2025	Ausgegeben
AUSLEIHER ↑	VON ↑	BIS ↑	STATUS ↑										
admin@hs-esslingen.de	17.01.2026	18.01.2026	Bestätigt										
admin@hs-esslingen.de	10.12.2025	15.12.2025	Ausgegeben										

Abbildung 44: Detailansicht des Gegenstandes für den Verleiher

10.2.2.2 Änderungshistorie

- **Tabellen-Komponente** Verwendung der selbsterstellten Tabellenkomponente.
Begründung: Einheitliche Darstellung von Daten in einer Tabelle (für das Anzeigen von Ausleihen und Gegenständen).
Auswirkungen: Schnellere Produktpflege und konsistentes UI-Design.
- **Tabellarischer Ausleihverlauf** Hinzufügen aller vergangenen Ausleihen für den jeweiligen Gegenstand in Tabellenform.
Begründung: Einfachere Nachvollziehbarkeit der Gegenstandshistorie.
Auswirkungen: Besserer Überblick für den Verleiher.

10.2.3 Verleiher-Menü: Ausleihe Übersicht

Die Ausleihe-Übersicht ist dafür da, dass der Verleiher jederzeit einen schnellen Überblick über den aktuellen Stand der Geräte hat, siehe Abbildung 45. Sie zeigt, welche Geräte

gerade aktiv ausgeliehen sind und welche Ausleihen zwar bestätigt wurden, aber noch als Abholung ausstehen.

Status	Student	Gerät	Inventar-Nr.	Ausgeliehen	Rückgabe fällig	Campus
2 Tag(e) überfällig	leihsy.lender@hs-esslingen.de 1	HTC Vive Pro 2	VR-011	10.01.2026	18.01.2026	VR-Labor F01.403
Aktiv	leihsy.lender@hs-esslingen.de 1	MacBook Pro 14 Zoll M3	IT-001	18.01.2026	22.01.2026	F01.402 (KEIM)
Aktiv	leihsy.user@hs-esslingen.de 4	Canon EOS R5	CAM-011	19.01.2026	22.01.2026	Bibliothek Flandernstrasse

Abbildung 45: Ausleihe-Übersicht im Verleiher-Menü

10.2.3.1 Funktionen

- Anzeige von aktive Ausleihen, überfällige Ausleihen, heute fällige Ausleihen, ausstehende Abholungen)
- Suche nach Student, Gerät oder Inventarnummer
- Filter nach Standort/Campus
- Sortierung nach Datum)
- Tab-Wechsel zwischen „Aktive Ausleihen“ und „Ausstehende Abholungen“
- Anzeige wichtiger Informationen pro Eintrag (Student, Gerät, Inventar-Nr., Ausleih-/Rückgabedatum bzw. Abholtermin, Campus)
- Hervorhebung von kritischen Fällen (z. B. überfällige Rückgaben)
- Aktion „Ausgabe“ für bestätigte Abholungen

10.2.3.2 Technische Umsetzung

Die Daten werden im Frontend über zwei getrennte Modelle abgebildet: `ActiveLoan` für aktive Ausleihen (inkl. Rückgabedatum, Status `active`/`overdue` und optionalen Überfälligkeitstagen) sowie `PendingPickup` für bestätigte, aber noch nicht ausgegebene Abholungen (inkl. Abholdatum/-zeit und Bestätigungsdatum). Die Kennzahlen in den Status-Kacheln werden über Getter wie `activeCount`, `overdueCount`, `dueTodayCount` und `pendingCount` berechnet.

Die Anzeige in den Tabellen basiert auf den beiden berechneten Listen `filteredAndSortedLoans` und `filteredAndSortedPickups`. Dort werden zunächst Suchbegriff und Campus-Filter angewendet und anschließend entsprechend der gewählten Sortierung (Datum, Student, Gerät) sortiert. Bei der Datumssortierung werden überfällige Ausleihen priorisiert, sodass kritische Fälle oben erscheinen.

10.2.4 Verleiher-Menü: Anfragen

Die Funktion *Anfragen* dient als zentrale Übersicht für alle eingehenden Ausleihanfragen von Studierenden. Verleiher können hier nachvollziehen, welche Anfragen neu eingegangen sind, welche bereits bearbeitet wurden und für welche Geräte eine Ausleihe angefragt wurde. Ziel der Ansicht ist es, den Bearbeitungs- prozess von Anfragen übersichtlich darzustellen und eine schnelle Entscheidung über Annahme oder Ablehnung zu ermöglichen.

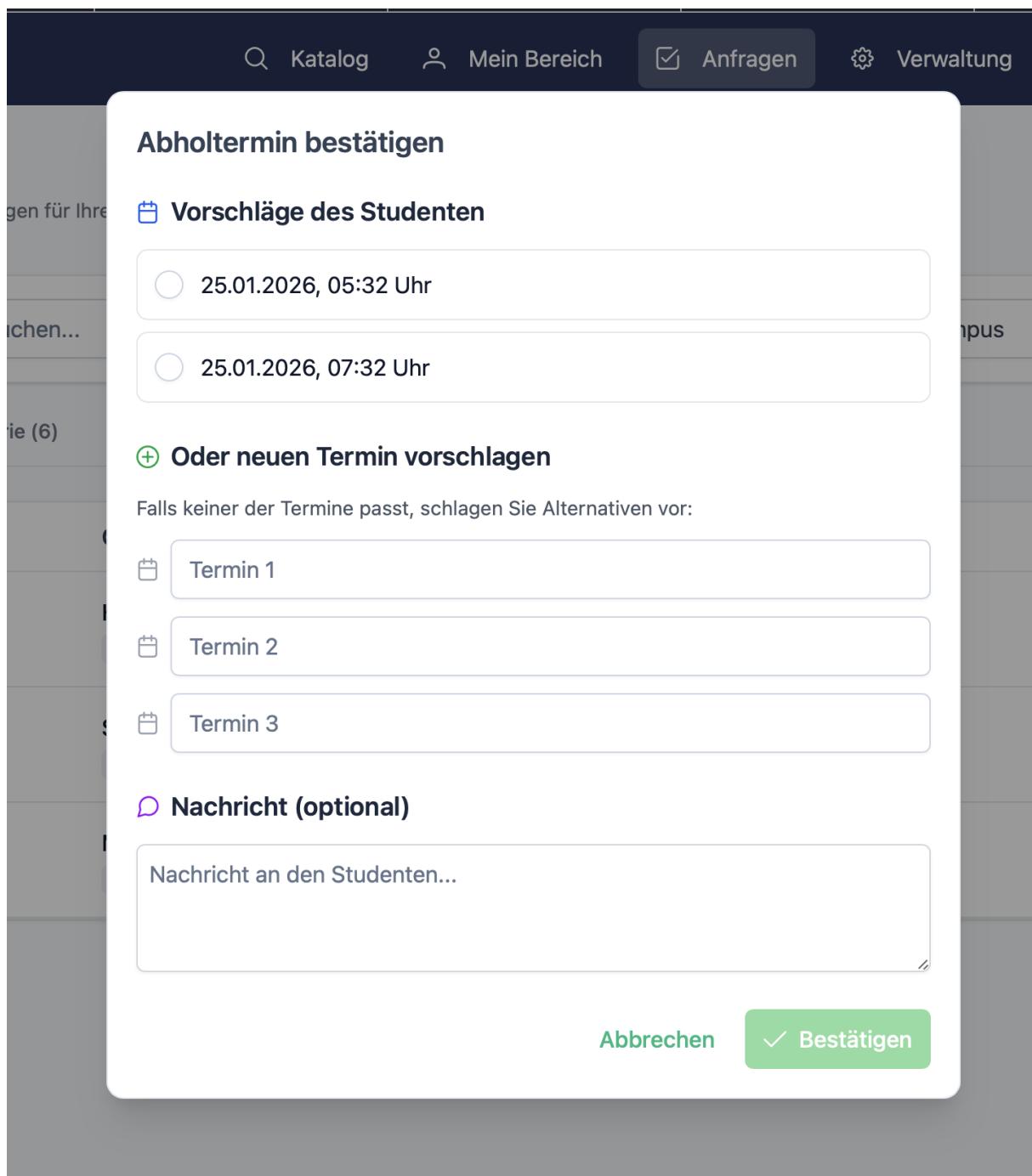


Abbildung 46: Anfragen-Übersicht

10.2.4.1 Funktionen

- Anzeige aller offenen und bereits bearbeiteten Ausleihanfragen
- Aktionen zur Annahme oder Ablehnung einer Anfrage
- Im Falle einer Ablehnung ist eine Begründung notwendig
- Direkte Kommunikation an den Verleiher/Studenten

- Termin Vorschläge
- Über das Tooltip die Begründung einsehen

10.2.4.2 Abholtermin-Auswahl (Pickup Selection Dialog)

Für das Annehmen einer Anfrage wird eine eigene Unterfunktion verwendet, die den `PickupSelectionDialogComponent` einbindet. Dieser Dialog wird über `openAcceptDialog()` geöffnet und ermöglicht es dem Verleiher, entweder einen vom Studenten vorgeschlagenen Abholtermin auszuwählen oder alternativ eigene Termine vorzuschlagen, wie in Abbildung 46. Die Auswahl wird anschließend über Events an die Hauptkomponente zurückgegeben: Bei der Bestätigung eines bestehenden Termins erfolgt die Verarbeitung über `onPickupSelected(...)`, bei Gegenvorschlägen über `onNewPickupsProposed(...)`. Dadurch ist die Abholtermin-Logik klar vom restlichen Anfragen-Workflow getrennt und übersichtlich gekapselt.

10.2.4.3 Technische Umsetzung

Beim Laden werden Standorte für den Filter aufgebaut (`loadLocations()`) und anschließend die Anfragen des aktuell eingeloggten Verleiher geladen (`loadData()`). Offene und erledigte Einträge werden über `filteredPending()` bzw. `filteredDone()` getrennt, wobei beide Listen die gemeinsame Filter- und Sortierlogik `applyAll()` verwenden. Das Annehmen startet den Abholtermin-Flow: `openAcceptDialog()` öffnet den Termin-Dialog und die Rückgabe erfolgt über `onPickupSelected(...)` (Termin bestätigen) oder `onNewPickupsProposed(...)` (Gegenvorschläge). Das Ablehnen wird über `confirmDecline()` gespeichert; nach Aktionen wird die Liste neu geladen und es erscheinen Toast-Meldungen.

10.2.5 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.

10.2.5.1 Beschreibung des Menüs

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

10.2.5.2 Änderungshistorie

- **Verschiebung der Funktionalität** Die Funktionalität wurde aus dem Admin-Dashboard und User-Dashboard ins Lender-Dashboard verschoben.
Begründung: Verleiher sind autorisierte Personen und können somit auch ihre eigenen Gegenstände verleihen.
Auswirkungen: Reduktion der Komplexität der privaten Gegenstände.
- **Entfernen der Funktionalität aus Nutzer- und Admin-Menü** Die Funktionalität wurde gänzlich aus dem User-Dashboard und Admin-Dashboard entfernt.
Begründung: Wird aufgrund des Verschiebens nicht mehr benötigt.
Auswirkungen: Funktionalität ausschließlich über Verleiher-Dashboard zugänglich.
- **Änderung der Logik für das Anlegen privater Gegenstände** Die Logik, private Gegenstände mit einem JSON-Array anzulegen, wurde entfernt und durch die Logik aus dem Admin-Dashboard zum Anlegen der Gegenstände und Produkte ersetzt.
Begründung: Logik war zu aufwendig und durch das Verschieben überflüssig.
Auswirkungen: Einfachere Erstellung von privaten Gegenständen und Produkten.

10.2.6 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.

10.2.7 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 storniert. 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 Dashboard

Im folgenden werden die Funktionen, die sich im Nutzer-Dashboard befinden ausführlich beschrieben und erklärt, sowie durch Bilder visualisiert.

10.3.1 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.

10.3.1.1 Beschreibung des Menüs

Die Menüpunkte sind Listenartig angeordnet und jeder Menüpunkt enthält:

- Icon
- Titel,
- Beschreibung des Menüs.

10.3.1.2 Änderungshistorie

- **Menü-Komponente** Erstellung einer Menü-Komponente zur einheitlichen Darstellung der Menüpunkte.

Begründung: Verbesserung der Übersichtlichkeit und Vereinheitlichung der Darstellung aller Nutzerfunktionen.

Auswirkungen: Schnellere Produktpflege und konsistentes UI-Design.

- **Änderung des Menü-Designs** Umstellung des Designs von quadratischen Kacheln zu flachen, schmalen Kacheln.

Begründung: Es passen mehr kleinere Kacheln auf den Bildschirm.

Auswirkungen: Schnellere Orientierung.

- **Änderung des Menü-Layouts** Anpassung des Layouts von zwei quadratischen Menükacheln pro Zeile zu einer schmalen Menükachel pro Zeile.

Begründung: Klarere Anordnung mit mehr Kacheln auf dem Bildschirm.

Auswirkungen: Bessere Anordnung und Orientierung.

10.3.2 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 Ausleihers.
- **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, vgl. Abbildung 47, 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

[← Zurück zur Übersicht](#)[Export PDF](#)[QR-Codes anzeigen](#)

The screenshot displays a booking summary for a Meta Quest Pro VR headset (Inventarnummer: VR-PRO-003). At the top right, there are buttons for "Export PDF" and "QR-Codes anzeigen". The booking ID is #76.

Ausleihzeitraum:
Geplante Abholung: 10.12.2025
Geplante Rückgabe: 15.12.2025

Verleiher:
Name: admin@hs-esslingen.de
Verleiher-ID: #5

Gegenstand:
Produkt: Meta Quest Pro
Inventarnummer: VR-PRO-003
Produkt-ID: #1

BUCHUNGSVERLAUF:

- Ausgegeben:** Gegenstand abgeholt am 16.01.2026 um 17:07
- Bestätigt:** Von admin@hs-esslingen.de bestätigt am 16.01.2026 um 17:07
- Buchung erstellt:** Buchungsanfrage von admin@hs-esslingen.de am 16.01.2026 um 17:04

Nachricht / Kommunikationsverlauf:

- + Test
- Test Test

Abbildung 47: Buchungsansicht aus Sicht des Users

Der Nutzer kann außerdem die Buchung stornieren. Falls die Buchung bestätigt wurde, kann der Nutzer 3 Termine mit Uhrzeiten vorschlagen, an welchen dieser 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. Genaue 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.

10.3.3 Nutzer-Menü: Mein Gruppen

Die Seite "Meine Gruppen" erlaubt es dem Nutzer Gruppen für Studentenprojekte zu erstellen, bearbeiten und zu löschen.

10.3.3.1 Beschreibung des Menüs

Die Seite wird in Abbildung 48 dargestellt.

The screenshot shows a user interface for managing groups. At the top, there is a back button labeled "← Zurück zur Übersicht". Below it, the title "Projekt SWTM" is displayed, along with three buttons: "QR-Code anzeigen" (QR code), "Bearbeiten" (Edit), and "Löschen" (Delete). The main content area is divided into two sections: "Gruppeninformationen" and "Mitglieder".

Gruppeninformationen:

Beschreibung	Budget
Projektgruppe für das zentrale Softwareprojekt im vierten Semester	200 €
Mitglieder	Erstellt am
3	2026-01-06T11:57:48.166454

Mitglieder:

Information: Jedes Gruppenmitglied kann den QR-Code anzeigen und teilen. Neue Mitglieder treten der Gruppe bei, indem sie den QR-Code scannen.

USER ID	NAME	EIGENTÜMER	AKTIONEN
5	admin@hs-esslingen.de	Ja	
7	leihsy.user@hs-esslingen.de	-	
6	leihsy.lender@hs-esslingen.de	-	

Pagination: << < 1 > >>

Abbildung 48: Ansicht des Nutzers im Gruppenmenü

Im Menü werden die eigenen Gruppen tabellarische 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

10.3.3.2 Änderungshistorie

- **Hinzufügen der Erstellungsfunktion für Gruppen** Gruppen können nun auch im Nutzer-Menü erstellt werden, nicht mehr nur im Admin-Menü.
Begründung: Nutzer sollen eigene Projektgruppen erstellen und nicht vom Admin abhängig sein.
Auswirkungen: Reduktion der Abhängigkeit vom Admin; schnellere Möglichkeit, Gruppen zu erstellen.
- **QR-Code für den Gruppenbeitritt** Hinzufügen eines QR-Codes, mit dem man durch Scannen der Gruppe beitreten kann.
Begründung: Gruppeneigentümer muss die Mitglieder nicht manuell hinzufügen.
Auswirkungen: Reduktion der Komplexität und des Aufwandes.

10.4 E-Mails für den Ausleihprozess

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) gefüllt 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

Um den Ausleihprozess effizienter zu gestalten und die Anzahl einzelner Buchungsanfragen zu reduzieren, wurde das System um eine Warenkorb-Funktionalität erweitert. Dieses Feature ermöglicht es Benutzern, mehrere Gegenstände temporär zu sammeln und anschließend als gebündelte Sammelbestellung in einem einzigen Vorgang anzufragen.

10.5.1 Erste Version des Warenkorbs

Der Warenkorb stellt eine Benutzeroberfläche bereit um gesammelte gewünschte Ausleihanfragen abzusenden, vgl. Abbildung 49. 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 **Zum Warenkorb hinzufügen** kann das Produkt zum Warenkorb hinzugefügt werden.

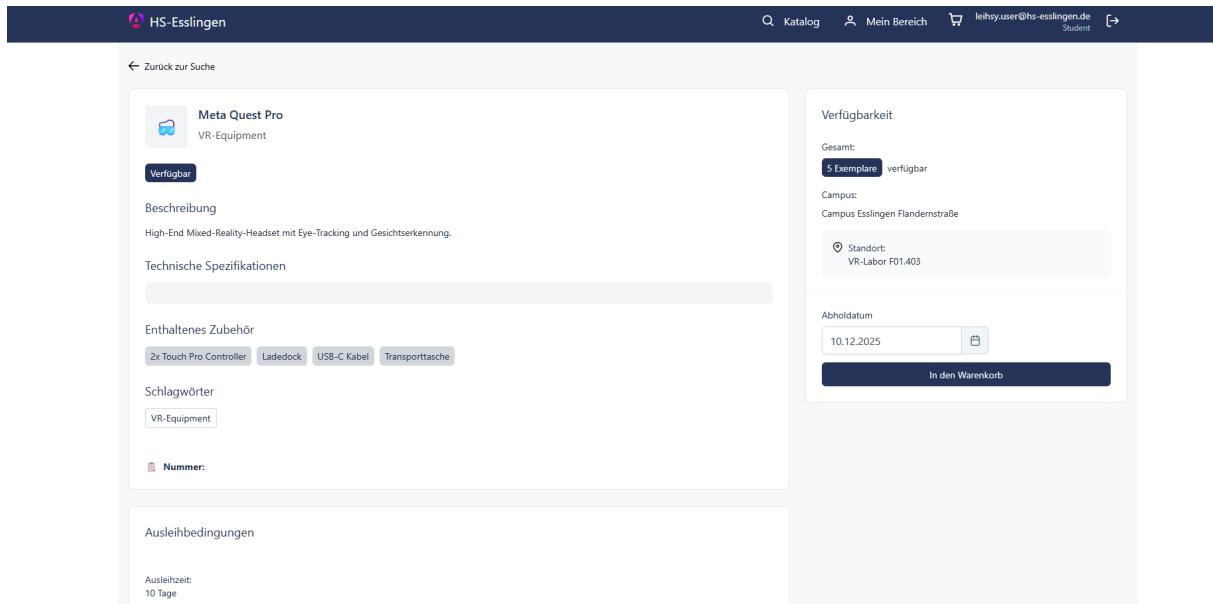


Abbildung 49: Produkt-Detailseite

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 **Zum Warenkorb hinzufügen** wird grün und ändert den Text in **Zum Warenkorb hinzugefügt**, vgl. Abbildung 50..

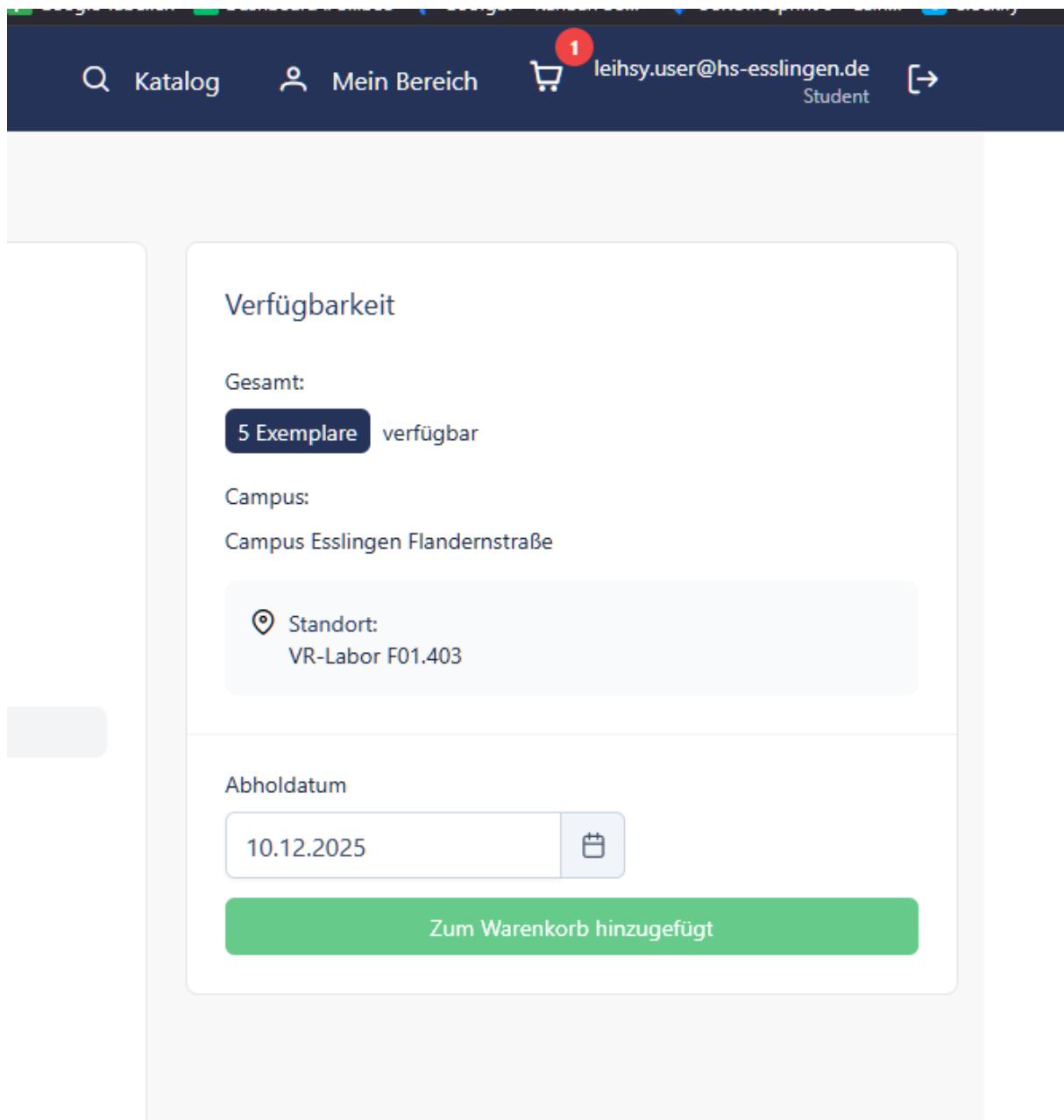


Abbildung 50: Zum Warenkorb Hinzugefügt

Aus technischer Sicht wird bei dem Click auf **Zum 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, vgl. Abbildung 51.

The screenshot shows a user interface for a shopping cart. At the top, there's a dark header bar with the HS-Esslingen logo, a search bar, a 'Mein Bereich' link, and a user icon. Below the header, the main content area has a title 'Warenkorb' with a count of 1. A product item 'Meta Quest Pro' is listed, showing its inventory number (1), category, and a red trash can icon in the top right corner. Below the product details, there are sections for 'Abholort' (Collection point) and 'Abholdatum' (Collection date, set to 10.12.2025). A 'Ausleihbedingungen' (Loan conditions) section includes a note about a 10-day loan period and availability extensions. At the bottom, a summary box shows 'Gesamtanzahl: 1 Produkt(e)' and contains a checkbox for accepting loan terms, followed by 'Weiter suchen' and 'Auslehanfrage abschicken' buttons.

Abbildung 51: 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 unbeschädigt, vollständig und rechtzeitig zurückzugeben. Nur wenn diese Zusicherung erteilt wurde kann der Button zum Absenden der Auslehanfrage(n) betätigt werden, vgl. Abbildung 52.

This screenshot shows the checkout step of the shopping cart process. It displays the same summary information as Abbildung 51: 'Gesamtanzahl: 1 Produkt(e)'. Below the summary is a checked checkbox for accepting loan terms. At the bottom, there are two buttons: 'Weiter suchen' and a dark blue 'Auslehanfrage abschicken' button.

Abbildung 52: 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.

10.5.2 Verbesserungen des Warenkorbs

In den Verbesserungen des Warenkorbs wurden Fehler der ersten Version ausgebessert und Eingabemöglichkeiten und die Behandlung von falschen Eingaben hinzugefügt.

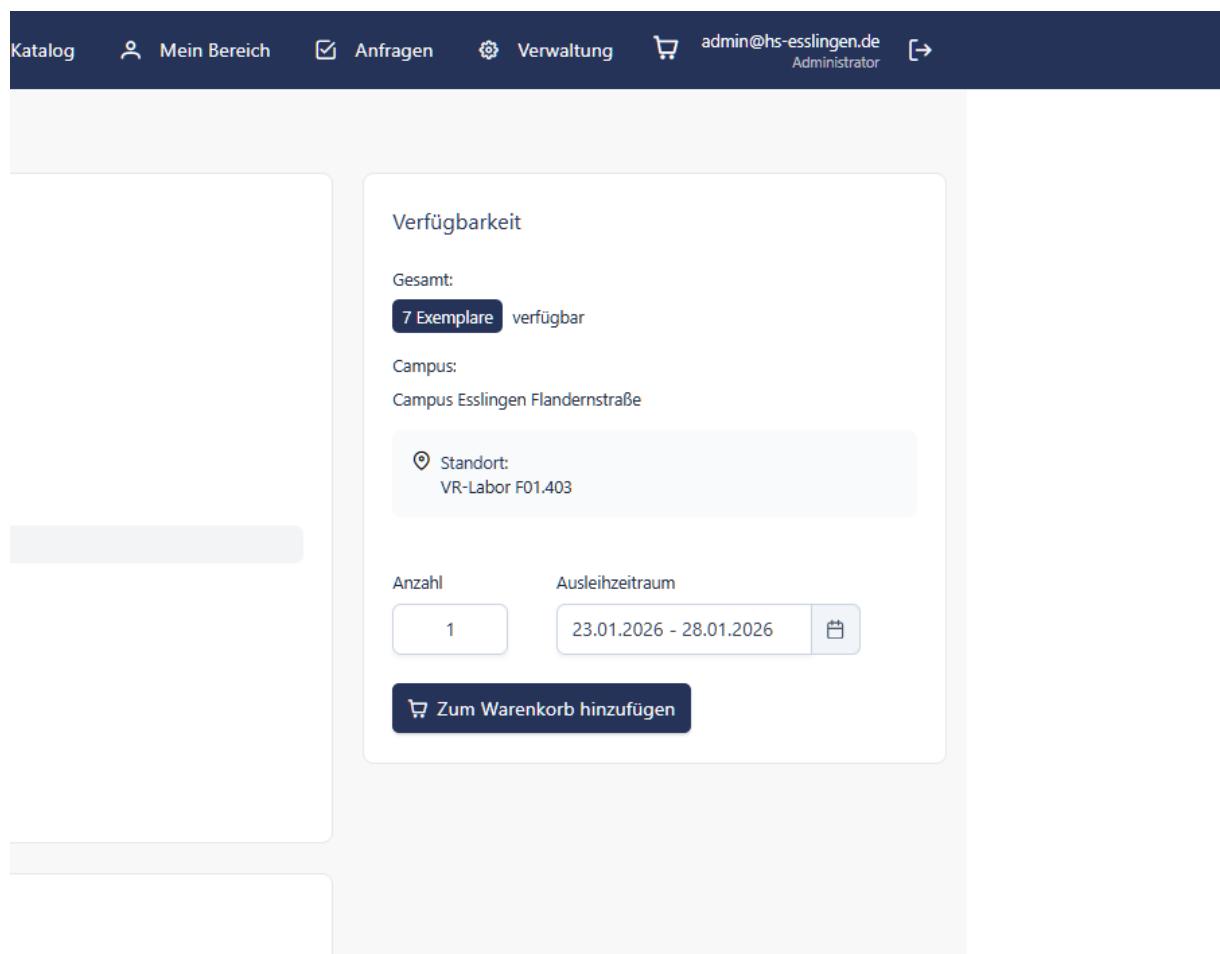


Abbildung 53: Add To Cart Neu

Die Abbildung 53 zeigt die Änderungen in den Eingabefeldern auf der Device-Details Page wo man das Produkt zum Warenkorb hinzufügt:

- Der DatePicker für das Abholdatum wurde durch einen DatePicker für den Ausleihzeitraum ersetzt
- Es wurde ein Zahleneingabefeld für die Anzahl an Items hinzugefügt

- Der Button wurde durch Verwendung der im Projekt definierten Component vereinheitlicht

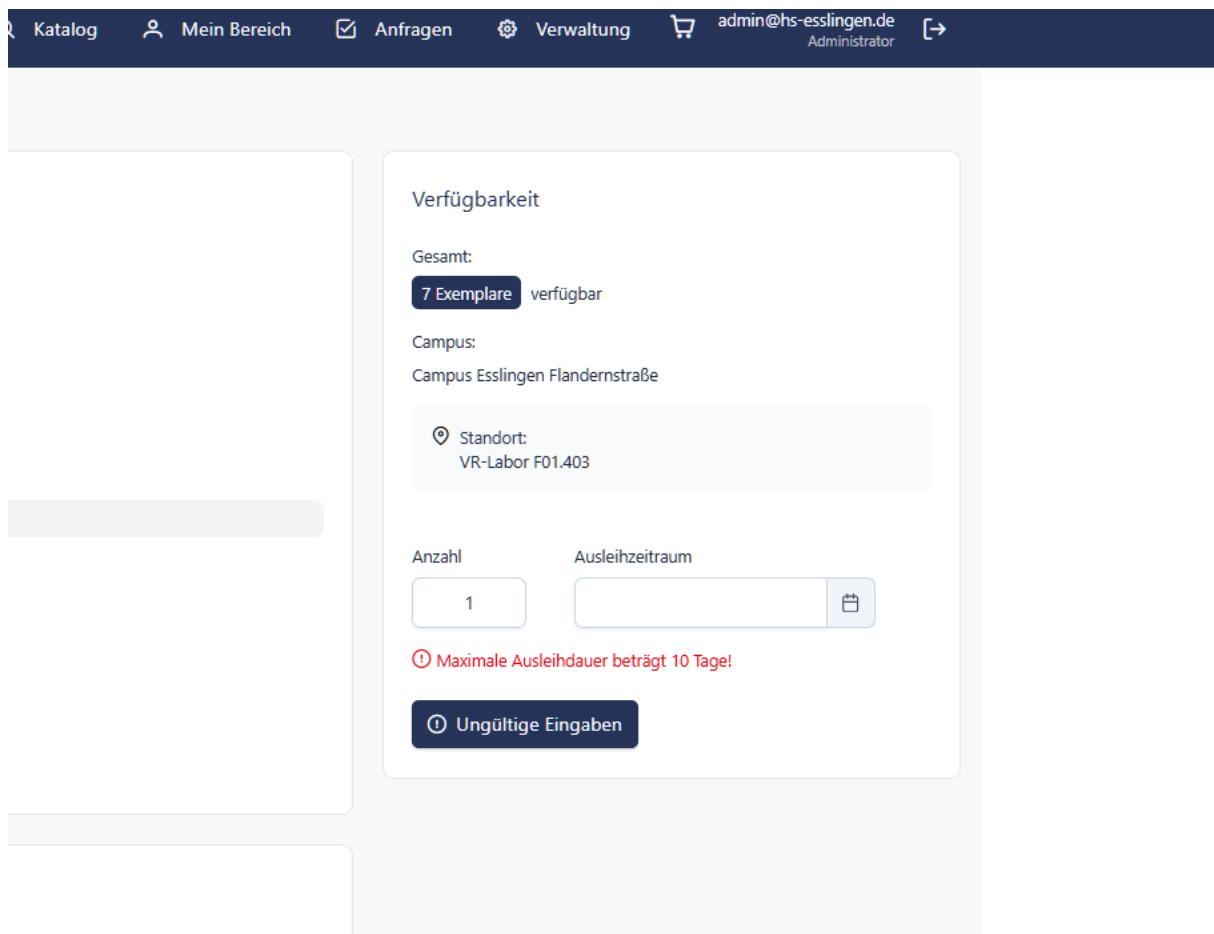


Abbildung 54: Add To Cart Eingabefehler

Die Abbildung 54 zeigt eine der neuen Fehlermeldungen für falsche Eingaben. Darunter zählen ein Ausleihzeitraum der die maximale Ausleihdauer überschreitet und ein Ausleihzeitraum der Tage enthält an denen nicht genug Items dieses Produkts verfügbar sind. Hierbei wird bei jeder Änderung der ausgewählten Anzahl ein API-Call an den Endpunkt `/api/products/periods` mit dem Parameter `unavailable` als type durchgeführt. Die API Antwort enthält alle Zeiträume in denen die ausgewählte Anzahl nicht verfügbar ist. Diese Zeiträume werden dann in genaue Tage umgerechnet, die im DatePicker ausgegraut angezeigt werden. Bei jeder Eingabe des Ausleihzeitraums wird dann überprüft ob dieser nicht verfügbare Tage enthält. Falls dies der Fall ist wird der DatePicker zurückgesetzt und die Fehlermeldung angezeigt. Der Button zum Hinzufügen des Items zum Warenkorb kann nur betätigt werden wenn die Eingaben in den Auswahlfeldern valide sind. Ansonsten zeigt er den Text Ungültige Eingaben an und ist nicht clickbar.

🛒 Warenkorb 1

The screenshot shows a shopping cart interface for a product named "Meta Quest Pro".
Product details:
- Inventarnummer 1
- Kategorie
- Abholort: Campus Esslingen Flandernstraße VR-Labor F01.403
- Anzahl: 1
- Ausleihzeitraum: 17.01.2026 - 22.01.2026
- Nachricht (optional): Nachricht zur Ausleihe (z. B. Verwendungszweck)
Lending conditions:
- Ausleihfrist: 10 Tage
- Verlängerungen: Nach Verfügbarkeit
Summary:
- Gesamtanzahl: 1 Produkt(e)
- Ich akzeptiere die Ausleihbedingungen und verpflichte mich, den/die Artikel unbeschädigt, vollständig und rechtzeitig zurückzugeben.
Buttons:
- Weiter suchen
- Ausleihanfrage abschicken

Abbildung 55: Verbesserter Warenkorb

Die Abbildung 55 zeigt den verbesserten Warenkorb. Das Layout der Auswahlfelder wurde verbessert. Es wurde ein Eingabefeld für die Nachricht an den Verleiher hinzugefügt. Das Nachladen der Produktinformationen wurde in den CartService verlagert, da eine Anreicherung der Daten aus dem ProduktService Probleme bei Änderung der Cart Items nach sich gezogen hatte. Nun werden bei jedem Laden des CartServices sowie bei jedem Update des Carts die Produktdaten für jedes Cart Item neu vom Backend geladen um zu gewährleisten dass die angezeigten Daten immer aktuell sind. Die Daten werden dann im lokalen Browserstorage gespeichert um zu verhindern dass beispielsweise bei einem Neuladen der Seite erst Informationen angezeigt werden, wenn diese neu vom Backend geholt wurden.

Auch im Warenkorb wurde ein Handling für fehlerhaften Input hinzugefügt. Dieses wird ebenfalls komplett im CartService durchgeführt. Wird die gewünschte Anzahl eines Produkts im Warenkorb erhöht, werden die nicht verfügbaren Zeiträume des Produkts vom

CartService vom Backend geholt und das zugehörige Attribut des betreffenden CartItems aktualisiert. Außerdem wird der ausgewählte Ausleihzeitraum zurückgesetzt. Wird der ausgewählte Ausleihzeitraum geändert, wird im CartService überprüft ob dieser gültig ist. Hierbei wird gegen die maximale Ausleihdauer und die nicht verfügbaren Zeiträume für die ausgewählte Anzahl überprüft. Ist der Zeitraum nicht valide, wird das Auswahlfeld zurückgesetzt.

Werden die Daten in einem der Eingabefelder gültig geändert, werden die Änderungen sofort im CartService und im lokalen Browserstorage gespeichert. Dadurch bleiben die Eingaben auch über ein Navigieren auf eine andere Seite, ein Neuladen der ganzen Website oder einen Neustart des Browsers hinweg gespeichert. Dies sorgt für ein verbessertes Nutzererlebnis da nicht andauernd neue Eingaben getätigt werden müssen.

The screenshot shows a user interface for a shopping cart. At the top, there's a header with a shopping cart icon and the text "Warenkorb 1". Below it, a product item is listed: "Meta Quest Pro" with "Inventarnummer 1" and a "Kategorie" link. The main form area contains the following fields:

- Abholort:** Campus Esslingen Flandernstraße VR-Labor F01.403
- Anzahl:** 1
- Ausleihzeitraum:** (date range input field)
- Maximale Ausleihdauer überschritten** (red error message)
- Nachricht (optional):** Nachricht zur Ausleihe (z. B. Verwendungszweck) (text input field)
- Ausleihbedingungen:** (grey box containing terms and conditions)
- Gesamtanzahl:** 1 Produkt(e)
- Ich akzeptiere die Ausleihbedingungen und verpflichte mich, den/die Artikel unbeschädigt, vollständig und rechtzeitig zurückzugeben.** (checkbox)

At the bottom, there are two buttons: "Weiter suchen" and "Ausleihanfrage abschicken".

Abbildung 56: Warenkorb Eingabefehler

Werden die Daten in einem der Eingabefelder ungültig geändert wird für das betreffende

Produkt im Warenkorb eine Warnnachricht angezeigt die den Eingabefehler beschreibt. Ein Beispiel ist in der Abbildung 56 zu sehen. Dazu zählen ein Ausleihzeitraum der die maximale Ausleihdauer überschreitet und ein Ausleihzeitraum der nicht verfügbare Tage für die gewünschte Anzahl des Produkts enthält. Auch für ein Ändern der gewünschten Anzahl wird eine Warnnachricht angezeigt um das Zurücksetzen des Ausleihzeitraums zu erklären.

Bei einem Neuladen der Seite wird von einem Neustart des Browsers mit gegebenenfalls größerem zeitlichem Unterschied ausgegangen. Daher wird der ausgewählte Ausleihzeitraum überprüft und zurückgesetzt, falls sich dieser nicht vollständig in der Zukunft befindet. Auch hierbei wird eine Fehlermeldung angezeigt.

Die Ausleihanfrage kann erst abgesendet werden wenn die Anzahl und der Ausleihzeitraum gültig gesetzt sind. Die Nachricht an den Verleiher ist optional. Außerdem müssen weiterhin die Ausleihbedingungen akzeptiert werden und es muss versichert werden den oder die Artikel unbeschädigt und rechtzeitig zurückzugeben.



Abbildung 57: Warenkorb Buchungen erfolgreich erstellt

Beim Absenden der Ausleihanfrage wird für jede Position im Warenkorb eine Buchung mit der jeweiligen Product ID, der gewünschten Quantität und dem gewünschten Ausleihzeitraum angelegt. Nur wenn alle Buchungen gültig erstellt wurden wird eine Erfolgsmeldung angezeigt und der Warenkorb geleert, wie in Abbildung 57 zu sehen.

10.5.3 Backend-Änderungen für Warenkorb

Die Einführung des Warenkorbs erforderte signifikante Anpassungen an der serverseitigen Architektur. In diesem Abschnitt wird die notwendigen Erweiterungen im Backend dokumentiert, um die temporäre Sammlung von Artikeln zu verarbeiten und diese anschließend in einen konsistenten Buchungsvorgang zu überführen.

10.5.3.1 Änderung des Booking Endpoints

Der GET-Endpoint für `/api/bookings` wurde geändert, da die vorherige Implementierung als Parameter die genaue Item ID des Items benötigte das ausgeliehen werden sollte. Dies hätte eine Clientseitige Überprüfung und Auswahl des genauen Items erfordert das

ausgeliehen werden soll. Mit der neuen Version muss nur noch eine Product ID angegeben werden. Außerdem kann eine Quantität angegeben werden. Die Backend-Logik erstellt dann automatisch die Buchungen für die benötigte Menge freier Items dieses Produkts falls verfügbar. Die neue Logik orientiert sich somit mehr an der trivialen Herangehensweise Ich hätte gerne diese Anzahl von diesem Produkt in diesem Zeitraum.

Technische Umsetzung: Die Parameter im ProductController wurden geändert: itemID wurde durch productID ersetzt und der neue Parameter quantity wurde hinzugefügt. Der Rückgabetyp wurde in eine Liste von BookingDTOs geändert. Die createBooking Methode im BookingService wurde geändert. Sie erhält nun als Parameter ebenfalls eine productID und eine quantity. Es wird zuerst nach freien Items des Produkts im angegebenen Zeitraum gesucht. Dabei werden alle Items eines Produkts durchsucht bis genug Items gefunden sind die im angegebenen Zeitraum verfügbar sind. Für diese Items wird dann jeweils einzeln im BookingRepository ein Booking erstellt. Somit wird auch berücksichtigt dass unter Umständen verschiedene Verleiher für die Items zuständig sind.

10.5.3.2 Hinzufügen eines GET-Endpoints im ProductController

Um im Frontend anzuzeigen, wann ein Gegenstand verfügbar ist, muss diese Information vom Backend bereitgestellt werden. Hierzu wurde ein neuer Endpoint unter /api/products/productId angelegt der als Parameter die Product ID, die gewünschte Anzahl und den Typ (verfügbar / nicht verfügbar) erhält und eine Liste von Zeitperioden zurückgibt die entweder die verfügbaren oder die nicht verfügbaren Zeiträume abbildet. Im ProductService wurden dementsprechend zwei ähnlich aufgebaute Funktionen hinzugefügt. Sie haben folgende Funktionsweise: Es werden alle Items des Produkts geladen. Dann werden für jedes Item alle Bookings geladen die nicht vollständig in der Vergangenheit liegen. Für diese Bookings werden dann BookingEvents in einer Liste erstellt die jeweils die Änderung in den ausgeliehenen Items (+1 für beginnende Ausleihe, -1 für endende Ausleihe) und das zugehörige Datum enthalten. Diese Liste wird chronologisch sortiert und funktioniert wie ein Zeitstrahl auf dem die BookingEvents chronologisch sortiert sind. Wenn die nicht verfügbaren Zeiträume abgefragt werden ist nun der Punkt gekommen an dem für jedes Booking Event auf dem Zeitstrahl überprüft wird ob sich der verfügbare Bestand signifikant ändert so dass er unter der benötigten Anzahl an Items fällt oder wieder darüber liegt. Gibt es eine signifikante Änderung wird diese als Start beziehungsweise Ende einer Zeitperiode gesetzt in der das Produkt für die gewünschte Anzahl nicht verfügbar ist. Endprodukt und Rückgabewert ist eine Liste mit Zeiträumen in denen das gewünschte Produkt für die gewünschte Anzahl nicht verfügbar ist. Analog mit kleinen Anpassungen funktioniert die Variante in der die verfügbaren Zeiträume abgefragt werden.

10.6 Backend-Implementierung der Buchungsverwaltung

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.

10.6.1 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).

10.6.2 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. In der Tabelle 5 sind die verschiedenen Möglichkeiten zu sehen.

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

10.6.3 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 gespeicher-

ten 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 6.2.2) 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.

10.6.4 Zusammenfassung

In der Tabelle 6 sind alle Änderungen des Booking-Systems zu sehen

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

10.7 Authentifizierung und Benutzerverwaltung

Ein sicherer Zugang und eine klare Rechteverwaltung sind die Grundvoraussetzung für den Betrieb von LeihSy. In diesem Abschnitt wird die Umsetzung der Authentifizierung demonstriert und gezeigt, wie die unterschiedlichen Benutzerrollen im System verwaltet und voneinander abgegrenzt werden.

10.7.1 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.

10.7.1.1 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.

10.7.1.2 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

10.7.2 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.

10.7.2.1 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`)

10.7.2.2 Autorisierungsregeln

In der Tabelle 7 ist zu sehen welche URL-Muster spezifische Berechtigungen erhalten

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

Tabelle 7: Zugriffskontrolle nach Endpoint-Mustern

10.7.3 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.

10.7.4 Frontend-Integration

Die technische Anbindung des Frontends an den Identitätsanbieter ist entscheidend für eine nahtlose User Experience. In diesem Abschnitt wird beschrieben, wie die Webanwendung den Authentifizierungsstatus verwaltet wird, Tokens sicher gehandhabt werden und die Benutzeroberfläche dynamisch an die Berechtigungen des angemeldeten Nutzers angepasst werden.

10.7.4.1 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

10.7.4.2 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.

10.7.4.3 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.

10.7.4.4 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.

10.7.4.5 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

10.7.4.6 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

10.7.4.7 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.

10.7.5 Automatische Benutzersynchronisation

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.

10.7.5.1 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.

10.7.5.2 Lösungsansatz

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

10.7.5.3 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.

10.7.5.4 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.

10.7.5.5 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.

10.7.5.6 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.

10.7.5.7 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 10.7.3 dokumentiert.

10.7.5.8 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.

10.8 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.

10.8.1 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

10.8.2 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.

10.8.3 Backend-Implementierung

In diesem Abschnitt wird die Implementierung der Geschäftslogik innerhalb der Spring Boot Architektur erläutert sowie das Zusammenspiel von REST-Controllern, Services und der Datenbank.

10.8.3.1 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.

10.8.3.2 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 `FileStorageException` geworfen.

10.8.3.3 Dateinamenskonvention

Ursprünglich wurden UUIDs als Dateinamen verwendet (z.B. `550e8400-e29b-41d4-a716-446655440000`). 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.

10.8.3.4 ImageController

Der REST-Controller stellt die drei Endpunkte der Tabelle 8 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.

10.8.3.5 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`

10.8.4 Frontend-Integration

In diesem Abschnitt wird gezeigt, wie die Benutzeroberfläche die bereitgestellten API-Endpunkte konsumiert und die Daten für eine interaktive Nutzung visuell aufbereitet.

10.8.4.1 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.

10.8.4.2 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.

10.8.4.3 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).

10.8.5 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.

10.8.6 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.

10.9 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.

10.9.1 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.

10.9.2 Implementierung

Es wurde ein Spring Scheduler implementiert, der periodisch die Datenbank prüft und veraltete Buchungen per Soft-Delete storniert.

10.9.2.1 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)

10.9.2.2 Scheduler-Komponente

Die Klasse `BookingScheduler` im Package `scheduler` enthält zwei zeitgesteuerte Methoden welche in Tabelle 9 zu sehen sind:

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

Methode	Ausführung	Funktion
autoCancelPendingBookings	Jede volle Stunde	Storniert PENDING-Buchungen ohne Termin
autoExpireConfirmedBookings	Jede halbe Stunde	Markiert nicht abgeholtene Buchungen als EXPIRED

Tabelle 9: Scheduler-Methoden

10.9.3 Zusammenfassung

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

10.10 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.

10.10.1 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

10.10.2 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.

Die Entscheidung fiel auf Option B, da diese Architektur flexibler ist und das spätere Hinzufügen von Gruppen-Budgets ermöglicht.

10.10.3 Datenmodell

Das Feature führt zwei neue Datenbanktabellen ein, die in Tabelle 10 detailliert aufgeführt sind.

Tabelle	Spalten	Beschreibung
<code>student_groups</code>	<code>id, name, description, created_by_id, timestamps</code>	Haupttabelle für Gruppen
<code>student_group_members</code>	<code>group_id, user_id</code>	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.

10.10.4 Backend-Implementierung

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

10.10.4.1 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

10.10.4.2 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

10.10.4.3 REST-API Endpoints

In der Tabelle 11 sind die Endpunkte des `StudentGroupController`s zu sehen:

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

10.10.4.4 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.

10.10.5 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.

10.11 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.

10.11.1 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)

10.11.2 Architektur-Entscheidung: Staging-Tabelle vs. Direktimport

Für die Verarbeitung eingehender InSy-Daten wurden zwei Ansätze evaluiert:

10.11.2.1 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.

10.11.2.2 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

Die Entscheidung fiel auf Option B, da die manuelle Kontrolle über importierte Gegenstände ein wichtiges Qualitätsmerkmal darstellt und der Kunde explizit eine Admin-Oberfläche zur Prüfung gewünscht hat.

10.11.3 Backend-Implementierung

In diesem Abschnitt wird die Implementierung der Geschäftslogik innerhalb der Spring Boot Architektur erläutert sowie das Zusammenspiel von REST-Controllern, Services und der Datenbank.

10.11.3.1 Entity: InsyImportItem

In der Tabelle 12 sind die Felder der Staging-Entity zu sehen, welche die BaseEntity erweitert und alle relevanten Felder aus InSy speichert:

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

10.11.3.2 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)

10.11.3.3 REST-Endpoints

Für die Interaktion mit dem Import-Modul wurden dedizierte Schnittstellen bereitgestellt. Tabelle 13 listet die implementierten REST-Endpoints auf, die für den Datenempfang sowie die Verwaltung der Import-Einträge zur Verfügung stehen.

Methoden	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

10.11.3.4 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. In Tabelle 14 sind diese Actions zu sehen:

10.11.4 Entwicklungsgeschichte und Iterationen

Die Implementierung erfolgte in mehreren Phasen, wobei auf Basis von Tests und Feedback iterative Verbesserungen durchgeführt wurden.

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

10.11.4.1 Phase 1: Initiale Backend-Implementierung

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

10.11.4.2 Phase 2: Erweiterung um Import-Logik

Nach ersten Tests zeigte sich, dass der Admin über erweiterte Kontrollmöglichkeiten verfügen muss. Die Aktionen wurden auf IMPORT_AS_NEW, IMPORT_TO_EXISTING und REJECT umgestellt, wie in Tabelle 15 dargestellt.

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

10.11.4.3 Phase 3: Frontend-Backend-Synchronisation

Bei der Frontend-Implementierung wurde festgestellt, 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.

Es wurde 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: "..." }
```

10.11.4.4 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

10.11.4.5 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)

10.11.4.6 UX-Iterationen

Während der Frontend-Entwicklung wurden 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' }"`.

10.11.4.7 Service-Methoden

In Tabelle 16 sind die vom HTTP-Service bereitgestellten Methoden dargestellt:

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

10.11.5 Mock-Daten für Entwicklung

Für die Entwicklung ohne echte InSy-Anbindung wurde ein Mock-Endpoint implementiert:

`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.

10.11.6 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.

10.12 Sicheres Token-basiertes QR-Code-System

Das ursprüngliche, statische QR-Code-System wurde 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.

10.12.1 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.

10.12.2 Architektur-Entscheidung: Token-Transaktionen

Es wurde eine Entkopplung von Buchungs-ID und QR-Code implementiert. Anstelle des Scannens der ID wird eine `BookingTransaction`-Entity generiert.

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.

10.12.3 Backend-Implementierung

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

10.12.3.1 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

10.12.3.2 Idempotenz und Cleanup

Um Datenbank-Ressourcen zu schonen und die User Experience zu verbessern, wurde eine Idempotenz-Prüfung implementiert: Fragt ein User einen Token an, während noch ein gültiger (nicht abgelaufener) Token für diese Buchung existiert, wird der bestehenden Token zurückgegeben, anstatt einen neuen zu generieren. Alte, ungenutzte Tokens werden vor einer Neu-Generierung invalidiert. In Tabelle 17 sind die neu hinzugefügten Endpunkte zu sehen.

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

10.12.4 Frontend-Integration

Die Integration im Angular-Frontend erfolgt rollenspezifisch getrennt.

10.12.4.1 Studenten-Sicht (User Dashboard)

In der `BookingQrComponent` wird der QR-Code angezeigt. Ein Live-Timer informiert den Studierenden über die verbleibende Gültigkeitsdauer des Codes (Countdown von 15 Minuten). Nach Ablauf der Gültigkeit wird eine visuelle Benachrichtigung angezeigt, und es besteht die Möglichkeit, einen neuen Code anzufordern.

10.12.4.2 Verleiher-Sicht (Lender Dashboard)

Im Dashboard des Verleiher wurde 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").

10.12.5 Zusammenfassung

Durch die Implementierung des Token-Systems wurde 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.

10.13 Qualitätssicherung und Backend-Hardening

Um die Stabilität der Geschäftslogik und die Sicherheit der API-Endpunkte zu gewährleisten, wurde eine zweigleisige Strategie verfolgt: eine umfassende Testabdeckung gemäß der Testpyramide und ein gezielter Security-Hardening der bestehenden Endpoints.

10.13.1 Test-Architektur

Wie in Tabelle 18 dargestellt, wurden die Tests in vier logische Ebenen unterteilt. Dies ermöglicht die isolierte Identifikation von Fehlerquellen – sei es in der Datenbankabfrage, der Geschäftslogik oder der API-Schnittstelle.

Ebene	Technologie	Fokus
Repository Layer	@DataJpaTest, H2 DB	Validierung von Custom JPQL-Queries
Service Layer	Mockito, JUnit 5	Isolierte Geschäftslogik ohne Datenbank
Controller (Functional)	@WebMvcTest	HTTP-Status, JSON-Mapping, Validierung
Controller (Security)	@WebMvcTest, Spring Security	Zugriffsschutz und Rollen-Checks

Tabelle 18: Übersicht der implementierten Test-Ebenen

10.13.2 Validierung der Geschäftslogik

Der Kern unserer Geschäftslogik liegt in den Services. Da hier komplexe Abläufe stattfinden, war eine hohe Testabdeckung essenziell.

10.13.2.1 BookingService

Der `BookingService` enthält die zentrale Logik für Ausleihen. Insbesondere folgende Szenarien wurden mit Mockito getestet:

- **Status-Berechnung:** Korrekte Übergänge von PENDING zu CONFIRMED, PICKED_UP und RETURNED.
- **Verfügbarkeit:** Die Prüfung, ob ein Item im gewählten Zeitraum bereits gebucht ist.
- **Gruppen-Logik:** Sicherstellung, dass nur Mitglieder einer Gruppe für diese buchen können.

10.13.2.2 InsyImportService

Für den Import externer Daten aus dem InSy-System wurde der `InsyImportService` intensiv getestet. Da hier Daten aus einer unzuverlässigen Quelle (Push-API) verarbeitet werden, musste sichergestellt werden, dass:

- Bestehende PENDING-Einträge aktualisiert statt dupliziert werden.
- Der Batch-Import hunderte Items korrekt einem Produkt zuordnet.
- Fehlerhafte Daten (z.B. fehlende Inventarnummer) sauber abgefangen werden.

10.13.2.3 Entity Logic

Zeitabhängige Logiken, wie die Berechnung des Status EXPIRED nach 24 Stunden ohne Abholung, wurden direkt in der `Booking`-Entity verankert und mittels performanter Unit-Tests (`BookingEntityTest`) ohne Spring-Kontext verifiziert.

10.13.3 Controller-Tests und Security-Integration

Eine besondere Herausforderung stellte das Testen der REST-Controller dar. Dazu wurden funktionale Tests (Datenebene) und Sicherheitstests (Zugriffsebene) getrennt, um die Komplexität zu reduzieren.

10.13.3.1 Funktionale Tests

Hierzu wird geprüft, ob die API korrekte HTTP-Statuscodes und JSON-Strukturen liefert. Besonders wichtig waren die Tests für den `BookingController` und `InsyImportController`, da diese komplexe DTOs erwarten.

- POST /api/bookings: Liefert 201 Created bei Erfolg.
- POST /api/bookings: Liefert 400 Bad Request, wenn Pflichtfelder fehlen.

10.13.3.2 Security-Tests und Mocking

Um sicherzustellen, dass unsere `@PreAuthorize`-Annotationen greifen, wurden dedizierte Security-Tests implementiert. Ein technisches Hindernis war der `UserSyncFilter`, der bei jedem Request versucht, User-Daten zu synchronisieren. Da in den Tests keine Datenbankverbindung besteht, wurde der Filter gemockt und die Filter-Chain manuell weitergeleitet:

```
doAnswer(invocation -> {
    chain.doFilter(request, response);
    return null;
}).when(userSyncFilter).doFilter(any(), any(), any());
```

10.13.4 Sicherheitsoptimierungen (Hardening)

Parallel zur Testautomatisierung wurde ein Sicherheitsaudit durchgeführt. Dabei wurde festgestellt, dass Spring Security zwar die Authentifizierung (JWT-Token), aber oft nicht die Autorisierung (Rollenrechte) auf Endpunkt-Ebene prüfte.

10.13.4.1 Method-Level Security

Dazu wurden die `@EnableMethodSecurity`-Annotation aktiviert und konsequent `@PreAuthorize`-Checks in allen Controllern nachgerüstet. Insgesamt wurden 43 Endpoints abgesichert.

- **Admin-Schutz:** Schreibende Zugriffe in `ProductController`, `ItemController` und `InsyImportController` erfordern nun `hasRole('ADMIN')`.
- **Besitzrechte:** Bei Buchungen wird dynamisch geprüft, ob der User berechtigt ist: `@bookingSecurityService.canView(#id, authentication)`.

10.13.4.2 CORS-Bereinigung und Logging

Zusätzlich wurden die Sicherheitskonfiguration zentralisiert und bereinigt:

- **CORS:** Alle `@CrossOrigin`-Annotationen wurden entfernt. Die Konfiguration erfolgt nun zentral in der `SecurityConfig`, was inkonsistente Regeln verhindert.
- **Endpoint-Härtung:** Der Zugriff auf `/api/images/**` wurde eingeschränkt. Nur noch GET-Anfragen sind öffentlich, während DELETE-Operationen Admin-Rechte erfordern.
- **Logging:** Unsichere `System.out`-Aufrufe, die potenziell sensitive Daten leakten, wurden durch SLF4J-Logging ersetzt.

10.13.5 Fazit

Durch die Kombination aus automatisierten Tests und striktem API-Hardening konnte das Sicherheitsniveau signifikant erhöht werden. Kritische Prozesse wie der Datenimport sind nun gegen Regressionen geschützt, und unberechtigte Zugriffe werden zuverlässig unterbunden.

10.14 Deployment

Die Anwendung ist mit Docker auf einen Test- oder Production-Server deploybar. Jede Hauptkomponente kann unabhängig als Docker-Container deployed werden. Vorausgesetzt wird ein bereits laufender Keycloak-Server sowie eine laufende PostgreSQL Datenbank. Es wird empfohlen, das Backend und die Datenbank aus Latenzgründen auf dem gleichen physischen Server zu betreiben. Im Test-Deployment kam außerdem NginxProxyManager als Reverse Proxy zum Einsatz, um in der Backend URL keinen Port mit angeben zu müssen. Im Folgenden wird beschrieben wie die Möglichkeit das System zu deployen für Frontend und Backend realisiert wurde.

10.14.1 Frontend

Das Frontend wird mit dem bereitgestellten Dockerfile mit `docker build -t leihsy-frontend .` zu einem Docker Image gebaut. Enthalten sind die durch den build generierten HTML- und JavaScript- Dateien sowie ein NGINX Webserver der die Inhalte ausliefert.

10.14.1.1 Dockerfile Das Dockerfile installiert zunächst NPM und Angular CLI in den temporären Container, kopiert dann den Sourcecode und baut diesen zu fertigen HTML- beziehungsweise JavaScript- Dateien. Dann wird der NGINX Webserver installiert und die Webdateien werden auf den erwarteten Pfad des Webservers kopiert. Als nächstes wird die nginx.conf aus dem Projektordner in den temporären Container kopiert. Schlussendlich wird der Entrypoint des Docker Containers als Start des NGINX Webservers festgelegt, so dass dieser direkt gestartet wird wenn der Container erstellt wird.

10.14.1.2 Konfiguration Der NGINX Webserver kann durch ein anpassen der mitgelieferten nginx.conf frei konfiguriert werden. Es ist zu beachten dass die nginx.conf im gleichen Pfad bleiben muss, damit diese für den Container übernommen wird. Nach jeder Änderung der nginx.conf muss das Docker Image neu gebaut werden.

10.14.1.3 Docker-Compose Wenn das Docker Image gebaut ist kann es mit der mitgelieferten docker-compose.yml mit dem Befehl `docker compose up -d` gestartet werden. Damit ist der Docker Container gestartet und das Frontend abrufbar.

10.14.2 Backend

Das Backend wird mit dem bereitgestellten Dockerfile mit `docker build -t leihsy-backend .` zu einem Docker Image gebaut. Enthalten ist die gebaute Java Applikation.

10.14.2.1 Dockerfile Das Dockerfile holt zuerst die Dependencies aus der pom.xml aus dem Projekt. Dann wird der Sourcecode in den temporären Container kopiert und die Anwendung mit Maven gebaut. Die fertig gebaute Java Applikation wird in das Arbeitsverzeichnis kopiert. Danach wird der Entrypoint als Start der Java Applikation gesetzt, damit diese beim Start des Containers direkt ausgeführt wird.

10.14.2.2 Konfiguration Das Backend kann durch das Setzen von Umgebungsvariablen konfiguriert werden. Es wird empfohlen, die Umgebungsvariablen in der mitgelieferten docker-compose.yml zu modifizieren. Es können die URL der PostgreSQL Datenbank, der Nutzernamen für die Datenbank, das Passwort für die Datenbank, die Allowed Origins und die Keycloak Client ID als Umgebungsvariablen konfiguriert werden. Die Allowed Origins Umgebungsvariable muss immer auf die Basis-URL des Frontends gesetzt werden

um ein Blockieren der API-Calls durch den Browser aus Sicherheitsgründen zu vermeiden. Der Speicherort der im System hochgeladenen Bilder kann ebenfalls in der mitgelieferten docker-compose.yml geändert werden. Hier kann ein relativer Pfad zum Verzeichnis der docker-compose.yml angegeben werden. Nach einer Änderung der Umgebungsvariablen muss der Docker Container neu gestartet werden damit die Änderungen wirksam werden.

10.14.2.3 docker-compose.yml Wenn das Docker Image gebaut ist kann der Docker Container mit dem `docker compose up -d` gestartet werden. Damit ist der Docker Container gestartet und die bereitgestellten Endpunkte der REST-API aufrufbar.

11 Aufgabenverteilung

Im Rahmen des Projekts wurden die anfallenden Aufgaben in die Bereiche Frontend, Backend sowie projektübergreifende Tätigkeiten unterteilt. Die folgende Auflistung dokumentiert die Aufgabenverteilung innerhalb des Teams. Die hauptverantwortliche Person wird fett-markiert dargestellt, während andere Person die ebenfalls an der Aufgabe beteiligt waren normal dargestellt werden.

11.1 Frontend

Die detaillierte Zuweisung der Verantwortlichkeiten für die einzelnen Komponenten der Benutzeroberfläche ist in Tabelle 19 aufgeführt.

Tabelle 19: Aufgabenverteilung im Frontend

Aufgabe	User Story	Personen
Grundgerüst	–	Asinas
Katalogseite und weiterführende Seiten	US-007, US-008	Asinas , Dennis, Malte
Menü-Seiten	–	Dennis
Nutzer-Buchungsseite und weiterführende Seiten	US-017, US-032	Dennis
Nutzer Gruppen und weiterführende Seiten	US-044	Dennis
Verleiher-Gegenständeseite und weiterführende Seiten	US-003	Dennis

Aufgabe	User Story	Personen
Verleiher Ausleihübersichtseite und weiterführende Seiten	US-029	Ceyda
Verleiher Anfragenseite und weiterführende Seiten	US-013, US-014	Ceyda
Verleiher-Privategegenstände und weiterführende Seiten	US-042	Dennis
QR-Code	US-030	Dennis, Yannick
Admin Produktverwaltung und weiterführende Seiten	US-004, US-006, US-009, US-041	Dennis, Yannick
Admin Gegenstandsverwaltung und weiterführende Seiten	US-004, US-005, US-006	Dennis
Admin Gegenstandsübersicht und weiterführende Seiten	–	Dennis
Admin Buchungsseite und weiterführende Seiten	US-025, US-036	Dennis
Admin Insy Import und weiterführende Seiten	US-037	Yannick
Admin Verleiherzuordnung und weiterführende Seiten	US-034	Ceyda
Admin Kategorieverwaltung und weiterführende Seiten	US-027	Ceyda
Admin Zusatzgegenstände und weiterführende Seiten	US-026	Ceyda

Aufgabe	User Story	Personen
Admin Standortverwaltung	–	Dennis
Admin Gruppenübersicht und weiterführende Seiten	US-044	Dennis
Warenkorb	US-010, US-011, US-012	Malte
Auth Guards & Interceptors	US-001, US-002	Yannick
Anbindung an REST-Schnittstellen	–	Yannick, Dennis

11.2 Backend

Analog zur Frontend-Entwicklung schlüsselt Tabelle 20 die Verantwortlichkeiten für die Umsetzung der serverseitigen Komponenten und Business-Logik auf.

Tabelle 20: Aufgabenverteilung im Backend

Aufgabe	User Story	Personen
Projekt-Setup & Grundkonfiguration (Spring Boot, Maven, Profiles)	US-039	Yannick
Entity-Modell & Datenbankstruktur (BaseEntity, Soft-Delete, Lifecycle-Hooks)	US-023, US-024	Yannick
Keycloak-Integration & Security-Konfiguration (OAuth2, JWT, Rollen)	US-001, US-002	Yannick, Malte
ReminderService (Scheduled Reminders)	US-0021	Asinas
Email-Service (Erinnerungsmails)	US-0021	Asinas

Aufgabe	User Story	Personen
Email-Service (Statusänderungsmails)	US-0022	Asinas
Email-Service (Bestätigungsmaile)	US-0018	Asinas
Automatische Benutzersynchronisation (UserSyncFilter, UserService)	US-001	Yannick
User-Ressource (CRUD, Suche))	US-001	Yannick
Product-Ressource (CRUD, Suche, Verfuegbarkeitszeitraume)	US-004, US-005, US-006, US-007	Yannick
Item-Ressource (CRUD, Related Items, Lender-Zuordnung)	US-004, US-034	Yannick
Verknüpfung Zusatzgegenstände mit Hauptgegenständen	US-026	Ceyda
Category-Ressource (CRUD)	US-027	Yannick
Location-Ressource (CRUD)	–	Yannick
Booking-Ressource (CRUD, Status-Workflow, Ping-Pong-Terminvorschlaege)	US-012, US-013, US-014, US-016, US-017	Yannick
BookingTransaction & QR-Code-System (Token-Generierung, Ausgabe/Rueckgabe)	US-018, US-019, US-020, US-030	Yannick
Rückgabe-Workflow (Status-Update, Protokollierung, Zeitstempel)	US-020	Asinas

Aufgabe	User Story	Personen
Booking-Scheduler (Auto-Cancel, Auto-Expire)	US-015	Yannick
Lender-Endpoints (Upcoming, Active, Overdue, Pending)	US-029	Yannick, Asinas
StudentGroup-Ressource (CRUD, Mitgliederverwaltung, Gruppenbuchungen)	US-044	Yannick
InSy-Import-Ressource (Empfangen, Review-Workflow, Batch-Import)	US-037, US-038	Yannick
Bildverwaltung (Upload, Validierung, Speicherung)	US-009	Yannick
Lender-Anfragen (Pending-Dashboard, 24h-Frist, Sortierung)	US-013	Asinas
Item-Administration (Verleiher-Zuordnung, Admin-Filter)	US-034	Asinas
MapStruct-Mapper (DTO-Pattern, Entity-Konvertierung)	–	Yannick, Asinas
API-Umstrukturierung nach REST Best Practices	–	Yannick
Swagger/OpenAPI-Dokumentation	–	Yannick, Asinas, Malte
GlobalExceptionHandler & Custom Exceptions	–	Yannick

Aufgabe	User Story	Personen
Postman-Collection & API-Testing	–	Yannick, Asinas
Datenbank-Konfiguration (Spring Profiles, JPA, Connection Pool)	–	Yannick

11.3 Projektübergreifende Aufgaben

Abschließend fasst Tabelle 21 die Zuständigkeiten für organisatorische und infrastrukturelle Aufgaben zusammen, die das gesamte Projekt betreffen.

Tabelle 21: Aufgabenverteilung für projektübergreifende Tätigkeiten

Aufgabe	User Story	Personen
User Research	–	Ceyda, Asinas
Erstellung von Mockups und Wireframes	–	Asinas
Definition von User Stories	–	Yannick
Wireguard VPN Einrichtung	–	Malte, Ceyda
Server- und Datenbankeinrichtung	–	Malte, Ceyda
Emailservereinrichtung	–	Malte
Dockerdeployment	–	Malte
GitHub-Verwaltung und CI/CD-Pipeline	US-046	Dennis, Malte
Keycloak einrichten	US-001, US-002	Yannick, Dennis
Pflege der Projektdokumentation	–	Alle

12 Reflektion

Dieser Abschnitt handelt von der Reflektion der Projektdurchführung. Dabei werden die Zeitplanung, der Lernfortschritt sowie das Projektmanagement näher betrachtet.

12.1 Umsetzung der User Stories

In diesem Abschnitt wird reflektiert, welche der geplanten User Stories und Features während des Projekts umgesetzt wurden, welche zusätzlichen Features entstanden sind und welche Nice-to-Have Features noch offenstehen.

12.1.1 Umgesetzte Must-Have Features

Alle wesentlichen Must-Have Features konnten erfolgreich implementiert werden:

- **Authentifizierung & Benutzerverwaltung:** Keycloak SSO-Integration mit automatischem User-Sync über den `UserSyncFilter`, rollenbasierte Zugriffskontrolle (Admin, Lender, User) mit Security Services
- **Gegenstandsverwaltung:** Vollständiges 3-Ebenen-Modell (Category → Product → Item) mit CRUD-Operationen, Bildupload-Funktionalität, Kategorisierung und Standortverwaltung
- **Ausleihprozess:** Implementierung des kompletten Booking-Workflows mit 7 Status-Zuständen (PENDING, CONFIRMED, PICKED_UP, RETURNED, REJECTED, EXPIRED, CANCELLED), Ping-Pong Terminvorschlag-System mit `proposedPickups` als JSON-Array
- **E-Mail-System:** Erinnerungs- und Bestätigungsmails über den `EmailService` mit SMTP-Integration
- **Protokollierung:** Audit-Log über `BookingTransaction` mit Token-basiertem QR-Code-System für Ausgabe und Rückgabe
- **Datenschutz:** Soft-Delete-Mechanismus in allen Entities, `deletedAt`-Zeitstempel, datenschutzkonforme Speicherung nach DSGVO-Prinzipien
- **Containerisierung:** Vollständiges Docker-Setup mit Docker-Compose für PostgreSQL, Backend und Frontend

12.1.2 Umgesetzte Should-Have Features

Darüber hinaus konnten mehrere ursprünglich als Should-Have geplante Features realisiert werden:

- **Verleiher-Zuordnung:** Items können spezifischen Verleihern zugeordnet werden über `Item.lender_id`
- **Batch-Erstellung:** Vereinfachte Erstellung mehrerer Items zu einem Product

- **Detailseite mit Verfügbarkeitskalender:** Product-Detailseite zeigt verfügbare und nicht verfügbare Zeiträume an
- **Automatische Stornierung:** Scheduler-basierte Auto-Cancel (PENDING nach 24h) und Auto-Expire (CONFIRMED ohne Abholung nach 24h) Funktionen
- **Responsive UI:** Mobile-First Design mit PrimeNG 20.3.0 und TailwindCSS 4.1.16
- **Ausleihverlauf:** Studierende können ihren vollständigen Booking-Verlauf einsehen
- **Admin-Dashboard:** Statistiken mit Chart.js, Booking-Übersichten, PDF-Export-Funktionalität

12.1.3 Umgesetzte Nice-to-Have Features

Einige Nice-to-Have Features konnten ebenfalls implementiert werden:

- **StudentGroup-Feature (Höchste Priorität):** Vollständige Gruppenverwaltung mit Mitgliederverwaltung, Gruppenbuchungen und eigenem Budget-Feld
- **InSy-Import-System (Hohe Priorität):** Staging-Bereich für Importe aus dem Inventarisierungssystem mit Review-Workflow, `InsyImportItem` Entity, Batch-Import und Mock-Daten
- **QR-Code-System (Hohe Priorität):** Token-basierte Ausgabe und Rückgabe mit 8-stelligen Tokens, 15 Minuten Gültigkeit, automatische Typ-Ermittlung über `BookingTransactionType`
- **Private-Lend-Bereich (Mittlere Priorität):** Lender-Dashboard mit separatem Bereich für eigene Produkte, Upcoming Pickups, Active Loans und Overdue Returns
- **Related Items (Mittlere Priorität):** M:N Beziehung zwischen Items für Accessories und Requirements
- **CI/CD-Pipeline (Niedrige Priorität):** GitHub Actions Workflow für automatische Builds und Tests, Integration mit SonarQube für Code-Qualitätsanalyse

12.1.4 Zusätzlich umgesetzte Features

Während der Entwicklung wurden zusätzliche Anforderungen identifiziert, die ursprünglich nicht in den User Stories enthalten waren:

- **MapStruct Integration:** Automatische Entity-DTO-Konvertierung über 7 Mapper (Booking, Product, Item, StudentGroup, InsyImport, Category, Location) zur Vermeidung von Circular Reference Problemen

- **Swagger/OpenAPI-Dokumentation:** Umfassende API-Dokumentation mit über 80 Endpoints, `@Operation` und `@ApiResponse` Annotations
- **Security Services:** Dedizierte Services für Authorization (`BookingSecurityService`, `StudentGroupSecurityService`, `UserSecurityService`) mit `@PreAuthorize` Integration
- **Export-Funktionen:** PDF-Export für Buchungen und Statistiken mit jsPDF und html2canvas
- **Cart-System:** Warenkorb mit LocalStorage-Persistenz für nahtlose User Experience
- **Item-Location:** Items haben eigene Standorte unabhängig vom Product-Standort für präzisere Lagerverwaltung
- **Unit-Tests:** Backend-Tests mit JUnit 5 und Mockito für Repository- und Service-Layer, JaCoCo für Code-Coverage-Analyse

12.1.5 Nicht umgesetzte Nice-to-Have Features

Folgende Nice-to-Have Features konnten im Projektzeitraum nicht realisiert werden:

- **ProductSet-Feature:** Entity und Repository existieren (`findRecommendedProducts`), aber kein Controller/Service oder Frontend-UI vorhanden
- **Budget-System Business-Logik:** `User.budget` und `StudentGroup.budget` Felder existieren, aber keine Berechnungslogik für Tagessätze oder Budget-Tracking implementiert
- **Dozenten-Freigabe-Workflow:** Kein Genehmigungs-System für Dozenten implementiert
- **Detaillierte Budget-Reports:** Keine umfassenden Auswertungen über Budgetauslastung und Kosten
- **Verleihgruppen:** Keine Zusammenarbeit mehrerer Verleiher als Gruppe
- **Frontend-Tests:** Jasmine und Karma sind konfiguriert, aber keine Unit- oder E2E-Tests geschrieben
- **Umfassendes Audit-Log:** Die `BookingTransaction`-Entity wurde zur Protokollierung von Ausgabe- und Rückgabevorgängen mit QR-Code-Token implementiert. Ein vollständiges Audit-Log für alle Systemaktionen fehlt jedoch weiterhin. Login-Versuche, Änderungen an Gegenständen und administrative Eingriffe werden derzeit

nicht systematisch erfasst. Die geforderte DSGVO-konforme Datensparsamkeit mit automatischer Löschung temporärer Daten und verschlüsselter Session-Übertragung wurde nur teilweise umgesetzt: Ein Soft-Delete über den `deletedAt`-Zeitstempel ist vorhanden, jedoch sind definierte Aufbewahrungsfristen sowie die automatische Löschung von Warenkorb-Daten nach Session-Ende noch nicht implementiert.

12.1.6 Zusammenfassung

Von den ursprünglich geplanten Features wurden:

- **100% der Must-Have Features** vollständig umgesetzt
- **Alle wichtigen Should-Have Features** realisiert
- **6 von 8 Nice-to-Have Features** mit hoher und mittlerer Priorität implementiert (inkl. CI/CD-Pipeline)
- **10 zusätzliche Features** entwickelt, die während des Projekts als notwendig identifiziert wurden

Die Priorisierung hat sich bewährt: Durch den Fokus auf Must-Have Features entstand eine vollständig funktionsfähige Verleihplattform. Die zusätzlich implementierten Features wie MapStruct, Multi-Profile-Konfiguration, Security Services und Unit-Tests waren essenziell für eine professionelle und wartbare Code-Basis. Die nicht umgesetzten Features (Budget-System, ProductSet, Verleihgruppen, Frontend-Tests) stellen sinnvolle Erweiterungen für zukünftige Iterationen dar, beeinträchtigen aber nicht die Kernfunktionalität des Systems.

12.2 Reflektion zum Projektmanagement

Im Projekt wurde deutlich, dass erfolgreiches Projektmanagement nicht nur aus Boards und Tools besteht, sondern vor allem von klarer Kommunikation abhängt. Ein zentraler Aspekt war die frühzeitige Klärung von Erwartungen und Anforderungen innerhalb des Teams. Unklarheiten führten andernfalls schnell zu Missverständnissen und im schlimmsten Fall zu doppelter Arbeit. Eine konsequente Abstimmung – durch Nachfragen, klar formulierte Aufgaben und dokumentierte Entscheidungen – trug wesentlich zu einer ruhigeren und effizienteren Zusammenarbeit bei.

Darüber hinaus wurde die Notwendigkeit realistischer Planung erkannt. Zu Beginn wurde häufig angenommen, dass Aufgaben „schnell erledigt“ seien, während in der Praxis Abstimmungen, kleine Änderungen, Bugfixes oder Dokumentation zusätzlichen Aufwand verursachten. Mit zunehmender Erfahrung konnten Aufgaben besser strukturiert und Fort-

schritt sowie erforderlicher Aufwand präziser eingeschätzt werden.

Außerdem zeigte sich, dass kontinuierliches Feedback und regelmäßige Check-ins von großem Nutzen sind. Insbesondere Meetings innerhalb des Teams sowie mit der Betreuungsperson ermöglichen ein frühzeitiges Erkennen von Fehlentwicklungen oder Blockaden. Insgesamt verdeutlichte das Projekt, dass die Zusammenarbeit im Team eine ebenso zentrale Rolle spielt wie die technische Umsetzung.

12.3 Reflektion zur Zeitplanung

Die ursprüngliche Zeitplanung des Projekts basierte auf der Annahme, dass die Entwicklung des Backends den größten zeitlichen Aufwand verursachen würde, während das Frontend vergleichsweise schneller umzusetzen sei. Diese Einschätzung erwies sich im Projektverlauf als unzutreffend. In der Praxis nahm insbesondere die Entwicklung und Feinabstimmung des Frontends deutlich mehr Zeit in Anspruch als zunächst geplant, etwa durch die Umsetzung von einer Komponenten basierten Darstellung, responsiven Designs und wiederholter Anpassungen an geänderte Anforderungen.

Ein weiterer Aspekt, der in der initialen Zeitplanung nicht ausreichend berücksichtigt wurde, war der kontinuierliche Wartungsaufwand. Sowohl im Backend als auch im Frontend mussten bestehende Codebestandteile regelmäßig an neue Anforderungen, Schnittstellenänderungen oder strukturelle Anpassungen angepasst werden. Dieser fortlaufende Aufwand führte zu zusätzlichen Zeitinvestitionen, die in der ursprünglichen Planung nicht eingeplant waren.

Ähnliches gilt für die Dokumentation des Projekts. Änderungen am Quellcode machten wiederholte Aktualisierungen der technischen Dokumentation notwendig, was ebenfalls mehr Zeit beanspruchte als erwartet. Die Dokumentation konnte daher nicht ausschließlich als abschließender Projektschritt betrachtet werden, sondern begleitete den Entwicklungsprozess kontinuierlich.

Darüber hinaus wurden unvorhergesehene Probleme, wie technische Schwierigkeiten, Fehlannahmen in der Architektur oder kurzfristig auftretende Fehler, in der Zeitplanung nicht explizit einkalkuliert. Diese führten zu zusätzlichen Verzögerungen.

Trotz dieser Abweichungen war die erste Planung nicht allzu knapp bemessen, da die Features insgesamt großzügig eingeplant wurden. Dies stellte sicher, dass genügend Zeit für die Umsetzung der Kernfunktionalitäten zur Verfügung stand, auch wenn einzelne Aspekte - insbesondere Frontend-Feinabstimmungen und Wartung - mehr Zeit benötigten als

ursprünglich angenommen.

Im GitHub Repository der Dokumentation sind außerdem umfangreiche Time-Tracking Reports von Clockify zu finden. In diesen sind die getrackten Zeiten und Aufgaben aller Teammitglieder.

12.4 Reflektion zum Lernfortschritt

Im Verlauf des Projekts konnte ein deutlicher Lernfortschritt in mehreren fachlichen und organisatorischen Bereichen erzielt werden. Durch den Einsatz moderner Technologien und Frameworks, die sich am Industriestandard orientieren, wurde praxisnahes Wissen aufgebaut, das über theoretische Inhalte hinausgeht. Die Arbeit mit aktuellen Frontend- und Backend-Technologien förderte ein besseres Verständnis für etablierte Entwicklungswerkzeuge und -prozesse in realen Softwareprojekten.

Ein wesentlicher Lernaspekt war das vertiefte Verständnis von Fullstack-Architekturen. Insbesondere die Kommunikation zwischen Frontend und Backend über REST-Schnittstellen verdeutlichte, wie einzelne Systemkomponenten miteinander interagieren und welche Anforderungen an Schnittstellendesign, Datenformate und Fehlerbehandlung gestellt werden.

Darüber hinaus wurde die Bedeutung einer strukturierten und nachvollziehbaren Dokumentation deutlich. Das Erstellen und fortlaufende Pflegen der technischen Dokumentation erwies sich als essenziell, um Architekturentscheidungen, Schnittstellen und Implementierungsdetails verständlich festzuhalten und langfristig wartbar zu machen.

Im Umgang mit Herausforderungen konnte gelernt werden, Probleme systematisch zu analysieren und lösungsorientiert anzugehen. Unerwartete technische Schwierigkeiten oder Planabweichungen machten es notwendig, flexibel zu reagieren, Alternativen zu evaluieren und getroffene Entscheidungen kritisch zu hinterfragen.

Auch im Bereich der Projektorganisation ergaben sich wichtige Lernerfahrungen. Die Planung von Zeit und Ressourcen sowie die Analyse und Priorisierung von Anforderungen erwiesen sich als zentrale Faktoren für den Projekterfolg. Anforderungen mussten regelmäßig neu bewertet und an den verfügbaren zeitlichen Rahmen angepasst werden.

Nicht zuletzt trug die Zusammenarbeit im Team maßgeblich zum Lernfortschritt bei. Die Aufgabenverteilung, Abstimmung von Schnittstellen und gegenseitige Unterstützung förderten nicht nur die Effizienz, sondern auch die Kommunikations- und Kooperationsfähigkeit innerhalb des Teams. Insgesamt stellte das Projekt eine wertvolle Lernerfahrung

dar, die sowohl technische als auch methodische Kompetenzen nachhaltig stärkte.

13 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.

13.1 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.

13.2 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 Rückgabeprozesse)
- **Projektgruppenverwaltung** (Erstellung und Verwaltung von Projektgruppen im Vorlesungskontext)

13.3 Funktionale Erweiterungen

13.3.1 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.

13.3.2 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.

13.3.3 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.

13.3.4 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.

13.3.5 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)