Dokumentation Webanwendung LoanIt

Finn Gorell

ON24-3, November 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Thema des Projekts

LoanIt ist eine kompakte Webanwendung, die das Leihen und Verleihen von Gegenständen innerhalb kleiner Gruppen, Nachbarschaften, Wohngemeinschaften oder Lerngruppen, digital und nachvollziehbar gestaltet. Die Anwendung ermöglicht es Nutzer*innen, ein persönliches Inventar anzulegen, verfügbare Objekte anderer Personen einzusehen und diese auszuleihen oder zurückzugeben. Dabei bildet die Anwendung für jedes Objekt klar definierte Zustände ab, sodass jederzeit ersichtlich ist, wem etwas gehört, wer es aktuell nutzt und in welchem Status es sich befindet.

1.1 Motivation und Zielsetzung

Im privaten Umfeld kommt es häufig dazu, dass Gegenstände an Freunde, Familie oder Bekannte ausgeliehen, aber nur selten wieder zurückgegeben werden. Die Kommunikation und “Dokumentation” findet hierbei in den meisten Fällen über den direkten Kontakt in Form von Chats oder in mündlichen Absprachen statt. Dadurch fehlt es an Transparenz, welche Person ein Objekt aktuell besitzt und wie lange es bereits verliehen ist.

Diese Situation ist vermutlich bei jedem bereits aufgekommen, wenn auch nicht regelmäßig.

LoanIt setzt genau an diesem Problem an: Sie soll nicht nur eine bessere Struktur schaffen, sondern gleichzeitig demonstrieren, wie moderne Web Architekturen solche Alltagsprobleme lösen können und wie eine solche Anwendung in diesem Fall mit Vue und Deno/Oak umgesetzt werden können.

Im Fokus stehen dabei wesentliche Ziele wie die Authentifizierung, Session-Management, die in der Server-Runtime integrierte Datenbankspeicherung und reaktive Benutzeroberflächen.

1.2 Zielgruppe

Die Zielgruppe umfasst kleinere Haushalte oder Freundes-/Bekanntenkreise, die Ressourcen wie Werkzeuge, Lernmaterialien oder Spiele regelmäßig verleihen und sich hierfür eine transparente Übersicht wünschen.

In späteren Versionen der Anwendung ist es möglich, auch unter einem Finanzrouter seine gemeinschaftlich getätigten Ausgaben abzubilden. Ein solcher Schulden-Tracker, in dem Schulden direkt beglichen werden können, eignet sich für eine solche Struktur sehr gut.

2. Ausgangssituation

Zu Beginn des Projekts verfügte ich über grundlegende Kompetenzen im Bereich der Frontend- und Webentwicklung. Meine bisherigen Erfahrungen lagen vor allem im Umgang mit HTML und CSS sowie in der Entwicklung kleinerer Weboberflächen und der Komponentenentwicklung mit Kotlin. JavaScript beherrschte ich auf einem Grundniveau, sodass ich grundlegende Logiken, die Fetch API und einfache Design-Patterns anwenden konnte. Auch wenn ich in kleineren Projekten bereits mit Vue.js gearbeitet hatte, beschränkte sich diese Arbeit größtenteils auf UI-bezogene Demos ohne Persistenz. Entsprechend fehlte mir praktische Erfahrung in der Entwicklung vollständiger, integrierter Anwendungen, bei denen Client und Server aufeinander abgestimmt sein müssen.

TypeScript hatte ich vor Projektbeginn lediglich in kleineren Übungen genutzt, meist zur Ergänzung bereits bestehender JavaScript-Logik. Auch die serverseitige Entwicklung mit Deno war mir bis dahin nur oberflächlich begegnet. Zwar hatte ich einzelne Aufgaben aus Vorlesungen umgesetzt, jedoch ohne eine weiterführende, tiefgehende Vertiefung. Kenntnisse zu sicherheitsrelevanten Aspekten wie Sessions, Zugriffskontrolle und Fehlerbehandlung waren zwar durch die Lehrveranstaltungen von Prof. Dr. Mester vermittelt, aber auf ein komplexeres Projekt noch nicht vertieft worden.

Die Ausgangssituation war somit geprägt von einem vorhandenen Grundverständnis der Webentwicklung, jedoch ohne tiefere Erfahrung im Aufbau eines vollständigen End-to-End-Systems. In GitHub und Visual Studio Code war ich hingegen bereits routiniert, sodass das technische Arbeitsumfeld selbst keine Herausforderung darstellte. Die eigentliche Herausforderung lag vielmehr in der Integration der verschiedenen Technologien zu einem Gesamtsystem, das sowohl funktional als auch strukturell konsistent ist.

3. Vorgehensweise und Planung

Zu Beginn des Projekts gab es eine konzeptionelle Phase, in der grundlegende Überlegungen zur Systemarchitektur, zu den technischen Schichten und zu den funktionalen Anforderungen gestellt wurden. In diesem Zusammenhang wurde definiert, wie Client und Server kommunizieren sollten, welche Datenstrukturen benötigt werden und wie die Anwendung später strukturiert sein soll. Diese Phase diente insbesondere der Orientierung und der Ausarbeitung eines sinnvollen Startpunktes. Meine Entscheidung war es, das Projekt entlang der Schichten einer modernen Webanwendung aufzubauen: ein Server auf Basis von Deno und Oak, eine Datenhaltung über Deno KV sowie ein komponentenbasiertes Frontend mit Vue.

Ab diesem Punkt entwickelte sich die Planung in einzelnen Arbeitspaketen. Anstelle den gesamten Projektverlauf vorab zu definieren, wurde nach jedem abgeschlossenen Arbeitsschritt als Erstes das Frontend und gegebenenfalls die Backend Funktionalität bewertet. Sobald eine Implementierung als funktionsfähig und abgeschlossen angesehen wurde, wurden weitere Implementierungen geplant. Dieses Vorgehen entspricht dem agilen Prinzip aus dem vergangenen Semester: Planung erfolgt nicht einmalig zu Beginn, sondern wird kontinuierlich erweitert und an neue Erkenntnisse angepasst. In der Praxis bedeute das, dass jede Entwicklung einen eigenen kleinen Zyklus aus Problemdefinition, Recherche, Implementierung, Test und Reflektion bildet.

Ein wesentlicher Bestandteil dieses Vorgehens war der intensive Wissenserwerb, der parallel zur eigentlichen Entwicklung stattfand. Da

Technologien wie Deno KV, Oak-Middleware oder bestimmte Vue-Patterns für mich neu waren, spielte die Einarbeitung eine zentrale Rolle. Diese erfolgte primär über die offizielle Dokumentation der jeweiligen Frameworks und Bibliotheken, ergänzt durch Tutorials und Beispielprojekte. Für mich waren hierbei “Experimente” besonders hilfreich, aber auch zeitintensiv: Viele Funktionen wurden zunächst testweise ausprobiert, bevor sie endgültig committed und fertiggestellt wurden. Auf diese Weise entstand ein „learning by doing“-Prozess. Unterstützend dazu kam die Nutzung von ChatGPT hinzu, die vor allem in der frühen Phase des Projekts dabei half, komplexe Zusammenhänge schneller zu verstehen oder Unsicherheiten aufzulösen. Die KI diente dabei jedoch nicht als Ersatz für eigene Überlegungen, sondern vielmehr als Ergänzung zu den verwendeten Dokumentationen und als Unterstützung bei der Fehlersuche.

Parallel zu diesem Lernprozess bildete sich eine klare Strukturierung des Projekts in Komponenten und Verantwortungsbereiche. Während die erste Phase noch stark von dem Versuch geprägt war, eine funktionierende Grundstruktur aufzubauen, verlagerte sich der Fokus im späteren Verlauf zunehmend auf die Ausarbeitung der Geschäftslogik und der Benutzerinteraktionen. Erst nachdem die Basisseiten existierten und die Routing-Struktur eingebaut wurde, habe ich die Frage geklärt, wie Nutzersitzungen verwaltet werden sollen. Diese Entscheidungen wurden im Verlauf des Projekts getroffen, sobald die Notwendigkeit entstand und das technische Verständnis ausreichend vorhanden war.

So entstand nach und nach eine vollständig funktionsfähige Oberfläche zur Verwaltung persönlicher Gegenstände und zur Ausleihe zwischen Nutzern. Funktionen wie die Statusanzeige, das dynamische Umschalten zwischen ausgeliehenen, verfügbaren und eigenen Items oder die Integration der borrowerId zeigen diese fortlaufende Weiterentwicklung.

Auch der Umgang mit Fehlern und unvorhergesehenen Herausforderungen wirkte sich auf die Vorgehensweise aus. Immer wieder zeigte sich, dass bestimmte Strukturen angepasst oder komplett neu überarbeitet werden mussten. Ein Beispiel hierfür ist die Verlagerung der Borrow-Logik von

BorrowView nach HomeView, die entstand, weil im Anwendungstest deutlich wurde, dass die Logik dort für den Nutzer besser aufgehoben ist. Diese Änderungen habe ich nicht als Rückschritt, sondern als Teil des Entwicklungsprozesses angesehen

Ein weiterer Teil des Vorgehens bestand in der Qualitätssicherung. Obwohl keine automatisierten Tests implementiert wurden, habe ich durch manuelles Testen, die Funktionsfähigkeit sicherstellen können. Dadurch konnten Fehler frühzeitig erkannt werden, und die Anwendung blieb auch während der Entwicklungsphasen großteils lauffähig. Die Arbeit am User Interface folgte einem ähnlichen Prinzip: Nach funktionaler Fertigstellung wurde die visuelle und interaktive Ebene mehrfach überarbeitet, etwa durch die Einführung einer Beschreibung pro Item, die Ergänzung von Status Labels oder die Verbesserung der Darstellung der allgemeinen Oberfläche und einzelner Komponenten.

4. Anforderungsliste

Nr.	Anforderung	Umsetzung	Kommentar
1	Nutzer*innen können sich registrieren	✓	Basisformular + Speicherung umgesetzt
2	Nutzer*innen können sich einloggen	✓	Sessions + Passwortprüfung
3	Session-Handling erfolgt serverseitig über Deno/Oak	✓	Cookies + Session Store
4	Aktive Sitzungen werden serverseitig validiert	✓	Geschützt durch Middleware
5	Nutzer*innen können eigene Gegenstände anlegen	✓	Formular + Persistenz
6	Beim Anlegen eines Gegenstands werden Name und Beschreibung erfasst	✓	Erweiterte Item-Modelle (title + description)

7	Gegenstände besitzen einen Status (verfügbar / ausgeliehen)	✓	Wird serverseitig verwaltet
8	Nutzer*innen sehen ihre eigenen Gegenstände in der Home-View	✓	Mit currentUserId gefiltert
9	Nutzer*innen können Gegenstände anderer Personen einsehen	✓	Grundfunktion vorhanden
10	Gegenstände können ausgeliehen werden	✓	Statusänderung + Borrow-View
11	Gegenstände können zurückgegeben werden	✓	Statuswechsel umgesetzt
12	Borrow-View zeigt Beschreibung und Status des Items an	✓	Änderungen im Template + Script
13	Datenpersistenz erfolgt zentral im Backend	✓	KV-Store oder JSON-Datei
14	API folgt REST-Prinzipien	✓	Endpoints sauber getrennt

15	Validierung serverseitig (Pflichtfelder)	✓	Basisvalidierung vorhanden
16	Anwendung hat ein einfaches, klares UI	✓	Vue-Komponenten + Styling
17	Clientseitige Anzeige reagiert automatisch auf Änderungen	✓	Reaktivität über Vue
18	Sicherheitsaspekte: Keine Aktionen ohne Session	✓	Middleware verhindert Zugriff

5. Konzeption

5.1 Technologie- & Werkzeugauswahl

Die Wahl fiel auf eine Kombination aus Deno, Oak, Deno KV und Vue 3, ergänzt durch gängige Werkzeuge wie GitHub und Visual Studio Code.

Deno wurde als serverseitige Laufzeitumgebung ausgewählt, da diese einen klar strukturierten Ansatz verfolgt. Insbesondere die Möglichkeit, ohne Abhängigkeit von paketbasierten Tools zu arbeiten, bot einen guten Einstieg und ermöglichte es, genau die Funktionen einzusetzen, die im Projekt benötigt wurden.

Die Verwendung von Oak als Middleware-Framework ermöglichte eine Request- und Routing-Logik. Gleichzeitig war die Wahl von Deno KV als persistente Datenspeicherung besonders attraktiv, da sie eine im Server-Runtime integrierte Datenbank als Lösung geboten hat.

Auf der Clientseite wurde bewusst Vue 3 gewählt, da die komponentenbasierte Architektur bereits zu den Vorlesungsinhalten passte und grundlegende Erfahrungen vorlagen. Vue bot durch seine Reaktivität einen idealen Rahmen für die Umsetzung interaktiver Oberflächen und Komponenten. Zudem erwies sich die Single-File-Component-Struktur als geeignet, um UI-Logik, Styles und Template-Strukturen abzubilden. GitHub und Visual Studio Code begleiteten den Entwicklungsprozess, wobei insbesondere GitHub durch die Commits nicht nur als Repository, sondern auch als Versionshistorie diente.

5.2 Entwurf

Client-Server-Architektur mit getrennten Frontend- und Backend-Services:

- Frontend: Vue 3 SPA (Composition API) mit Vite, Vue Router, vue-final-modal
- Backend: Deno mit Oak Framework (REST API)
- Kommunikation: HTTP/REST über CORS

Komponentenstruktur:

- App.vue: Root-Komponente, Auth-State, Modal-Steuerung
- AppHeader.vue: Navigation, Login/Logout
- HomeView.vue: Eigene Items verwalten, ausgeliehene Items anzeigen
- BorrowView.vue: Verfügbare Items durchsuchen und ausleihen
- Router: Zwei Routen (/, /ausleihen)

Hinweis: Die Projektstruktur befindet sich auf Seite 19

Backend-Routen:

- itemsRouter: CRUD für Items (GET, POST, PATCH, DELETE)
- usersRouter: Registrierung und Login
- Middleware: Joi-Validierung für Item-Operationen, CORS

Zusammenwirken:
Das Frontend nutzt localStorage für Session-Management (currentUserId). API-Calls erfolgen direkt per Fetch. Keine zentrale State-Verwaltung, jeder View verwaltet seinen eigenen State.

5.2.1 UI-Entwurf

Design-System:

- Vue-Farbpalette
- “Karten” Layout mit Schatten und abgerundeten Ecken
- Responsive Grid (2 Spalten)

Softwarestrukturen:

- Komposition über Vererbung: Composition API mit <script setup>
- Zentralisierte Styles: Globales CSS in App.vue
- Modal-Pattern: vue-final-modal für Authentifizierung (in App.vue)
- Conditional Rendering: v-if/v-else für Login-State und leere Listen

5.2.2 Datenmodell & Persistenz

Datenstrukturen:

```
Item {
 id: string (UUID)
 ownerId: string
 name: string
 description?: string
 status: "available" | "borrowed"
 borrowerId?: string | null
 }
User {
 id: string
 email?: string
 password: string
 }
```

Persistenz:

- Deno KV: Key-Value Store
- Key-Pattern: ["items", id] und ["users", id]
- Queries: Prefix für Item-Listen (kv.list({ prefix: ["items"] }))

Überlegungen:

- Deno KV: Einfach, integriert, ausreichend
- Keine Passwort-Hashing: Passwörter im Klartext (Sicherheitsrisiko)
- Keine Relationen: borrowerId als String-Referenz (keine Foreign Keys)

6. Ergebnis des Projekts

LoanIt ist eine vollständig funktionsfähige Webanwendung zur Verwaltung und zum Verleih von Gegenständen. Die Anwendung ermöglicht es Benutzern, ihre eigenen Gegenstände zu verwalten, verfügbare Gegenstände anderer Nutzer zu finden und auszuleihen sowie ausgeliehene Gegenstände zurückzugeben.

6.1 Funktionalität im Detail

Benutzerauthentifizierung

Die Anwendung verfügt über ein vollständiges Authentifizierungssystem. Neue Benutzer können sich mit Benutzername, optionaler E-Mail und Passwort registrieren. Bestehende Benutzer können sich mit Benutzername und Passwort anmelden. Für nicht angemeldete Nutzer öffnet sich automatisch ein Login-Modal vgl. vue-final-modal.org. Der Login-Status wird im LocalStorage gespeichert. Eine Logout-Funktion ist ebenfalls vorhanden und kann im Navigationsbereich gefunden werden. Die technische Umsetzung erfolgt über ein Modal auf Basis von vue-final-modal, REST-Endpunkte für Registrierung und Login sowie einfachen Passwortvergleich auf der Serverseite. Nach erfolgreicher Authentifizierung erfolgt eine automatische Weiterleitung.

Gegenstände-Verwaltung

Die Hauptansicht dient zur Verwaltung eigener Gegenstände. Benutzer können neue Gegenstände mit Name und optionaler Beschreibung hinzufügen. Jeder neu angelegte Gegenstand wird sofort gespeichert und erscheint mit dem Status „Verfügbar“. Alle eigenen Gegenstände werden als Karten in einem Grid angezeigt und enthalten einen Namen, Beschreibung, einen farblich markierten Status sowie gegebenenfalls die Ausleiher-ID. Gegenstände können jederzeit gelöscht werden. Zusätzlich gibt es einen Bereich für ausgeliehene Gegenstände. Dort werden alle Objekte angezeigt, die der Benutzer aktuell ausgeliehen hat. Mit einem Klick lassen sich diese zurückgeben, wodurch sie wieder als verfügbar markiert werden.

Ausleihen-Funktion

In der Ausleihen-Ansicht werden alle Gegenstände angezeigt, die nicht dem aktuellen Benutzer gehören und als verfügbar markiert sind. Zum Ausleihen genügt ein Klick auf den Ausleihen-Button. Der Status wird auf „Ausgeliehen“ gesetzt, die Ausleiher-ID gespeichert, und der Gegenstand verschwindet aus der Liste der verfügbaren Items.

Benutzeroberfläche und Design

Die Oberfläche basiert auf einem modernen Design mit Vue-typischen Farben, abgerundeten Ecken, Schatten und klaren Status-Anzeigen. Ein Sticky Header sorgt für dauerhafte Sichtbarkeit der Navigation. Die Benutzeroberfläche ist vollständig responsive und mobil optimiert.

6.2 Screenflow-Beschreibung

Szenario 1: Ein neuer Benutzer registriert sich und leiht einen Gegenstand aus

Ein nicht angemeldeter Benutzer sieht die Startansicht, das Login-Modal öffnet sich automatisch. Nach der Registrierung wird er eingeloggt und zur Home-Ansicht weitergeleitet. Er kann dort eigene Gegenstände hinzufügen, anschließend zur Ausleihen-Ansicht wechseln und Gegenstände anderer Nutzer ausleihen. Ein ausgeliehener Gegenstand erscheint schließlich in der entsprechenden Sektion der Home-Ansicht, wo er ihn später zurückgeben kann.

Szenario 2: Ein bestehender Benutzer verwaltet seine Gegenstände

Nach dem Login sieht der Benutzer alle eigenen Gegenstände in Kartenform. Gegenstände können gelöscht werden. Ausgeliehene Gegenstände können über die Rückgabefunktion wieder freigegeben werden.

Szenario 3: Fehlerbehandlung

Bei einem falschen Passwort erscheint im Login-Modal eine Fehlermeldung. Bei bereits existierenden Benutzernamen meldet die Registrierung einen Fehler. Nicht angemeldete Benutzer erhalten Hinweise, dass sie sich einloggen müssen.

6.3 Beurteilung des Ergebnisses

Stärken

Die gesamte Kernfunktionalität der Leihplattform ist vollständig implementiert. Das Projekt nutzt einen modernen Technologiesatz, und das Design ist benutzerfreundlich und übersichtlich. Die Anwendung reagiert direkt auf Änderungen und bietet eine gut strukturierte Codebasis mit klarer Trennung der Bereiche.

Verbesserungspotenziale

Sicherheitsaspekte wie tokenbasierte Authentifizierung fehlen. Es gibt keine Bestätigungsdialoge, keine Suchfunktionen, keine Historie und weitere Komfortfunktionen. Auch im technischen Bereich gibt es noch Potenzial für automatisierte Fehlerbehandlung, Tests, Ladeindikatoren und robustere Datenschemata.

Gesamtbewertung

Das Projekt erfüllt seine Ziele vollständig und stellt eine funktionale, moderne Webanwendung dar. Die Kombination aus klarer UI, vollständiger Kernfunktionalität und strukturierter Implementierung bietet ein solides Grunderlebnis für Nutzer in dieser kleinen Webanwendung. Für eine Produktionsumgebung wären erweiterte Sicherheitsmaßnahmen, bessere Fehlerbehandlung und zusätzliche Funktionen empfehlenswert. Insgesamt bietet die Anwendung eine starke Grundlage für eine weiter ausbaubare Leihplattform.

7. Reflexion

7.1 Herausforderungen

Zu Beginn waren insbesondere die Arbeit mit Deno KV sowie das Middleware-Konzept in Oak ungewohnt, da beide zwar in einem gewissen Rahmen dokumentiert sind, aber teilweise nicht so greifbar gemacht wurden.

Meine iterative Vorgehensweise schien mir zu Beginn als geeignet, um besser in das Projekt einsteigen zu können. Das hatte bis zur Verknüpfung mit dem Backend auch funktioniert, stellte mich aber im späteren Verlauf vor neue Herausforderungen. Ich musste die Komponenten häufig überarbeiten und die Struktur des Codes anpassen, was ich durch eine bessere Vorarbeit verhindern hätte können.

Durch kleine experimentelle Skripte, Community-Beiträge und durchgehende Verbesserungen ließen sich diese Herausforderungen erfolgreich lösen.

7.2 Unterstützung

Offizielle Dokumentationen und Foren stellten wichtige Grundlagen bereit und halfen mir dabei, technische Unsicherheiten zu klären.

Um hier ganz transparent zu sein, möchte ich anmerken, dass ich bei der Arbeit mit Deno KV auch auf das KI-Tool ChatGPT zurückgegriffen habe, da mir die Dokumentationen teilweise nicht vollständig helfen konnten.

7.3 Lernerfolge / Fazit

Ich konnte wertvolle Erfahrungen im Zusammenspiel moderner Webtechnologien sammeln, insbesondere in der Kombination aus Deno-Backend und Vue-Frontend.
Die Bedeutung guter Vorarbeit, sauberer Fehlerbehandlung, regelmäßiger Repository-Einträge und reaktiver UI-Patterns wurde mir in diesem Projekt besonders bewusst.
Durch iterative Entwicklungsschritte konnte ich den gesamten Entwicklungsprozess deutlich strukturierter gestalten als in früheren Projekten.

Ich habe auch festgestellt, dass Fehler in einem Projekt, das mir wirklich Freude macht, nicht entmutigen, sondern mich vielmehr dazu motivieren, dranzubleiben.

Grundsätzlich bin ich sehr zufrieden mit meinem Ergebnis. Ich habe alle Anforderungen meiner Meinung nach erfüllen können und habe diese in einer einfachen und optisch sauberen Anwendung recht gut eingebaut. Ich hätte gerne noch weitere Funktionen eingebaut, wie eine Chatbox bei den einzelnen Items, wo Nutzer ähnlich wie bei Ebay Kleinanzeigen in Kontakt treten können, oder eine Reservierung für aktuell noch ausgeliehene Gegenstände aufgeben können. Ich möchte die Anwendung als Art Trainingsprojekt weiter ausbauen und meine Erkenntnisse aus diesem Modul festigen.

A Installationsanleitung

Eine moderne Webanwendung zur Verwaltung und zum Verleih von Gegenständen. Entwickelt mit Vue.js 3 (Frontend) und Deno mit Oak Framework (Backend).

Features

- **Benutzerauthentifizierung**: Registrierung und Login
- **Gegenstände verwalten**: Eigene Gegenstände hinzufügen, anzeigen und löschen
- **Ausleihen**: Verfügbare Gegenstände anderer Nutzer finden und ausleihen
- **Rückgabe**: Ausgeliehene Gegenstände einfach zurückgeben
- **Status-Übersicht**: Visuelle Anzeige des Verfügbarkeitsstatus

Installation mit Docker (Empfohlen)

Zunächst muss das Repository geklont und in das Projektverzeichnis gewechselt werden. Dazu werden folgende Befehle verwendet:

- git clone https://github.com/ON24FiGore/T4-LoanIt.git
- cd LoanIt

Anschließend werden die Docker-Container gestartet:

- docker-compose up -d

Dadurch werden das Backend auf Port 3000 und das Frontend auf Port 8080 automatisch bereitgestellt.

Die Anwendung ist anschließend über folgende Adressen erreichbar:

- Frontend: http://localhost:8080
- Backend API: http://localhost:3000

Manuelle Installation ohne Docker Backend (Deno)

Zur Ausführung ohne Docker muss Deno zunächst installiert werden.
Nach der Installation kann das Backend gestartet werden:

- Entwicklung:
  - deno task dev
- Alternativ manuell:
  - deno run --watch --unstable-kv --allow-net main.ts

Frontend (Vue)

Zunächst werden die benötigten Abhängigkeiten installiert:

- npm install

Optional kann eine .env-Datei im Projektroot angelegt werden, um die API-Basis-URL zu setzen:

- VITEAPIURL=http://localhost:3000

Anschließend wird der Development Server gestartet:

- npm run dev

Für einen Produktionsbuild stehen folgende Befehle zur Verfügung:

- npm run build
- npm run preview

Projektstruktur

Das Projekt ist folgendermaßen aufgebaut:

```
LoanIt/
├── src/                    Frontend (Vue.js)
│   ├── components/         Vue-Komponenten
│   ├── views/              Seiten-Komponenten
│   ├── router/             Vue Router Konfiguration
│   ├── config.js           API-Konfiguration
│   └── main.js             Entry Point
├── main.ts                 Backend Entry Point (Deno)
├── .routes.ts              API Routes
├── Dockerfile.backend      Backend Docker Image
├── Dockerfile.frontend     Frontend Docker Image
├── docker-compose.yml      Docker Compose Konfiguration
├── nginx.conf              Nginx Konfiguration (für Frontend)
```

Entwicklung

Für die Codequalität steht ein Linter zur Verfügung:

- npm run lint

Hot Reloading erfolgt sowohl im Frontend durch Vite als auch im Backend über den Deno Watch-Modus.

Datenbank

LoanIt verwendet Deno KV als persistente Datenbank.

Speicherorte:

- Docker: eigenes Docker Volume (deno-kv-data)
- Manuell: lokale Dateiablage durch Deno KV

Technologien

Frontend: Vue.js 3, Vue Router, Vue Final Modal, Vite
Backend: Deno, Oak Framework, Deno KV
Containerisierung: Docker, Docker Compose, Nginx

API-Endpunkte

Items

- GET /items
- GET /items/:id
- POST /items
- PATCH /items/:id
- DELETE /items/:id

Users

- POST /users/register
- POST /users/login

B. Benutzerdokumentation

Einführung

LoanIt ist eine Webanwendung, die es Ihnen ermöglicht, Gegenstände zu verwalten und mit anderen Nutzern zu teilen. Sie können Ihre eigenen Gegenstände zur Verfügung stellen, Gegenstände anderer Nutzer ausleihen und Ihre Ausleihen verwalten.

Welche Aufgaben können Sie mit LoanIt lösen?

LoanIt hilft Ihnen bei folgenden Aufgaben:

1. **Gegenstände verwalten**: Erstellen Sie eine Übersicht Ihrer eigenen Gegenstände, die Sie verleihen möchten
2. **Gegenstände ausleihen**: Finden und leihen Sie verfügbare Gegenstände von anderen Nutzern aus
3. **Ausleihen verwalten**: Behalten Sie den Überblick über ausgeliehene Gegenstände und geben Sie diese zurück
4. **Status überwachen**: Sehen Sie auf einen Blick, welche Ihrer Gegenstände verfügbar sind und welche gerade ausgeliehen wurden

Erste Schritte

Voraussetzungen

- Ein moderner Webbrowser (Chrome, Firefox, Edge, Safari)
- Eine aktive Internetverbindung
- Der Backend-Server muss laufen

Wichtig: Um alle Funktionen zu testen, wird die Erstellung von **zwei** Nutzern empfohlen. Da die Anwendung keine öffentliche Plattform darstellt, müssen andere Nutzer simuliert werden.

Zugang zur Anwendung

Öffnen Sie die LoanIt-Anwendung in Ihrem Webbrowser. Die Anwendung öffnet sich automatisch mit einem Login-Feld.

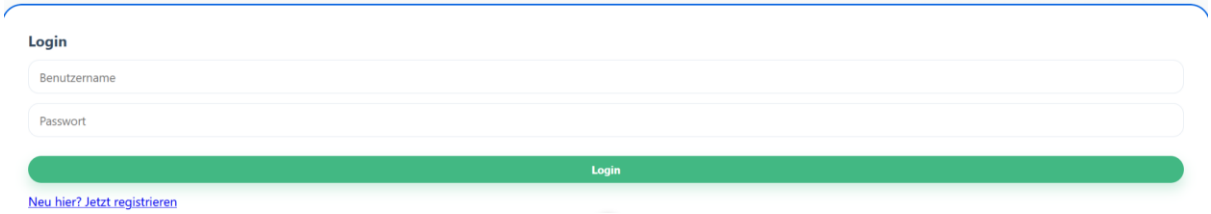

Abb. 1 Login Modal

Registrierung

Wenn Sie LoanIt zum ersten Mal nutzen, müssen Sie sich registrieren:

1. **Login-Modal öffnen**
    - Das Login-Modal öffnet sich automatisch, wenn Sie nicht angemeldet sind
    - Alternativ klicken Sie auf den Button "Login / Registrieren" im Header

Abb. 2 Login / Registrierung

2. **Zur Registrierung wechseln**

    - Klicken Sie auf den Link "Neu hier? Jetzt registrieren" im Login-Modal
    - Das Modal wechselt zur Registrierungsansicht

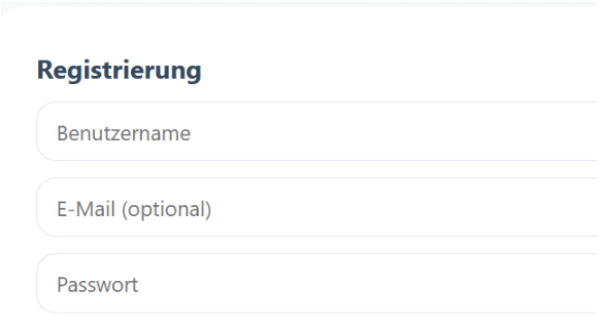

Abb. 3 Registrierung Modal

Daten eingeben

- **Benutzername**: Wählen Sie einen eindeutigen Benutzernamen (Pflichtfeld)
- **E-Mail**: Geben Sie optional Ihre E-Mail-Adresse ein
- **Passwort**: Wählen Sie ein Passwort (Pflichtfeld)

Registrierung abschließen

- Klicken Sie auf den Button "Registrieren"
- Bei erfolgreicher Registrierung schließt sich das Modal automatisch
- Die Seite lädt neu und Sie sind angemeldet

Hinweis: Falls der Benutzername bereits existiert, erhalten Sie eine Fehlermeldung. Wählen Sie dann einen anderen Benutzernamen.

Abmelden (Logout)

Um sich abzumelden:

1. Klicken Sie auf den "Logout"-Button im Header (oben rechts)
2. Die Seite lädt neu und Sie werden abgemeldet
3. Das Login-Modal öffnet sich automatisch wieder

Abb. 4 Logout Button

Gegenstände verwalten (Home-Ansicht)

Die Home-Ansicht ist Ihr zentraler Bereich zur Verwaltung Ihrer eigenen Gegenstände.

Navigation zur Home-Ansicht

- Klicken Sie auf "Home" in der Navigation im Header
- Die Home-Ansicht ist auch die Standardansicht nach dem Login

Neuen Gegenstand hinzufügen

1. **Eingabefelder finden**
    - Scrollen Sie zur Sektion "Meine Gegenstände"
    - Sie sehen eine weiße Karte mit Eingabefeldern
2. **Daten eingeben**
    - **Name des Gegenstands**: Geben Sie den Namen ein (z.B. "Bohrmaschine", "Fahrrad", "Bücher")
    - **Beschreibung**: Geben Sie optional weitere Details ein

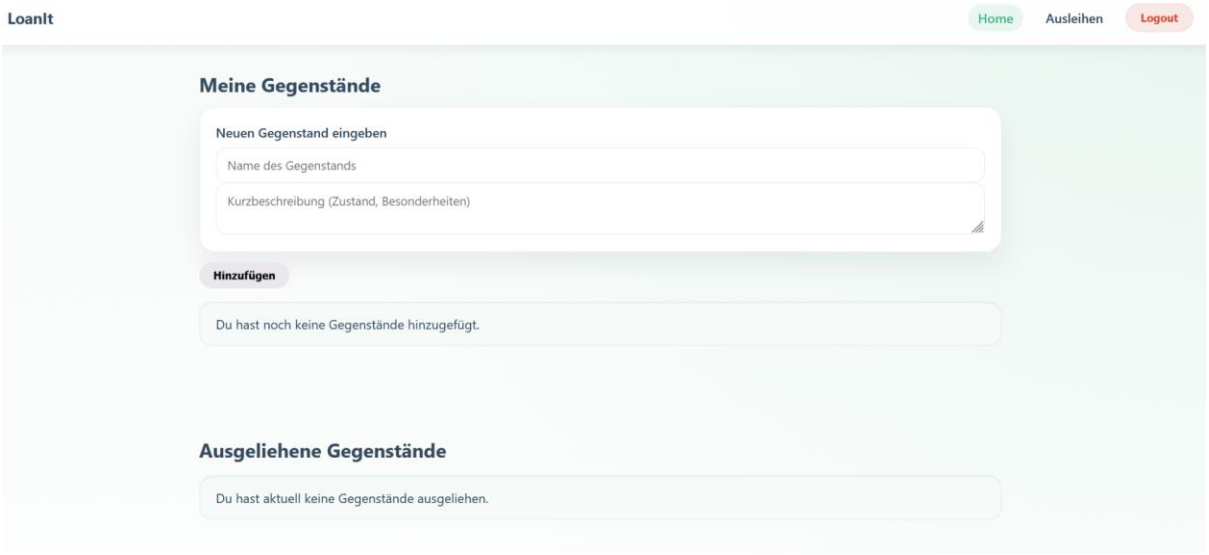

Abb. 5 HomeView

Gegenstand speichern

- Klicken Sie auf den Button "Hinzufügen"
- Der Gegenstand wird sofort in Ihrer Liste angezeigt
- Die Eingabefelder werden automatisch geleert

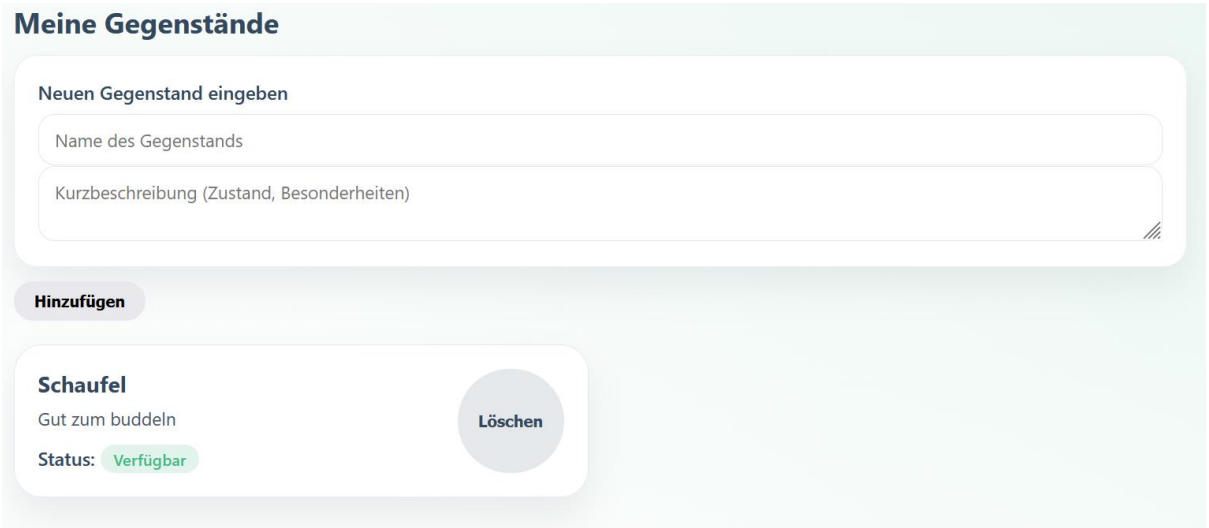

Abb. 6 Neues Item

Hinweis: Der Name ist ein Pflichtfeld. Ohne Namen wird der Gegenstand nicht hinzugefügt.

Eigene Gegenstände anzeigen

Alle Ihre Gegenstände werden in einer übersichtlichen Kartenansicht angezeigt:

Informationen pro Gegenstand:

- **Name**: Der Name des Gegenstands
- **Beschreibung**: Falls vorhanden, wird die Beschreibung unter dem Namen angezeigt
- **Status-Badge**:
    - **Grünes Badge "Verfügbar"**: Der Gegenstand kann ausgeliehen werden
    - **Rotes Badge "Ausgeliehen"**: Der Gegenstand ist aktuell ausgeliehen

- **Ausleiher-Information**: Wenn ausgeliehen, wird die Benutzer-ID des Ausleihers angezeigt

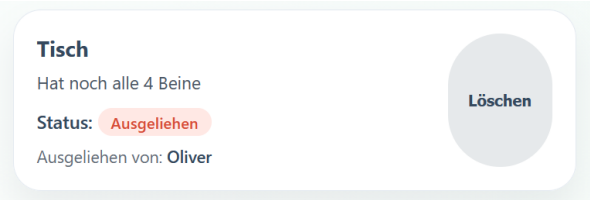

Abb. 7 Item Informationen

Gegenstand löschen

Wenn Sie einen Gegenstand nicht mehr verwalten möchten:

1. Finden Sie die entsprechende Karte in Ihrer Liste
2. Klicken Sie auf den Button "Löschen"
3. Der Gegenstand wird sofort aus der Liste entfernt

Wichtiger Hinweis: Das Löschen erfolgt ohne Bestätigungsabfrage. Stellen Sie sicher, dass Sie den richtigen Gegenstand löschen möchten.

Ausgeliehene Gegenstände verwalten

In der Home-Ansicht finden Sie weiter unten eine separate Sektion "Ausgeliehene Gegenstände":

Diese Sektion zeigt:

- Alle Gegenstände, die Sie selbst von anderen Nutzern ausgeliehen haben
- Gleiche Kartenansicht wie bei eigenen Gegenständen
- Status-Anzeige zeigt "Ausgeliehen" (rot)

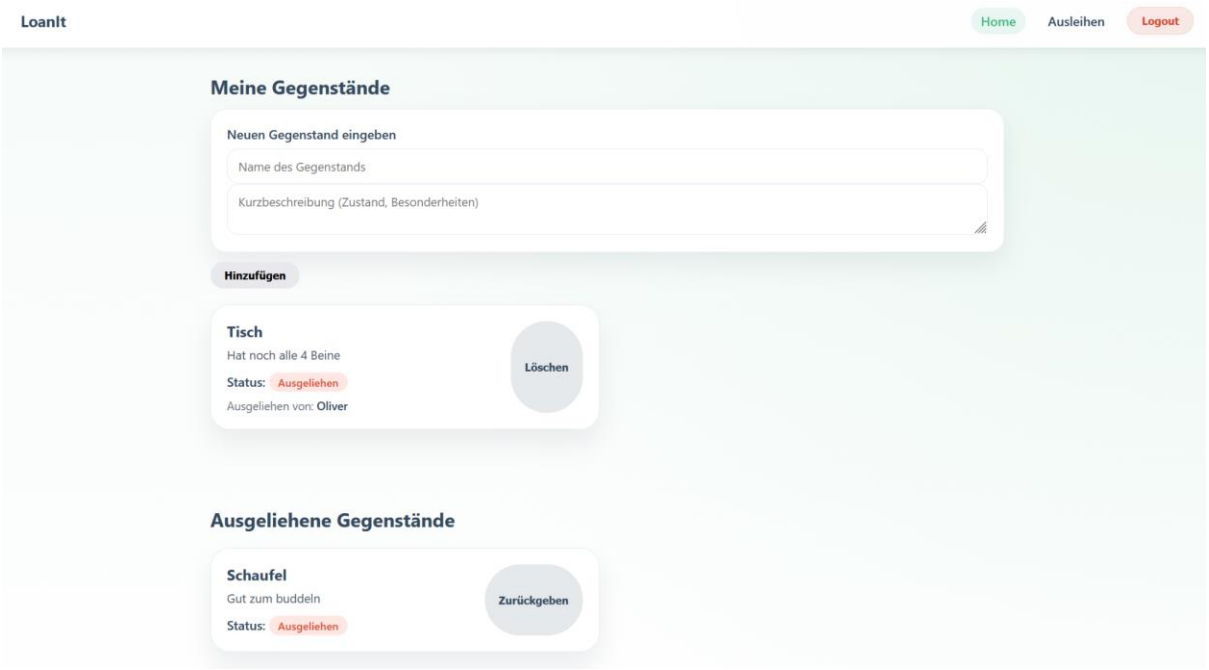

Abb. 8 Zurückgeben / Ausleihen

Ausgeliehenen Gegenstand zurückgeben

Wenn Sie einen ausgeliehenen Gegenstand zurückgeben möchten:

1. Scrollen Sie zur Sektion "Ausgeliehene Gegenstände"
2. Finden Sie den entsprechenden Gegenstand
3. Klicken Sie auf den Button "Zurückgeben"
4. Der Gegenstand wird aus dieser Liste entfernt
5. Der Status des Gegenstands beim Besitzer wird automatisch auf "Verfügbar" gesetzt

Gegenstände ausleihen (Ausleihen-Ansicht)

Die Ausleihen-Ansicht ermöglicht es Ihnen, verfügbare Gegenstände anderer Nutzer zu finden und auszuleihen.

Navigation zur Ausleihen-Ansicht

- Klicken Sie auf "Ausleihen" in der Navigation im Header

Verfügbare Gegenstände anzeigen

In der Ausleihen-Ansicht sehen Sie:

- **Überschrift**: "Verfügbare Gegenstände"
- **Gefilterte Liste**: Es werden nur Gegenstände angezeigt, die:
  - Nicht Ihnen gehören
  - Den Status "Verfügbar" haben

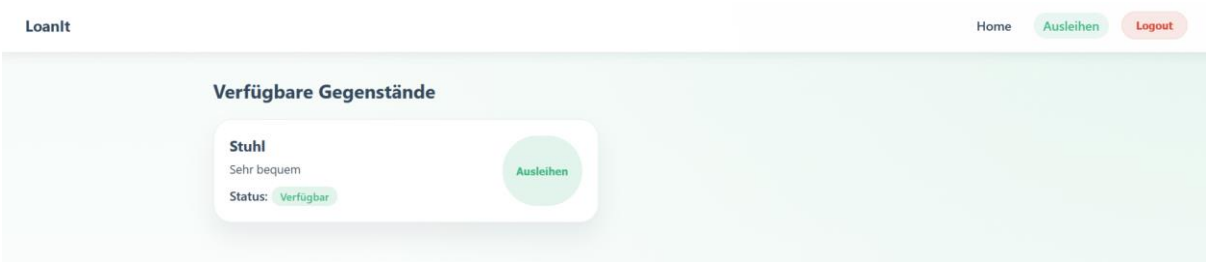

Abb. 9 Verfügbare Gegenstände

Gegenstand ausleihen

So leihen Sie einen Gegenstand aus:

1. **Gegenstand finden**
    - Durchsuchen Sie die Liste der verfügbaren Gegenstände
    - Lesen Sie Name und Beschreibung, um den passenden Gegenstand zu finden

2. **Ausleihen**
    - Klicken Sie auf den Button "Ausleihen" bei dem gewünschten Gegenstand
    - Der Gegenstand wird sofort ausgeliehen
    - Er verschwindet aus der Liste der verfügbaren Gegenstände

3. **Bestätigung**
    - Der Gegenstand erscheint nun in Ihrer "Ausgeliehene Gegenstände"-Sektion auf der Home-Ansicht
    - Der Besitzer sieht den Status als "Ausgeliehen" und Ihre Benutzer-ID als Ausleiher

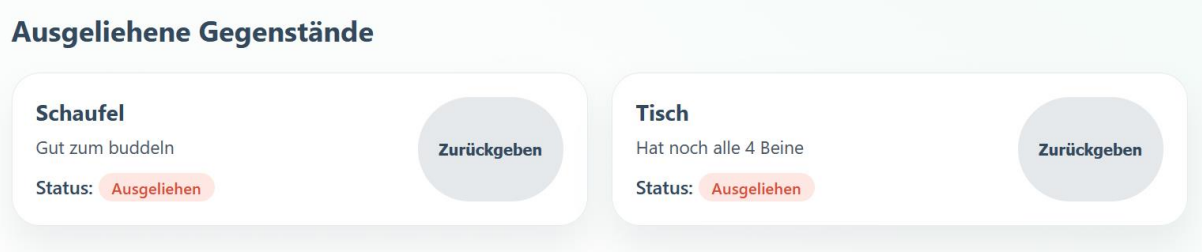

Abb. 10 Ausgeliehene Gegenstände

Wie gebe ich einen Gegenstand zurück?

1. Gehen Sie zur Home-Ansicht
2. Scrollen Sie zur Sektion "Ausgeliehene Gegenstände"
3. Klicken Sie auf "Zurückgeben" bei dem entsprechenden Gegenstand

Quellenverzeichnis

Vue.js (2025): *Vue.js 3 Dokumentation*. Verfügbar unter: https://vuejs.de/guide/introduction.html (Zugegriffen: 10.11.2025).

Vue.js (2025): *Vue.js 3 Tutorial für Anfänger*. Verfügbar unter: https://vuejs.de/tutorial/ (Zugegriffen: 10.11.2025).

Vue.js (2025): Vue.js 3 API Reference. Verfügbar unter: https://vuejs.org/api/ (Zugegriffen: 11.11.2025).

Vue.js (2025): State Management Guide. Verfügbar unter: https://vuejs.org/guide/scaling-up/state-management.html (Zugegriffen: 11.11.2025).

Vue.js (2025): Vue.js 3 API Reference. Verfügbar unter: https://vuejs.org/api/ (Zugegriffen: 11.11.2025).

Vue.js (2025): State Management Guide. Verfügbar unter: https://vuejs.org/guide/scaling-up/state-management.html (Zugegriffen: 11.11.2025).

Mozilla (2025): LocalStorage API – Beispiel. Verfügbar unter: https://developer.mozilla.org/de/docs/Web/API/Window/localStorage (Zugegriffen: 12.11.2025).

JavaScript.info (2025): Promises – Grundlagen. Verfügbar unter: https://javascript.info/promise-basics (Zugegriffen: 12.11.2025).

Vue Router (2025): Vue Router 4 Guide. Verfügbar unter: https://router.vuejs.org/guide/ (Zugegriffen: 15.11.2025).

Vite (2025): Vite Guide. Verfügbar unter: https://vite.dev/guide/ (Zugegriffen: 15.11.2025).

MDN (2025): CORS Explained. Verfügbar unter: https://developer.mozilla.org/de/docs/Web/HTTP/CORS (Zugegriffen: 15.11.2025).

Vue Final Modal (2025): Vue Final Modal Dokumentation. Verfügbar unter: https://vue-final-modal.org/ (Zugegriffen: 16.11.2025).

Oak/Telerik (2025): Building RESTful APIs with Deno & Oak. Verfügbar unter: https://www.telerik.com/blogs/building-restful-apis-deno-oak (Zugegriffen: 16.11.2025).
Async/Await Tutorial (2025): Async/Await – JavaScript Info. Verfügbar unter: https://javascript.info/async-await (Zugegriffen: 16.11.2025).
Deno (2025): Deno API Reference. Verfügbar unter: https://deno.land/api (Zugegriffen: 18.11.2025).
Deno (2025): Deno KV Manual. Verfügbar unter: https://deno.land/manual/runtime/kv (Zugegriffen: 18.11.2025).
Joi (2025): Joi Dokumentation. Verfügbar unter: https://joi.dev/api/ (Zugegriffen: 18.11.2025).
Deno (2025): Deno KV Dokumentation. Verfügbar unter: https://deno.land/api?unstable=&s=Deno.Kv (Zugegriffen: 19.11.2025).
Deno (2025): Deno KV Blog Post. Verfügbar unter: https://deno.com/blog/kv (Zugegriffen: 19.11.2025).
Vue.js (2025): Composition API Guide. Verfügbar unter: https://vuejs.org/guide/extras/composition-api-faq.html (Zugegriffen: 19.11.2025).
Vue.js RFC (2025): Composition API RFC 0013. Verfügbar unter: https://github.com/vuejs/rfcs/blob/master/active-rfcs/0013-composition-api.md (Zugegriffen: 19.11.2025).
Vue Router (2025): Vue Router 4 Migration Guide. Verfügbar unter: https://router.vuejs.org/guide/migration/ (Zugegriffen: 19.11.2025).

VS Code Extensions

[Deno Extension]
(https://marketplace.visualstudio.com/items?itemName=denoland.vscode-deno)
[ESLint Extension]
(https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)
[Vue](https://marketplace.visualstudio.com/items?itemName=Vue.volar)

Selbstständigkeitserklärung: Nachweis KI-Nutzung

Selbständigkeitserklärung und Erklärung zum Nachweis der Verwendung von KI-Systemen bei Erstellung einer wissenschaftlichen Arbeit

(einzufügen nach allen Verzeichnissen der Arbeit)

Gorell, Finn	5538099
Nachname, Vorname	Matrikelnummer

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: (…) selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fas- sung übereinstimmt.

Ich erkläre zudem, dass ich *(Zutreffendes ankreuzen)*

☐ generative Systeme (Text-, Bild-, Codeerstellung etc.), wie beispielsweise den text- basierten Chatbot ChatGPT von OpenAI, bei Erstellung der vorliegenden wissen- schaftlichen Arbeit **nicht** verwendet habe.

☒ generative Systeme (Text-, Bild-, Codeerstellung etc.) bei Erstellung der vorliegenden wissenschaftlichen Arbeit in der nachfolgend beschriebenen Art und dem nach- folgend beschriebenen Umfang verwendet habe (Bitte in der nachfolgenden Tabelle näher konkretisieren; Hinweis: bei Unsicherheiten, ob und wie ein verwendetes KI- System anzugeben ist, stimmen Sie sich mit Ihrer/m Betreuer/in ab). Auf Wunsch des Betreuers/der Betreuerin wurden die verwendeten Prompt-Anfragen im Anhang auf- geführt.

Mosbach, 22.11.2025	Finn Gorell
Ort, Datum	Unterschrift der/des Studierenden

Produktname des eingesetzten Hilfs-mittels	Link, URL	Genutzter Funktionsumfang/Beschreibung der Verwendungsweise/Sprache
ChatGPT	https://chatgpt.com/	Unterstützung beim Verständnis neuer Technologien, Klärung von Deno- und Vue-Konzepten, Hilfe bei Fehlersuche und Debugging, Validierung eigener Lösungsansätze, Erklärung sicherheitsrelevanter Themen wie Authentifizierung, Begleitung beim Lernprozess ohne Übernahme der bereitgestellten Ergebnisse Erstellung der Readme.md
Cursor	http://cursor.com/	