

Inhaltsverzeichnis

Abbildungsverzeichnis	II
Tabellenverzeichnis	III
1 Einführung	1
2 Anforderungen	1
2.1 Funktionale Anforderungen an die Web-Applikation	1
2.2 Nicht-funktionale Anforderungen an die Web-Applikation	3
2.3 Zusätzliche Anforderungen	3
3 Tech Stack	4
3.1 Framework	4
3.2 Programmiersprache	5
3.3 Datenbank	6
4 Arbeitsprozess	6
4.1 Entwicklungsmodell	6
4.2 Entwicklungsumgebung	7
4.3 Versionskontrolle	7
5 Nutzerhilfe	7
5.1 Anmeldung	8
5.2 Ausleihen & Zurückgeben	10
5.3 Bestandsinformation	12
5.4 Nutzerverwaltung [A] [M]	14
5.5 Gruppen [A]	16
5.6 Sensortypen [A]	17
5.7 Dateiverwaltung [A]	19
5.8 QR-Code [A]	20
5.9 Einstellungen	20
6 Technische Umsetzung	21
6.1 Struktur der Web-Applikation	21
6.2 Datenbank	24
6.3 Cookies	27
6.4 Deployment	27
7 Diskussion und Ausblick	28
Literaturverzeichnis	i

Abbildungsverzeichnis

1	Übersicht über das Dashboard	8
2	Anmeldung - nicht angemeldet	9
3	Anmeldung - angemeldet	9
4	Übersicht: Ausleihen & Zurückgeben	10
5	Abschnitt: Ausleihen & Zurückgeben	11
6	Abschnitt: Gruppen des ausgewählten Typs	11
7	Abschnitt: Dateien	12
8	Übersicht: Bestandsinformation	13
9	Übersicht: Ausleihübersicht	14
10	Übersicht: Nutzerverwaltung	15
11	Übersicht: Gruppen	16
12	Übersicht: Sensortypen	18
13	Übersicht: Dateiverwaltung	19
14	Übersicht: QR-Code	20
15	Übersicht: Einstellungen	21

Tabellenverzeichnis

1	Übersicht über die Nutzerrollen	2
2	Packages für {shiny}	5
3	Packages für die Programmierung	5
4	Übersicht über die Ausleihübersicht	14
5	Nutzertabelle	15
6	Gruppentabelle	16
7	Typtabelle	17
8	Untertypentabelle	18
9	Dateitabelle	19
10	Die <code>.values</code> -Environment	22
11	Datenbank: Tabelle <code>circulation</code>	24
12	Datenbank: Tabelle <code>groups</code>	24
13	Datenbank: Tabelle <code>group_type</code>	24
14	Datenbank: Tabelle <code>subtype</code>	25
15	Datenbank: Tabelle <code>type</code>	25
16	Datenbank: Tabelle <code>user</code>	25
17	Beispielhafter Inhalt der Tabelle <code>circulation</code>	26
18	Einstellungen in <code>app.yml</code>	28

1 Einführung

Die Verwaltung von Messmitteln bzw. Sensoren am Fachgebiet Fahrzeugantriebe der Technischen Universität Berlin gestaltet sich schwierig. Zurzeit werden Messmittel dezentral verwaltet. Die handelnden Personen besitzen wenig Information über den Ausleihzustand einzelner Messmittel über verschiedene Projekte hinweg. Daher hat sich das Fachgebiet entschlossen, im Rahmen des Moduls “Projekt Fahrzeugantriebe” eine Web-Applikation zur Verwaltung von Messmitteln erstellen zu lassen. Diese Web-Applikation soll die Ausleihe und Rückgabe einzelner Messmittel über einen QR-Code realisieren. Darüber hinaus sollen Messmittel gruppiert, zusätzliche Informationen, wie zum Beispiel Datenblätter, bereitgestellt und der Bestand erfasst werden. Um die unterschiedlichen Verantwortlichkeiten der Akteure zu berücksichtigen, ist zudem eine rechtebasierte Nutzerverwaltung vorzusehen.

Der Inhalt dieses Berichtes umfasst die Dokumentation des Arbeitsprozesses (Kapitel 2, 3 und 4), die Beschreibung des finalen Produktes hinsichtlich Funktionalität (Kapitel 5) und technischer Umsetzung (Kapitel 6) und zeigt zusätzlich Anknüpfungspunkte für Folgeprojekte (Kapitel 7) auf. Ergebnisse werden abschließend diskutiert (Kapitel 7).

2 Anforderungen

Aus der vom Fachgebiet bereitgestellten Aufgabenstellung wurden unmittelbar die folgenden Anforderungen abgeleitet: Aufbauend auf einer Literaturrecherche soll ein Konzept zur Erstellung einer Web-Applikation entworfen werden. Die Umsetzung besteht aus der Programmierung der entworfenen Web-Applikation sowie dem Anlegen einer zugehörigen Datenbank auf dem Server des Fachgebiets. Die verwendeten Technologien sind dabei begründet frei zu wählen. Die konkreten Anforderungen an die Web-Applikation werden aus dem in der Aufgabenstellung beschriebenen Funktionstest abgeleitet:

- Erstellen von Sensoren
- Erstellen von Nutzern
- Matching unterschiedlicher Sensorarten mit jeweils einem QR-Code
- Ausleihe und Rückgabe von Sensoren
- Konsistente Datenverwaltung in Form einer Datenbank

Diese Anforderungen stellen den minimalen Satz an Anforderungen dar. Während der Bearbeitung des Projektes zeigte sich, dass zusätzliche Anforderungen notwendig sind, um einen zweckmäßigen Einsatz am Fachgebiet sicherzustellen. Die Web-Applikation soll beispielsweise sowohl von Studenten, wissenschaftlichen Mitarbeitern und speziell geschultem Personal (üblicherweise wissenschaftliche Mitarbeiter) mit unterschiedlichen Verantwortlichkeiten genutzt werden. Daher müssen die Minimalanforderungen erweitert werden. Nutzer besitzen unterschiedliche Rechte und müssen sich gegenüber der Web-Applikation authentifizieren. Der finale Satz an Anforderungen kann den folgenden Abschnitten zu funktionalen und nicht-funktionalen Anforderungen entnommen werden.

2.1 Funktionale Anforderungen an die Web-Applikation

Funktionale Anforderungen spiegeln den Funktionsumfang der Web-Applikation wider [1, S.42]. Die Vielzahl funktionaler Anforderungen bedingt ein Aufgliedern in verschiedene Funktionsbereiche, die im Folgenden detailliert beschrieben werden.

2.1.1 Sensorverwaltung

Die Sensoren am Fachgebiet stehen untereinander in Beziehung: Ein Versuchsaufbau kann sich aus verschiedenen Sensoren zusammensetzen. Dabei ist weniger der Sensor als viel mehr seine Art von Interesse. So ist es zum Beispiel unerheblich von welchem Hersteller oder aus welcher Serie ein konkreter Sensor ist, solange die gleiche Funktionalität erbracht wird. Der konkrete Sensor wird als *Untertyp* bezeichnet. Die Funktionalität, die alle *Untertypen* verbindet, spiegelt sich im *Typen* wider. Ein Versuchsaufbau ist schließlich eine *Gruppe*, die verschiedene *Typen* beinhaltet. Zusammenfassend lässt sich festhalten:

Jeder Sensor hat einen *Untertypen*. Mehrere nur geringfügig unterschiedliche *Untertypen* werden in einem *Typen* zusammengefasst. Mehrere *Typen* können Teil einer *Gruppe* sein. Ein *Typ* kann Teil mehrerer *Gruppen* sein.

Ausgehend von diesen Definitionen sind die folgenden Anforderungen zu erfüllen:

- Erstellen und Entfernen von *Gruppen*, *Typen* und *Untertypen*
- Umbenennen von *Gruppen*, *Typen* und *Untertypen*
- Bestandserfassung auf Ebene der *Untertypen*
- Matching von QR-Codes auf Ebene der *Typen*
- Verknüpfung von *Gruppen* mit *Typen* und von *Typen* mit *Untertypen*

2.1.2 Dateiverwaltung

Mithilfe der Dateiverwaltung können zusätzliche Informationen bereitgestellt werden. Die folgenden Anforderungen sind an die Dateiverwaltung gestellt:

- Hochladen, Umbenennen und Löschen von Datenblättern im PDF-Format für *Gruppen*, *Typen* und *Untertypen*
- Herunterladen von einzelnen Datenblättern als PDF-Datei oder mehreren Datenblättern komprimiert in einer zip-Datei

2.1.3 Nutzerverwaltung

Wie eingangs beschrieben, ist eine rechtebasierte Nutzerverwaltung notwendig. Dazu werden drei Rollen angelegt. Der *Benutzer* kann Sensoren ausleihen und zurückgeben. Er kann seinen Benutzernamen und sein Passwort ändern. Der *Moderator* kann zusätzlich *Benutzer* anlegen. Der *Administrator* kann zusätzlich auf die Sensorverwaltung zugreifen und Nutzer mit einer beliebigen Rolle anlegen. Um ein hohes Maß an Übersichtlichkeit für den Leser zu gewährleisten, wurde sich ferner dazu entschieden, Operationen, die ausschließlich mit erweiterten Rechten durchführbar sind, mit einem der Rolle entsprechenden Kürzel zu markieren. So sind Operationen, die insbesondere für Administratoren bzw. Moderatoren von Relevanz sind, mit [A] bzw. [M] gekennzeichnet. Eine genaue Aufschlüsselung der Rechte der drei Rollen - und somit der Anforderungen an die Nutzerverwaltung - kann Tabelle 1 entnommen werden.

Tabelle 1: Übersicht über die Nutzerrollen

Operation	[Administrator]	[Moderator]	[Benutzer]
Ausleihen & Zurückgeben	x	x	x
Erweiterte Bestandsinformation	x	x	

Operation	[Administrator]	[Moderator]	[Benutzer]
Nutzer hinzufügen / löschen	x	x	
Moderator hinzufügen / löschen	x		
Erweiterte Nutzerverwaltung	x		
QR-Code generieren	x		
Sensoren verwalten	x		
Dateien verwalten	x		
Als anderer Nutzer operieren	x		

2.1.4 Ausleihverwaltung

An die Ausleihverwaltung werden die folgenden Anforderungen gestellt:

- Ausleihe und Rückgabe von *Untertypen* beliebiger Menge innerhalb eines verfügbaren Rahmens durch Nutzer oder für einen beliebigen Nutzer durch *Administrator*
- Abschreiben von *Untertypen* durch *Administrator*
- Anzeige von Datenblättern für *Gruppen*, *Typen* und *Untertypen*

2.1.5 Bestandsinformation

Um ohne Betreten des Lagerortes ermitteln zu können, wie viele Elemente eines *Typen* oder *Untertypen* verfügbar sind oder um festzustellen, welcher Nutzer einen benötigten *Typen* ausgeliehen hat, ist eine Übersicht über den Bestand und die Ausleihhistorie zu implementieren.

2.2 Nicht-funktionale Anforderungen an die Web-Applikation

Für einen nachhaltigen Einsatz der Web-Applikation sind die folgenden nicht-funktionalen Anforderungen zu erfüllen:

- Intuitive Nutzerführung
- Konsistenz durch wiedererkennbares Layout und Design
- Performance

Aufgrund ihres nicht-funktionalen Charakters ist die Erfüllung nicht an konkrete Bedingungen geknüpft. Alle unternommenen Bestrebungen zur Erfüllung der funktionalen Anforderungen sind stets hinsichtlich der hier aufgeführten nicht-funktionalen Anforderungen zu bewerten.

2.3 Zusätzliche Anforderungen

Zusätzlich soll eine Datenbank für die konsistente Verwaltung der anzulegenden Daten genutzt werden. Die Web-Applikation und die Datenbank sollen auf einer virtuellen Maschine (*VM*), die auf einem Server des Fachgebiets abgelegt wird, betrieben werden.

2.3.1 Datenbank

Die Datenbank soll die konsistente Datenverwaltung bewerkstelligen. Sie enthält Tabellen, welche entsprechend der funktionalen Anforderungen der Web-Applikation zu gestalten und miteinander zu verknüpfen sind. Hierzu müssen sowohl ein geeignetes Datenbankmodell als auch ein konkretes Datenbankmanagementsystem ausgewählt werden.

2.3.2 Deployment

Als Deployment wird die Integration der Web-Applikation und der Datenbank in die bestehende Infrastruktur bezeichnet [2, S.207ff]. Dazu sind folgende Schritte notwendig:

- Auswahl einer Virtualisierungssoftware
- Einrichten einer VM
- Installation von Servern, Wartungssoftware und Programmiersprache
- Transfer von Datenbank und Web-Applikation auf VM

3 Tech Stack

Als Tech Stack wird die Summe der verwendeten Technologien bezeichnet. Dazu gehören zum Beispiel die Programmiersprache, das Framework für die Web-Applikation, die Datenbanksoftware, aber auch die Entwicklungsumgebung und weitere Software, die im Entwicklungsprozess verwendet wird. [3, S.61-67]

3.1 Framework

Zuallererst muss das Framework zur Erstellung der Web-Applikation gewählt werden. Dieses legt die zu verwendende Programmiersprache fest und setzt möglicherweise Restriktionen in Bezug auf weitere Software. Eine Web-Applikation zeichnet sich dadurch aus, dass sie im Webbrower ausführbar ist [4, S.115]. Der Webbrower ist in der Lage, Dateien im HTML-Format (HTML: Hyper Text Markup Language [5, S.13]) darzustellen. Das HTML-Format spezifiziert dabei ausschließlich die Struktur der Webseite [5, S.13]. Um die visuelle Erscheinung der Webseite zu beeinflussen, können Regeln in CSS-Dateien (CSS: Cascading Style Sheet) hinterlegt werden [5, S.51]. Für interaktives Verhalten existiert die Sprache JavaScript, die es ermöglicht, das HTML-Dokument dynamisch anzupassen [5, S.55].

Ein Framework zur Erstellung von Web-Applikationen bietet ein Grundgerüst für Layout sowie Funktionalität und stellt einen Server bereit. Für das Layout werden beispielsweise makroskopische Komponenten, wie Dashboards und Landing Pages, oder mikroskopische Komponenten, wie Inputs, Tabellen und Plots, bereitgestellt. Die Funktionalität wird abstrahiert und der Zustand der Web-Applikation modelliert. Der Server bearbeitet Anfragen von Clients, also Nutzern der Web-Applikation. Frameworks können in beliebigen Programmiersprachen implementiert werden, solange eine Schnittstelle zwischen der vom Framework verwendeten Sprache und einer dem Brower verständlichen Sprache existiert. Frameworks können hinsichtlich verschiedener Kriterien unterschieden werden. Backend-Frameworks integrieren neben einem Server meist auch noch Datenbanken, wohingegen Frontend-Frameworks ihren Fokus mehr auf den visuellen Part legen. Multipage-Frameworks enthalten mehrere Seiten, wohingegen Singlepage-Frameworks nur eine einzige Seite darstellen. [6, S.10-12]

Für die Bearbeitung dieses Projektes wurde das Framework Shiny gewählt, das in der Programmiersprache R implementiert ist. Hierbei handelt es sich um ein Singlepage-Framework, das als Backend den sogenannten Shiny Server enthält [7]. Maßgeblich für die Entscheidung war, dass die beiden Autoren über Erfahrung im Umgang mit R und im Speziellen mit Shiny verfügen. Darüber hinaus zeichnet sich Shiny durch folgende Eigenschaften und Vorzüge aus [7]:

- Moderne Templates

- Fokus auf Funktionalität (Erleichterter Programmierprozess)
- Reaktives Zustandsmodell
- Modularisierbarkeit
- Für Anwendungsfall ausreichende Performance
- Einfache Integration von Datenbanken

3.2 Programmiersprache

R ist eine Multiparadigmen-Programmiersprache [8]. Je nach Anwendungsfall kann somit zum Beispiel objektorientiert oder funktional programmiert werden [8]. R verfügt über einen Pool an Standardbibliotheken und kann einfach durch selbstgeschriebene und frei verfügbare Packages erweitert werden [9]. Das Comprehensive R Archive Network (CRAN) stellt eine Vielzahl von quelloffenen Bibliotheken zur Verfügung wie zum Beispiel `{shiny}`, das die Funktionalitäten des Frameworks beinhaltet [10]. Die folgenden Tabellen 2 und 3 geben Aufschluss über die im Projekt verwendeten Packages und ihren Zweck. Packages können von anderen Packages abhängen. Es wird daher darauf verzichtet auf untergeordnete Bibliotheken einzugehen.

Tabelle 2: Packages für `{shiny}`

Package	Beschreibung
<code>{bs4Dash}</code> [11]	AdminLTE-Template
<code>{DT}</code> [12]	DataTables für <code>{shiny}</code>
<code>{htmltools}</code> [13]	HTML-Repräsentation in R
<code>{rclipboard}</code> [14]	Zwischenablage
<code>{shinydisconnect}</code> [15]	Verbindungsverlustbildschirm
<code>{shinyjs}</code> [16]	Integration von Custom-JavaScript
<code>{waiter}</code> [17]	Ladebildschirm

Tabelle 3: Packages für die Programmierung

Package	Beschreibung
<code>{Cairo}</code> [18]	PDF-/PNG-/SVG-Erstellung
<code>{DBI}</code> [19]	Datenbankinterface
<code>{dplyr}</code> [20]	Datentransformationen
<code>{glue}</code> [21]	String-Erzeugung
<code>{lubridate}</code> [22]	Datumsformat
<code>{RSQLite}</code> [23]	SQLite-Datenbank
<code>{stringr}</code> [24]	String-Manipulation
<code>{tibble}</code> [25]	Tabellenformat
<code>{purrr}</code> [26]	Funktionale Programmierung
<code>{qrcode}</code> [27]	Erstellung von QR-Codes
<code>{renv}</code> [28]	Packagemanagement
<code>{yaml}</code> [29]	YAML-Dateiformat

3.3 Datenbank

Zur konsistenten Datenverwaltung wird eine Datenbank benötigt. Datenbanken sind in der Lage, Anfragen von verschiedenen Clients zu bearbeiten und dabei zu gewährleisten, dass bestimmte Regeln hinsichtlich der Datenstruktur und Ausprägung der Daten eingehalten werden [30, S.68ff]. Es existieren verschiedene Datenbankmodelle, unter anderem das Netzwerkdatenbankmodell, das objektorientierte, das hierachische oder das relationale Datenbankmodell [31]. Diese unterscheiden sich hinsichtlich der Verknüpfung der beteiligten Daten. Aufgrund der hohen Flexibilität und der weiten Verbreitung wurde das relationale Datenbankmodell ausgewählt. Dieses speichert die Daten in miteinander verknüpften Tabellen [30, S.69-76]. Die Tabellenzeilen enthalten Beobachtungen, die Tabellenspalten stellen die beobachtbaren Merkmale dar. Zur eindeutigen Identifikation erhält jede Zeile eine Identifikationsnummer. Die Spalte der Identifikationsnummern wird als Primärschlüssel (*Primary Key*) bezeichnet [30, S.73]. Um verschiedene Tabellen miteinander zu verknüpfen, werden Identifikationsnummern referenziert. Eine Spalte, die auf einen Primärschlüssel einer anderen Tabelle verweist, wird als Fremdschlüssel (*Foreign Key*) bezeichnet [30, S.70].

Es gibt eine Vielzahl verschiedener relationaler Datenbankmanagementsysteme, die sich hinsichtlich ihrer Anwendungsbereiche und Skalierbarkeit unterscheiden. Für Projekte kleinen und mittleren Umfangs (unter 100.000 Aufrufe / Tag [32]) eignet sich SQLite. Hierbei werden alle Tabellen in einer einzigen Datei mit dem Suffix *.sqlite* gespeichert. Der Zugriff auf die Datenbank erfolgt grundsätzlich über die *Structured Query Language* (SQL). Für die Programmiersprache R gibt es die Packages {DBI} und {RSQLite}, die eine direkte Schnittstelle zur Datenbank bereitstellen.

4 Arbeitsprozess

Neben der technischen Umsetzung ist insbesondere der Arbeitsprozess von herausragender Bedeutung. Durch diesen wird festgelegt, in welcher Weise die Anforderungen letztendlich umgesetzt werden. Für Projekte im Allgemeinen beinhaltet der Arbeitsprozess das Zeit-, Personal- und Aufgabenmanagement. Ein gut strukturierter Arbeitsprozess hilft schließlich dabei, Ergebnisse effizient und nachvollziehbar zu erzielen. In der Softwareentwicklung wird der Arbeitsprozess zusätzlich durch die Wahl von Entwicklungsmodell, Entwicklungsumgebung und Versionskontrolle bestimmt.

4.1 Entwicklungsmodell

In der Softwareentwicklung existiert eine Vielzahl verschiedener Modelle zur Bewältigung eines Projektes. Die klassischen Modelle (Wasserfallmodell, V-Modell) fokussieren sich darauf, Phasen sequentiell abzuarbeiten [33, S.5]. Im Kontrast dazu stehen die agilen Modelle, in denen alle Phasen wiederholt durchlaufen werden. Agile Modelle sind in der Regel deutlich flexibler, da die Anforderungen kontinuierlich angepasst werden können. Für kleine Entwicklungsteams bietet sich die Verwendung eines agilen Entwicklungsmodells auch deswegen an, weil der Koordinationsaufwand zwischen den Teilnehmern gering ist. Das Entwicklungsmodell legt nicht nur fest, wie das Projekt auf der Makroebene strukturiert ist, sondern auch wie auf der Mikroebene konkret programmiert wird. [34]

Es wurde sich dafür entschieden, das agile Modell des Pair Programming - einer Unterform des Extreme Programming - einzusetzen. Hierbei arbeiten stets zwei Programmierer (also im vorliegenden Fall alle) gemeinsam an der Erstellung von Programmcode. Vorteile hiervon liegen im stetigen

Informationsaustausch, gemeinsamer Entscheidungsfindung und geringerer Fehlerhäufigkeit. Als nachteilig wird allgemeinhin der doppelte Personalaufwand angesehen. [34]

4.2 Entwicklungsumgebung

Als Entwicklungsumgebung wird die Software bezeichnet, die zur Erstellung und Verwaltung des Programmcodes genutzt wird. Für die Programmiersprache R empfiehlt es sich, die Entwicklungsumgebung RStudio zu verwenden. Diese ermöglicht es, Projekte anzulegen, die Web-App für das Testen unmittelbar auszuführen und den Code mit Git und GitHub für die Versionsverwaltung zu integrieren. Eine interaktive Konsole und eine integrierte Hilfe erleichtern den Arbeitsprozess.

4.3 Versionskontrolle

Versionskontrolle ist aus vielerlei Gründen für den Softwareentwicklungsprozess unerlässlich. Sie ermöglicht es,

- Versionen zu verwalten,
- Bugs durch Differenzbildung zwischen verschiedenen Versionen zu finden und zu beheben sowie
- den Projektfortschritt zeitlich und inhaltlich nachzuvollziehen. [vgl. 35, S.3-18]

Für die Versionskontrolle wurde Git in Verwendung mit GitHub eingesetzt. In Git werden inkrementelle Änderungen durch *Commits* erfasst. Jeder *Commit* ist dabei mit einem Kommentar versehen. Durch *Branches* können verschiedene Personen gleichzeitig zum Projekt beitragen oder verschiedene Features gleichzeitig entwickeln. *Branches* können wieder zusammengeführt werden (*Merging*). Ein Ordner, der mit Git initialisiert wurde, wird als *Repository* bezeichnet. *Repositories* können im Internet verfügbar gemacht werden und dann auf beliebigen Computern heruntergeladen werden. [vgl. 35, S.3-18]

Die Online-Plattform GitHub erleichtert die Kollaboration über Git. Sie stellt einen Ablageort für das *Repository* bereit und unterstützt den Arbeitsprozess durch ein Ticket-System (*Issues*). In diesem können Fehler und Verbesserungsvorschläge gemeldet werden. Die *Issues* können darüber hinaus als Notizblock für geplante Features verwendet werden. [vgl. 35, S.87-104]

5 Nutzerhilfe

Die folgenden Abschnitte fassen wesentliche Merkmale der Web-Applikation zusammen und erläutern ihre Bedienung. Die technischen Details können im anschließenden Kapitel nachgeschlagen werden. Die Web-Applikation ist ein sogenanntes Dashboard. Dieses besitzt eine Sidebar, eine Navbar und einen Body, vergleiche Abbildung 1.

Die genannten drei Elemente halten dabei wesentliche Bedienelemente bereit. Die Navbar stellt die Orientierungshilfe für den Nutzer dar. Sie ist stets visuell präsent, befindet sich oberhalb des Bodys und ermöglicht beispielsweise das Öffnen weiterer Menüs wie der Sidebar. Die Navbar enthält zudem die folgenden Elemente:

- einen Link zum GitHub-*Repository*, das den Quellcode der Web-Applikation enthält,
- einen Knopf, mit dem eine Aktualisierung der Daten erzwungen werden kann,
- einen Schalter, mit dem zwischen Tag- und Nachtmodus gewechselt werden kann.

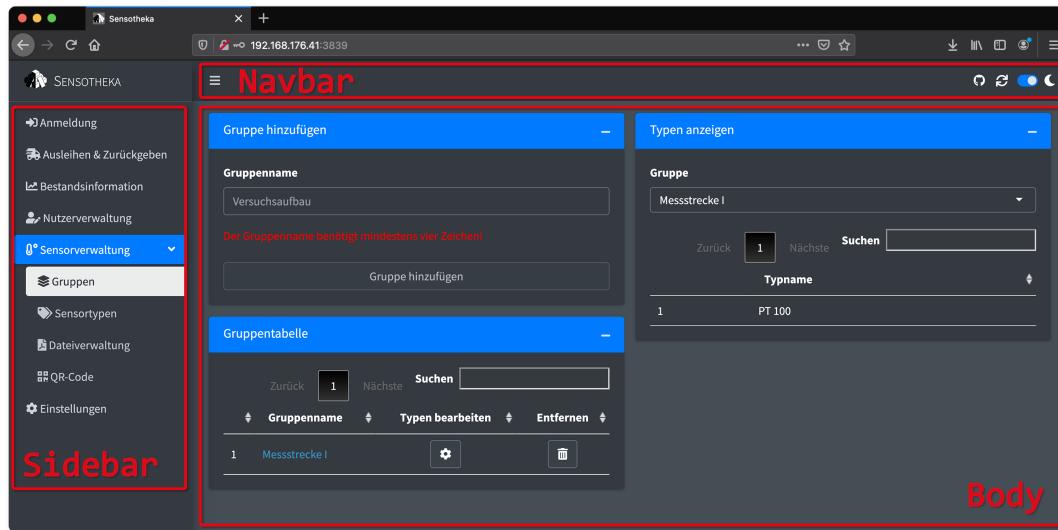


Abbildung 1: Übersicht über das Dashboard

Ausgehend von der Navbar kann, sofern nicht bereits geöffnet, das Sidebarmenü mithilfe der links befindlichen Schaltfläche geöffnet werden. In der Sidebar kann über einen Reiter die Funktionalität ausgewählt werden, die im Body dargestellt werden soll. Die Anzahl an Auswahlmöglichkeiten hängt vom Anmeldestatus und der Benutzerrolle ab. Im Folgenden gilt es daher die Reiter der Sidebar im Einzelnen näher zu betrachten.

5.1 Anmeldung

Dieser Reiter verändert sich in Abhängigkeit des Anmeldestatus.

5.1.1 Status: Nicht angemeldet

Registrierte Benutzer können sich durch Angabe ihres Benutzernamens und ihres Passwortes anmelden, vergleiche Abbildung 2. Nicht-registrierte Benutzer müssen sich von einem Moderator oder Administrator (Wissenschaftliche Mitarbeiter) registrieren lassen.

5.1.2 Status: Angemeldet

Nach erfolgreicher Anmeldung wird der Benutzer mit Informationen über sein Nutzungsverhalten versorgt, vergleiche Abbildung 3. Dargestellt werden:

- der Benutzername und der Benutzerstatus,
- die Dauer seit der momentanen Anmeldung,
- die Dauer seit der letzten Anmeldung,
- die Anzahl der Anmeldungen.

Benutzer können sich zudem abmelden.

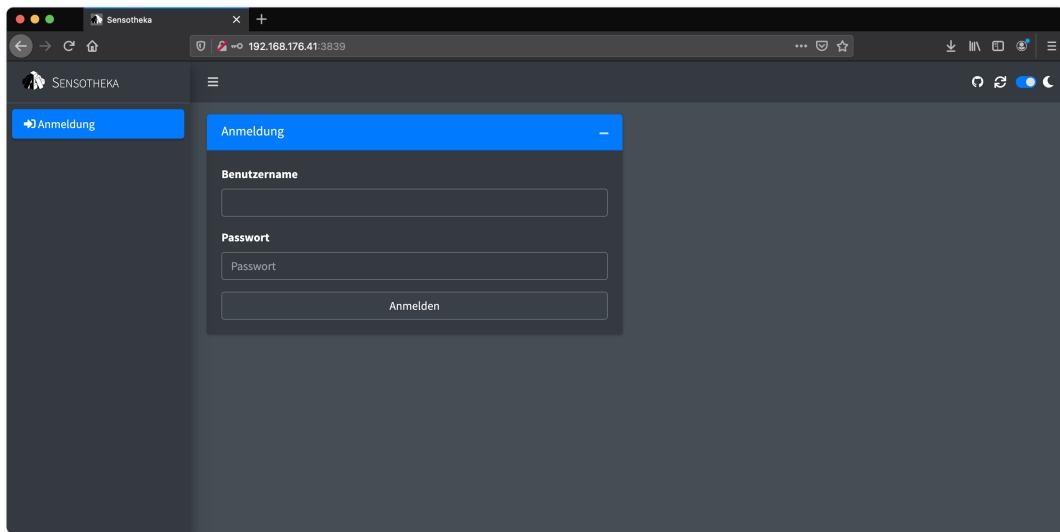


Abbildung 2: Anmeldung - nicht angemeldet

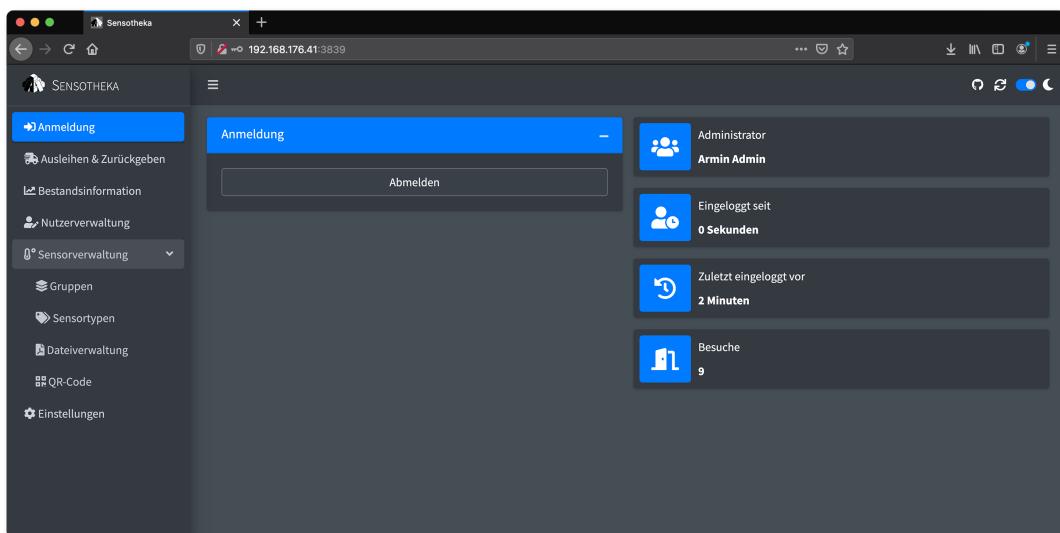


Abbildung 3: Anmeldung - angemeldet

5.2 Ausleihen & Zurückgeben

Die Ausleihe und Rückgabe setzt sich aus drei Abschnitten zusammen, vergleiche Abbildung 4. Im ersten Abschnitt *Ausleihen & Zurückgeben* kann die konkrete Operation vorgenommen werden. Zusätzliche Informationen stellen die Abschnitte *Gruppen des ausgewählten Typs* und *Dateien* bereit.

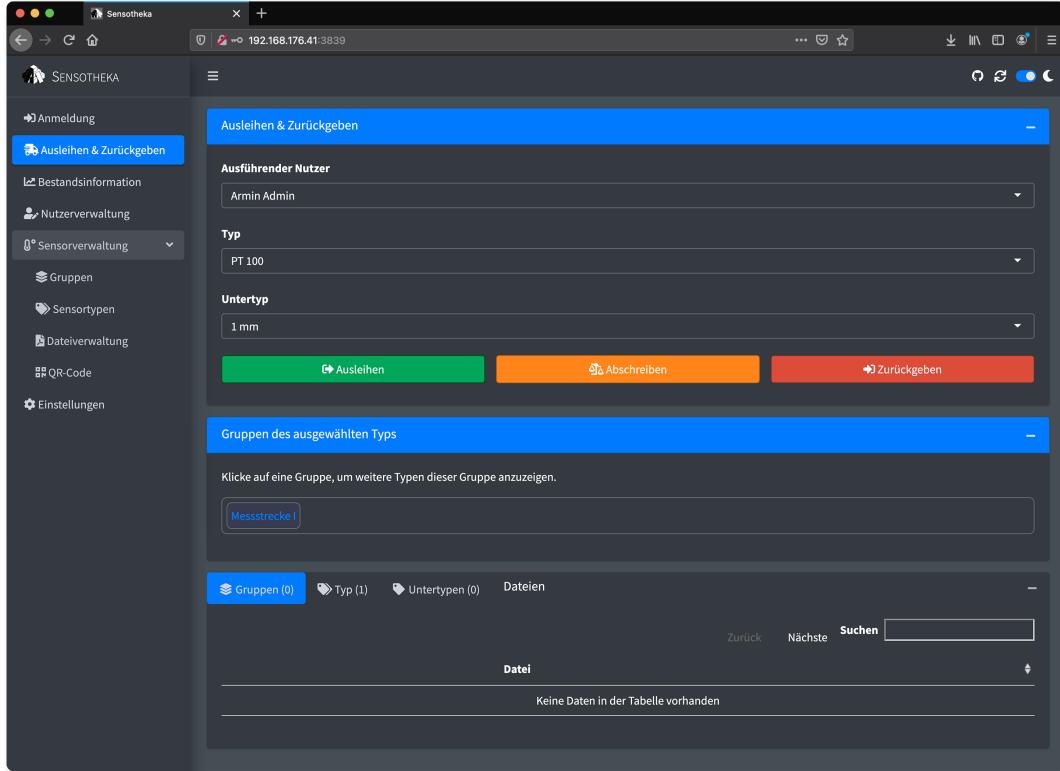


Abbildung 4: Übersicht: Ausleihen & Zurückgeben

5.2.1 Ausleihen & Zurückgeben

Zunächst muss ein *Typ* ausgewählt werden. Wenn die Web-Applikation über einen QR-Code aufgerufen wurde (siehe QR-Code), ist der zum QR-Code gehörende *Typ* bereits ausgewählt. Im nächsten Schritt muss ein zugehöriger *Untertyp* ausgewählt werden. Die konkrete Operation kann durch Klicken auf einen der verfügbaren Knöpfe angestoßen werden. Im sich darauf öffnenden Dialog muss die Menge angegeben und die Ausführung der Operation bestätigt werden. Abbildung 5 gibt einen Überblick über den Abschnitt.

Die grundlegenden Schaltflächen Ausleihen sowie Zurückgeben, stehen dabei allen Benutzerrollen zur Verfügung.

[Ausleihen] Es können bis zur maximal verfügbaren Menge Elemente ausgeliehen werden.

[Zurückgeben] Es können bis zur maximal ausgeliehenen Menge Elemente zurückgegeben werden.

Adminstatoren haben in Ergänzung dazu noch die folgenden Befugnisse:

[Abschreiben] [A] Es können bis zur maximal im Lager verfügbaren Menge Elemente abgeschrieben werden. Dies ist zum Beispiel notwendig, wenn ein Element ausfällt oder verloren geht.

Abbildung 5: Abschnitt: Ausleihen & Zurückgeben

[A] Administratoren können zusätzlich den ausführenden Nutzer auswählen, um Operationen für diesen durchzuführen. Das könnte zum Beispiel notwendig sein, wenn ein Student das Fachgebiet verlassen hat und es versäumt hat, alle ausgeliehenen Sensoren in der Web-Applikation zurückzugeben.

5.2.2 Gruppen des ausgewählten Typs

Alle *Gruppen* des ausgewählten *Typs* werden aufgelistet. Durch Klicken auf eine *Gruppe* öffnet sich ein Dialog, in dem alle *Typen* dieser *Gruppe* angezeigt werden, vergleiche Abbildung 6. Durch Klicken auf einen *Typen* wird dieser für eine weitere Operation ausgewählt. Somit können alle Elemente einer *Gruppe* (zum Beispiel ein Versuchsaufbau) komfortabel hintereinander ausgeliehen / zurückgegeben werden.

Abbildung 6: Abschnitt: Gruppen des ausgewählten Typs

5.2.3 Dateien

Hier werden alle dem ausgewählten *Typ* zugeordneten Dateien angezeigt. Dabei ist die Anzeige - wie Abbildung 7 entnommen werden kann - in drei Reiter gegliedert:

- *Gruppen*: Alle Dateien, die Informationen zu *Gruppen* des ausgewählten *Typs* enthalten
- *Typ*: Alle Dateien, die Informationen zum ausgewählten *Typ* enthalten
- *Untertypen*: Alle Dateien, die Informationen zu *Untertypen* des ausgewählten *Typs* enthalten.
Es ist zu beachten, dass die Wahl des *Untertypen* keinen Einfluss auf die angezeigten Dateien hat, da Dateien für alle *Untertypen* dargestellt werden.

Die Ziffer neben dem Reitertitel gibt Auskunft darüber, wie viele Dateien in der jeweiligen Kategorie vorhanden sind. Indem auf einen Dateinamen geklickt wird, öffnet sich das zugehörige PDF in einem separaten Browserfenster oder wird über den PDF-Viewer angezeigt. Alle Dateien einer Kategorie können in einem Archiv (.zip) heruntergeladen werden.

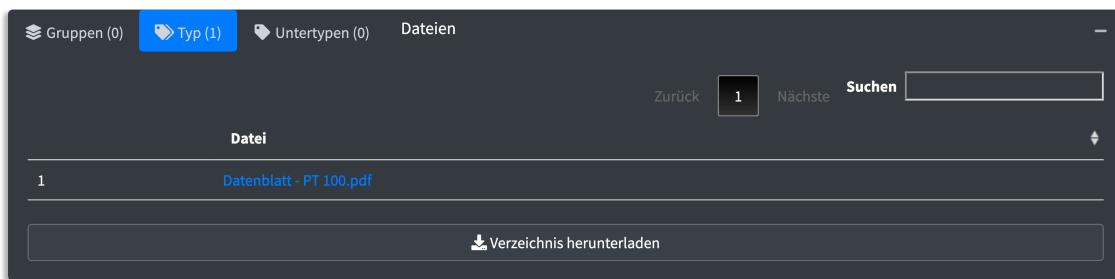


Abbildung 7: Abschnitt: Dateien

5.3 Bestandsinformation

Die Bestandsinformation gliedert sich in zwei Abschnitte, vergleiche Abbildung 8. In der *Bestandsübersicht* werden Informationen zum Lagerbestand von Sensoren dargestellt, wohingegen die *Ausleihübersicht* Informationen über ausgeliehene Sensoren enthält.

5.3.1 Bestandsübersicht

Zunächst muss ein *Typ* ausgewählt werden. Für diesen *Typen* werden tabellarisch alle *Untertypen* aufgelistet. Die Spalte *Verfügbar* enthält die gelagerte Menge, wohingegen die Spalte *Maximal verfügbar* die maximal gelagerte Menge (kein Sensor dieses *Untertypen* ausgeliehen) enthält. Über die Checkbox *Nur kritische Bestände anzeigen* kann ausgewählt werden, ob nur *Untertypen* angezeigt werden sollen, für die die verfügbare Menge kleiner als der definierte kritische Bestand ist. Der kritische Bestand eines *Untertypen* kann von einem [Administrator] im Reiter Sensortypen festgelegt werden.

5.3.2 Ausleihübersicht

Die Ausleihübersicht besteht aus vier Reitern, die unterschiedliche Fragestellungen in Bezug auf ausgeliehene Sensoren beantworten, vergleiche Tabelle 4 und Abbildung 9.

Abbildung 8: Übersicht: Bestandsinformation

Tabelle 4: Übersicht über die Ausleihübersicht

Reiter	Beschreibung
Gesamt	Übersicht über alle <i>Untertypen</i> mit einer gegenwärtig ausgeliehenen Menge größer Null.
Nach Untertyp	Übersicht über alle Nutzer, die den gewählten <i>Untertypen</i> zurzeit ausgeliehen haben. Hiermit kann herausgefunden werden, wer Ansprechpartner ist, falls alle Elemente eines <i>Untertyps</i> ausgeliehen sind.
Nach Nutzer	Übersicht über alle <i>Untertypen</i> , die der gewählte Nutzer zurzeit ausgeliehen hat.
Transaktionen	Übersicht über alle Transaktionen, die jemals im Rahmen der Sensorverwaltung stattgefunden haben. Ein Nutzer sieht ausschließlich seine eigenen Transaktionen. Ein [Administrator] sieht alle Transaktionen. Dazu gehören neben Ausleih- und Rückgabeoperationen auch Abschreibungen und sonstige Mengenänderungen.

Nummer	Typ	Untertyp	Menge	Zuletzt ausgeliehen
1	PT 100	1 mm	3	2021-03-18 16:39:50
2	PT 100	1,5 mm	20	2021-03-18 16:48:51
3	PT 100	2 mm	10	2021-03-18 16:48:32

Nummer	Nutzer	Typ	Untertyp	Datum	Menge
1	Armin Admin	Multimeter	Standard	2021-03-18 17:40:16	0
2	Bernd Benutzer	PT 100	1,5 mm	2021-03-18 16:48:51	5
3	Armin Admin	PT 100	1,5 mm	2021-03-18 16:48:42	-10
4	Modesta Moderator	PT 100	2 mm	2021-03-18 16:48:32	2
5	Modesta Moderator	PT 100	2 mm	2021-03-18 16:48:27	-12
6	Bernd Benutzer	PT 100	1,5 mm	2021-03-18 16:48:14	-15

Abbildung 9: Übersicht: Ausleihübersicht

5.4 Nutzerverwaltung [A] [M]

In der *Nutzerverwaltung* können neue Nutzer hinzugefügt und in der *Nutzertabelle* verwaltet werden, vergleiche Abbildung 10.

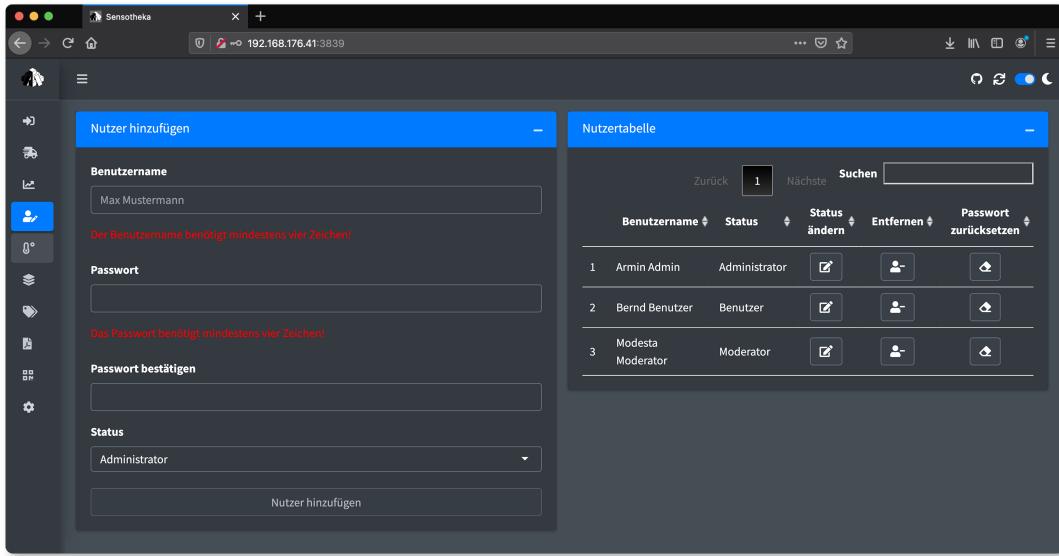


Abbildung 10: Übersicht: Nutzerverwaltung

5.4.1 Nutzer hinzufügen

Neue Nutzer können unter Angabe eines Benutzernamens und eines zur Sicherheit zweimal anzugebenden Passwortes hinzugefügt werden. Ein [Moderator] kann ausschließlich [Benutzer] hinzufügen, wohingegen ein [Administrator] Benutzer mit jeder Rolle hinzufügen kann.

5.4.2 Nutzertabelle

In der *Nutzertabelle* (vergleiche Tabelle 5) können Nutzer verwaltet werden. Neben dem Benutzernamen und dem gegenwärtigen Status gibt es drei Spalten, die Knöpfe enthalten, mit denen Eigenschaften eines Nutzers angepasst werden können.

Tabelle 5: Nutzertabelle

Spalte	Beschreibung
Status ändern [A]	Dieser Knopf öffnet einen Dialog, in dem ein neuer Status für den Nutzer ausgewählt werden kann.
Entfernen	Dieser Knopf öffnet einen Dialog, in dem bestätigt werden muss, dass der ausgewählte Nutzer gelöscht werden soll. Es ist zu beachten, dass diese Operation nicht rückgängig gemacht werden kann. Transaktionen, die der ausgewählte Nutzer vorgenommen hat, bleiben jedoch weiterhin erhalten. Es können nur Nutzer gelöscht werden, die zurzeit keine Sensoren ausgeliehen haben. Falls der Nutzer nicht in der Lage ist, die Sensoren eigenständig zurückzugeben, kann der [Administrator] die ausgeliehenen Sensoren im Reiter Ausleihen & Zurückgeben für diesen Nutzer zurückgeben. Ein [Moderator] kann nur [Benutzer] entfernen, die er selbst hinzugefügt hat.

Spalte	Beschreibung
Passwort zurücksetzen [A]	Dieser Knopf öffnet einen Dialog, in dem bestätigt werden muss, dass das Passwort des Nutzers zurückgesetzt werden soll. Das Passwort wird auf das Standardpasswort <i>1234</i> zurückgesetzt und sollte vom Nutzer sofort danach im Reiter Einstellungen geändert werden.

5.5 Gruppen [A]

Der Reiter *Gruppen* erlaubt es, neue *Gruppen* hinzuzufügen und bestehende *Gruppen* zu verwalten, vergleiche Abbildung 11.

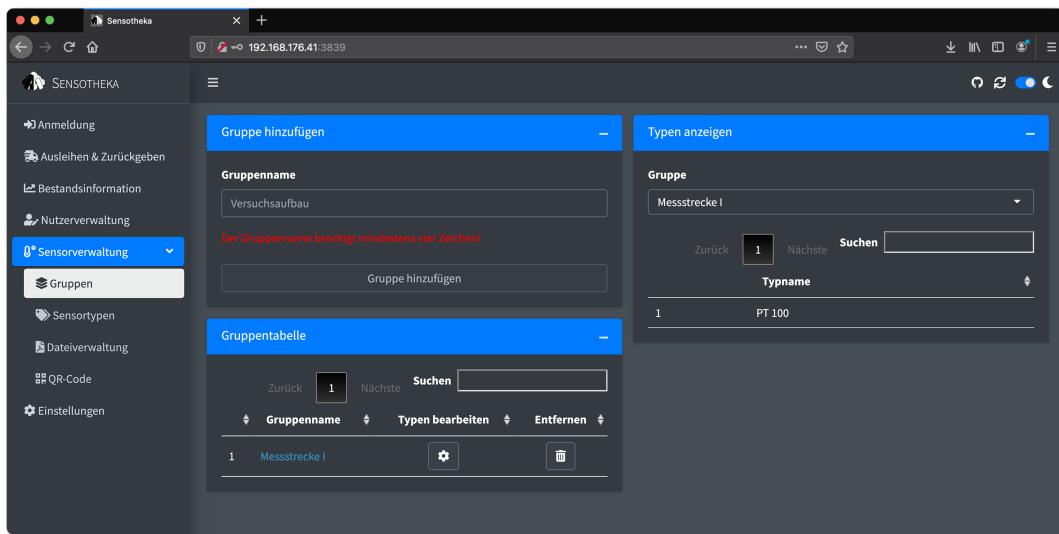


Abbildung 11: Übersicht: Gruppen

5.5.1 Gruppe hinzufügen

Eine neue *Gruppe* kann durch Angabe eines Gruppennamens hinzugefügt werden.

5.5.2 Gruppentabelle

In der *Gruppentabelle* (vergleiche Tabelle 6) können *Gruppen* bearbeitet werden.

Tabelle 6: Gruppentabelle

Spalte	Beschreibung
Gruppename	Ein Klick auf einen Gruppennamen öffnet einen Dialog, in dem der Gruppename angepasst werden kann.
Typen bearbeiten	Dieser Knopf öffnet einen Dialog, in dem die zur ausgewählten Gruppe zugehörigen <i>Typen</i> bearbeitet werden können.

Spalte	Beschreibung
Entfernen	Dieser Knopf öffnet einen Dialog, in dem bestätigt werden muss, dass die ausgewählte <i>Gruppe</i> gelöscht werden soll. Es ist zu beachten, dass diese Operation nicht rückgängig gemacht werden kann.

5.5.3 Typen anzeigen

Zunächst muss eine *Gruppe* ausgewählt werden. In der Tabelle werden alle zur ausgewählten *Gruppe* zugehörigen *Typen* dargestellt.

5.6 Sensortypen [A]

Der Reiter *Sensortypen* erlaubt es, neue *Typen* und *Untertypen* hinzuzufügen und bestehende *Typen* und *Untertypen* zu verwalten, vergleiche Abbildung 12.

5.6.1 Typ hinzufügen

Ein neuer *Typ* kann durch Angabe eines Typnamens hinzugefügt werden.

5.6.2 Typtabelle

In der *Typtabelle* (vergleiche Tabelle 7) können *Typen* bearbeitet werden.

Tabelle 7: Typtabelle

Spalte	Beschreibung
Typname	Ein Klick auf einen Typnamen öffnet einen Dialog, in dem der Typname angepasst werden kann.
Gruppen bearbeiten	Dieser Knopf öffnet einen Dialog, in dem die zum ausgewählten <i>Typ</i> zugehörigen <i>Gruppen</i> bearbeitet werden können.
Entfernen	Dieser Knopf öffnet einen Dialog, in dem bestätigt werden muss, dass der ausgewählte <i>Typ</i> gelöscht werden soll. Es ist zu beachten, dass diese Operation nicht rückgängig gemacht werden kann. Ein <i>Typ</i> kann nur gelöscht werden, wenn kein Element seiner <i>Untertypen</i> ausgeliehen ist. Transaktionen, die den ausgewählten <i>Typen</i> betreffen, bleiben weiterhin erhalten. Das Entfernen eines <i>Typen</i> schließt das Entfernen aller <i>Untertypen</i> dieses <i>Typs</i> ein.

5.6.3 Gruppen anzeigen

Zunächst muss ein *Typ* ausgewählt werden. In der Tabelle werden alle zum ausgewählten *Typen* zugehörigen *Gruppen* dargestellt.

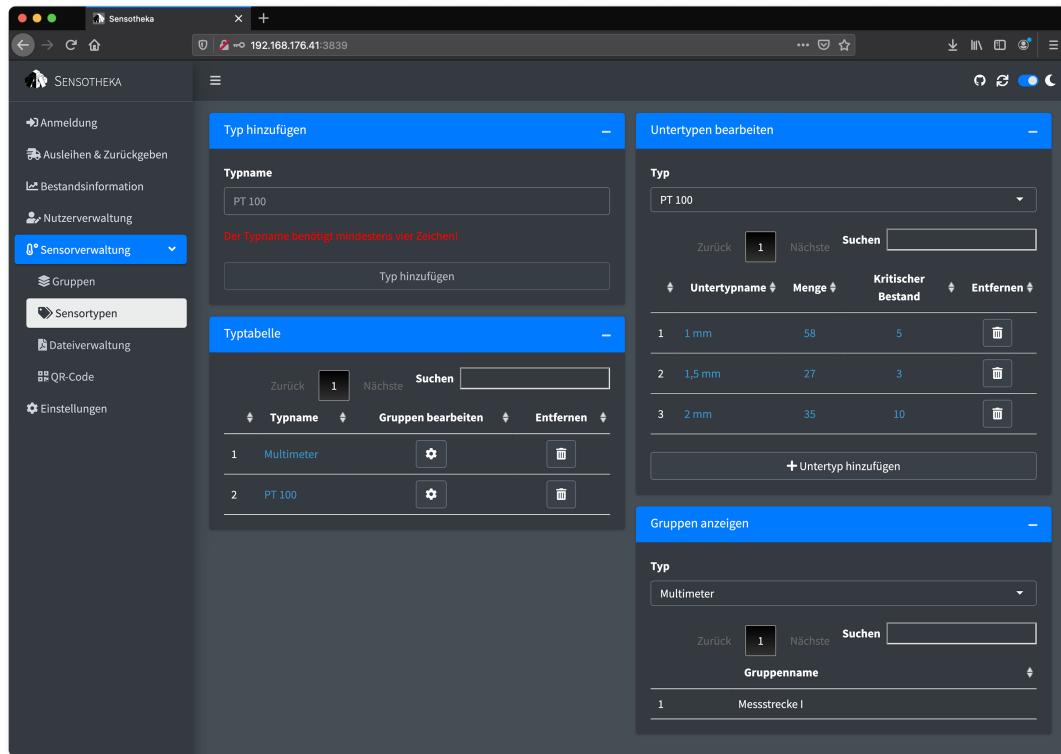


Abbildung 12: Übersicht: SensorTypen

5.6.4 Untertypen bearbeiten

Zunächst muss ein *Typ* ausgewählt werden. In der *Untertypentabelle* (vergleiche Tabelle 8) können Untertypen des ausgewählten Typen bearbeitet werden.

Tabelle 8: Untertypentabelle

Spalte	Beschreibung
Untertypname	Ein Klick auf einen Untertypnamen öffnet einen Dialog, in dem der Untertypname angepasst werden kann.
Menge	Ein Klick auf die maximal verfügbare Menge öffnet einen Dialog, in dem die maximal verfügbare Menge des <i>Untertypen</i> angepasst werden kann. Es ist zu beachten, dass die maximal verfügbare Menge nicht auf einen Wert gesetzt werden kann, der kleiner der Anzahl gegenwärtig ausgeliehener Elemente ist.
Kritischer Bestand	Ein Klick auf den kritischen Bestand öffnet einen Dialog, in dem der kritische Bestand des <i>Untertypen</i> angepasst werden kann. Der kritische Bestand kann als Filterkriterium in der <i>Ausleihübersicht</i> des Reiters Bestandsinformation verwendet werden.

Spalte	Beschreibung
Entfernen	Dieser Knopf öffnet einen Dialog, in dem bestätigt werden muss, dass der ausgewählte <i>Untertyp</i> gelöscht werden soll. Es ist zu beachten, dass diese Operation nicht rückgängig gemacht werden kann. Ein <i>Untertyp</i> kann nur gelöscht werden, wenn keine Elemente von diesem ausgeliehen sind. Transaktionen, die den ausgewählten <i>Untertypen</i> betreffen, bleiben weiterhin erhalten.

5.7 Dateiverwaltung [A]

In der *Dateiverwaltung* können PDF-Dateien als Informationsmaterial in den Reitern *Gruppen*, *Typen* und *Untertypen* hochgeladen werden, siehe Abbildung 13.

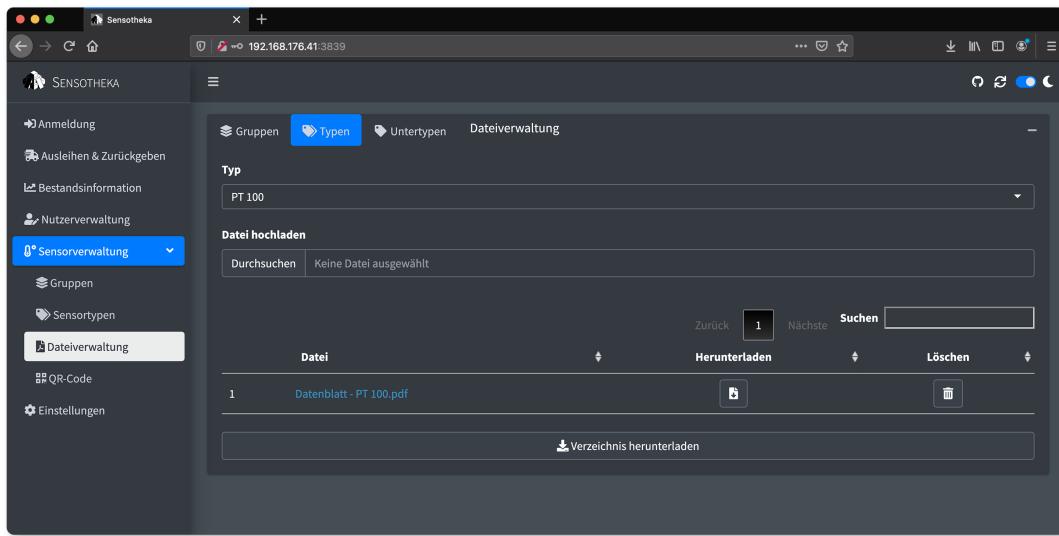


Abbildung 13: Übersicht: Dateiverwaltung

Die Reiter unterscheiden sich ausschließlich in der Auswahl des zu beschreibenden Objektes. Durch Klick auf *Datei hochladen* kann eine PDF-Datei auf dem lokalen Dateisystem ausgewählt werden. Hochgeladene Dateien werden in einer Tabelle (vergleiche Tabelle 9) angezeigt.

Tabelle 9: Dateitabelle

Spalte	Beschreibung
Datei	Ein Klick auf einen Dateinamen öffnet einen Dialog, in dem der Dateiname angepasst werden kann.
Herunterladen	Dieser Knopf lädt die Datei herunter.
Löschen	Dieser Knopf öffnet einen Dialog, in dem bestätigt werden muss, dass die ausgewählte Datei gelöscht werden soll. Es ist zu beachten, dass diese Operation nicht rückgängig gemacht werden kann.

5.8 QR-Code [A]

Zur beschleunigten Ausleihe und Rückgabe kann ein QR-Code für *Typen* erstellt werden, vergleiche Abbildung 14. Dazu muss zunächst der *Typ* und die Serverdomäne ausgewählt werden. Der erstellte QR-Code und der Link, auf den der QR-Code verweist, werden eingeblendet. Der QR-Code kann im Anschluss als PDF-, PNG- oder SVG-Datei heruntergeladen werden. Dazu kann zusätzlich die Breite und Höhe der zu erstellenden Datei in Millimetern angegeben werden.

Der ausgedruckte QR-Code kann im Lager platziert werden. Dort muss er zum Beispiel mit einem auf einem mobilen Endgerät installierten, handelsüblichen QR-Code-Reader eingescannt werden, wodurch die Web-Applikation geöffnet wird. [Benutzer] und [Moderator] werden - falls bereits angemeldet, siehe Cookies - sofort, ansonsten nachdem sie sich angemeldet haben, auf den Reiter Ausleihen & Zurückgeben weitergeleitet, wo der eingescannte *Typ* bereits vorausgewählt ist. Es muss dann nur noch der *Untertyp* ausgewählt werden.

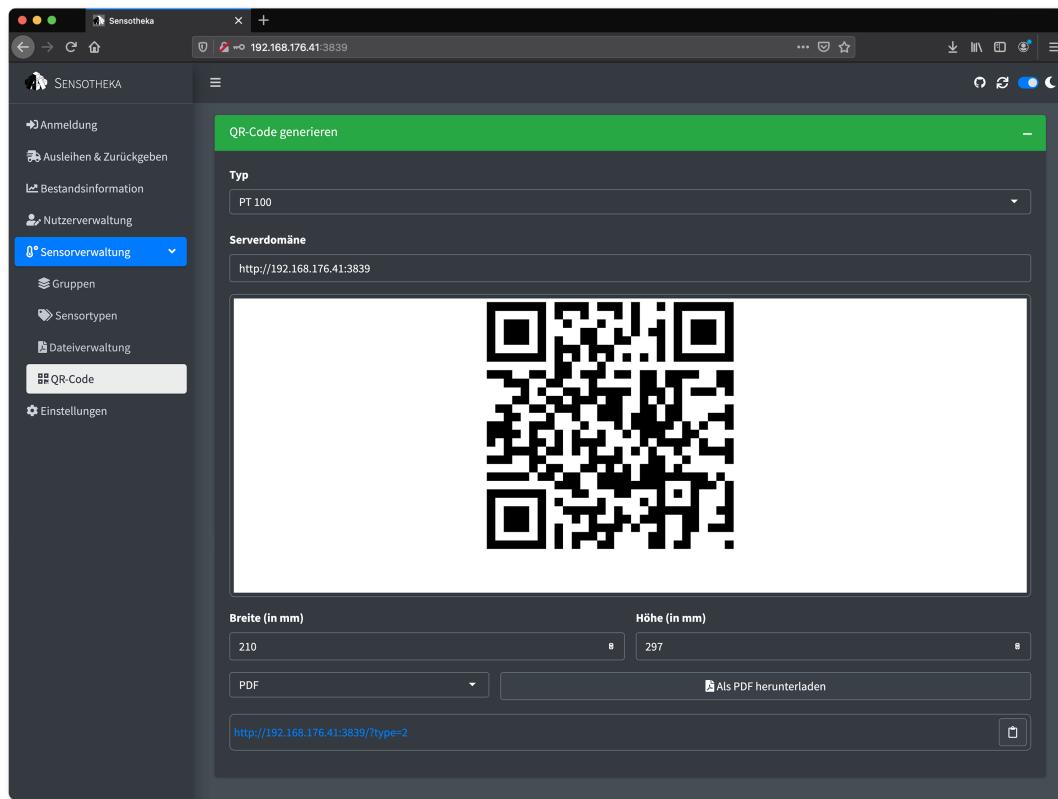


Abbildung 14: Übersicht: QR-Code

5.9 Einstellungen

In den Einstellungen können der Benutzername und das Passwort geändert werden, vergleiche Abbildung 15.

5.9.1 Benutzernamen ändern

Um den eigenen Benutzernamen zu ändern, muss ein neuer Benutzername eingegeben werden. Nach Eingabe des Passworts und Bestätigen wird der Benutzername geändert.

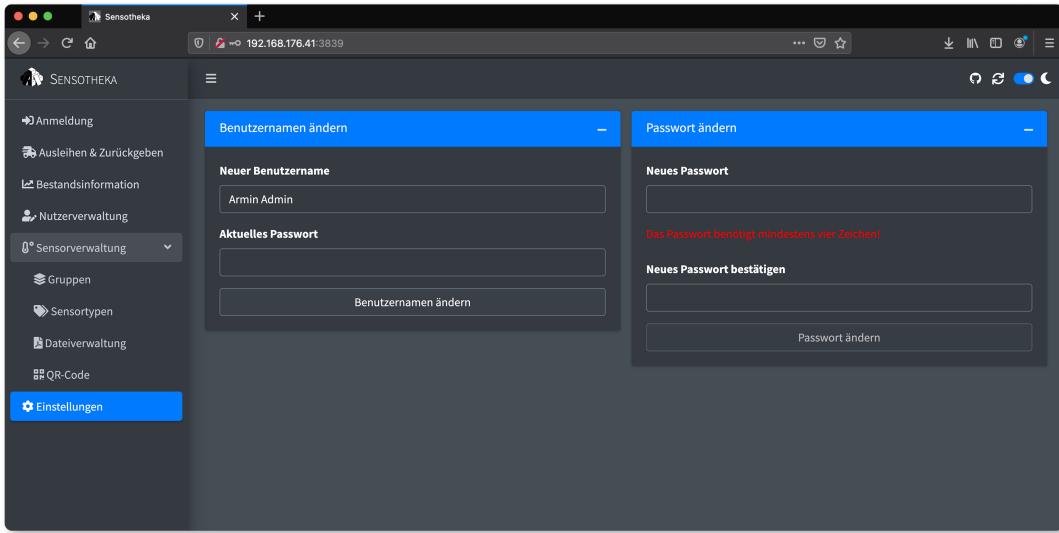


Abbildung 15: Übersicht: Einstellungen

5.9.2 Passwort ändern

Um das eigene Passwort zu ändern, muss ein neues Passwort zweimal eingegeben und bestätigt werden.

6 Technische Umsetzung

Die nachfolgenden Abschnitte erweitern die Nutzerhilfe um die Beschreibung der technischen Umsetzung ausgewählter Elemente der Web-Applikation. Die ausgewählten Elemente sollen dabei insbesondere eine Vielzahl des verwendeten Methodenspektrums darlegen, sodass eine nachhaltige Nutzung über die Laufzeit des Projektes hinaus erleichtert wird. Darüber hinaus sollen interessante Ansätze vorgestellt werden.

6.1 Struktur der Web-Applikation

Strukturell handelt es sich bei der Web-Applikation um eine Sammlung verschiedener Dateien innerhalb eines Ordners. Der Einstiegspunkt in die Web-Applikation ist die Datei `app.R`. Alle anderen Dateien werden zur Laufzeit der Web-Applikation eingebunden. Die Web-Applikation wird gestartet, indem der Shiny Server die Datei `app.R` ausführt. In `app.R` werden Initialisierungen durchgeführt und die Funktionen `ui` und `server` definiert. Die Datei endet mit dem Aufruf `shinyApp(ui, server)`, wobei die `ui`-Funktion das visuelle Layout der Web-Applikation beschreibt und die `server`-Funktion Eingaben verarbeitet. Mithilfe von sogenannten Modulen kann der Code strukturiert werden. Jedes Modul setzt sich dabei wiederum aus einer `ui`- und einer `server`-Funktion zusammen. Module bilden Teilespekte, die unter Umständen wiederverwendet werden können, ab. [36]

6.1.1 Struktur von `app.R`

Wie bereits beschrieben, ist die Datei `app.R` der Einstiegspunkt der Web-Applikation. Da sie Einfluss auf das Verhalten vieler weiterer Module hat, ist ihr Aufbau näher zu betrachten. Zur Übersicht

werden die Inhalte verkürzt dargestellt.

- Laden benötigter Packages mit `library()`. Die Mehrzahl aller Packages wird über den `::-`-Operator referenziert. Nur die Packages, die zwangsläufig mit `library()` geladen werden müssen, also Packages, die funktionsrelevanten Code mit dem Aufruf von `library()` verbinden, werden hier geladen.

```
library(shiny)
library(shinyjs)
library(dplyr)
library(qrcode)
```

- Die Funktion `source_directory()` wird manuell geladen und danach dazu verwendet, alle `.R`-Dateien in den Ordner `modules` und `db/func` einzubinden.

```
source("init/source_directory.R")

source_directory(
  path = "modules",
  encoding = "UTF-8",
  modifiedOnly = FALSE,
  chdir = TRUE,
  recursive = TRUE,
  envir = if (source_to_globalenv) globalenv() else environment()
)
```

- Die `ui`-Funktion bindet alle CSS- und JavaScript-Dateien ein und aktiviert spezielle Packages. Die Funktion `container_ui()`, die das Layout des Dashboardes festlegt, wird aufgerufen. In `container_ui()` wird für jeden Reiter eine eigene `ui`-Funktion aufgerufen.

```
# UI -----
ui <- htmltools::tagList(
  htmltools::includeScript("www/js/dark-mode.js"),
  htmltools::includeCSS("www/css/styles.css")
  container_ui(
    id = "container"
  ),
  rclipboard::rclipboardSetup(),
  shinyjs::useShinyjs(),
  waiter::use_waiter()
)
```

- In der `server`-Funktion wird die `.values`-Environment angelegt. Diese enthält die in Tabelle 10 aufgelisteten Elemente.

Tabelle 10: Die `.values`-Environment

Name	Beschreibung
<code>query\$type</code>	Aus Query-String ausgelesene Typ-ID

Name	Beschreibung
settings	Liste, die Namenslängen festlegt und Dictionaries enthält
update	Liste, die <code>reactiveVal()</code> s enthält, die bei Datenaktualisierung getriggert werden
user	Liste, die Informationen zum angemeldeten Nutzer enthält
yaml	Inhalt von <code>app.yml</code>

Diese Environment wird jeder `server`-Funktion übergeben, sodass die Werte in jeder `server`-Funktion verfügbar sind.

- In der `server`-Funktion von `app.R` wird die `server`-Funktion `container_server()` aufgerufen.

6.1.2 Ordnerstruktur

- `.gitignore` Index aller Dateien, die nicht der Versionskontrolle unterliegen
- `app.R` Einstiegspunkt in die Web-Applikation
- `app.yml` Konfigurationsdatei
- `db/`
 - `db.sqlite` Datenbank
 - `func` Funktionssammlung zur Interaktion mit der Datenbank
- `files/` Dateiensammlung für Dateiverwaltung. Enthält drei Unterordner für *Gruppen*, *Typen* und *Untertypen*. Jeder dieser Unterordner enthält für jedes Element der jeweiligen Kategorie einen Ordner mit der Identifikationsnummer des jeweiligen Elementes als Namen
- `init/source_directory.R` Hilfsfunktion, die beim Start der Web-Applikation alle `.R`-Dateien einliest
- `modules/` Shiny-Module und sonstige Funktionalitäten
 - `container.R` Definition des Dashboards
 - `dt_options.R` Optionen für `DT::datatable`
 - `object` Generalisierte Module für *Gruppen*, *Typen* und *Untertypen* (z.B. Element hinzufügen, umbenennen etc.)
 - `sidebar_menu.R` Definition der Sidebar
 - `tab_file_management` Dateiverwaltung
 - `tab_group` Gruppen
 - `tab_login` Anmeldung
 - `tab_operate` Ausleihen & Zurückgeben
 - `tab_qrcode` QR-Code
 - `tab_reporting` Bestandsinformation
 - `tab_settings` Einstellungen
 - `tab_type` Typen
 - `tab_user_management` Nutzerverwaltung
 - `utils.R` Hilfsfunktionen
- `renv`, `renv.lock` Package-Management
- `www/`
 - `css/` CSS-Dateien
 - `favicon.ico` Sensotheka-Icon

- `js/` JavaScript-Dateien

6.2 Datenbank

Die Datenbank ist eine SQLite-Datenbank. Sie liegt in `db/db.sqlite`. Der Zugriff erfolgt über das Package `{DBI}` [19].

6.2.1 Struktur

Alle Tabellen enthalten die Spalte `rowid`, die den Primärschlüssel der jeweiligen Tabelle enthält. Diese kann somit zur eindeutigen Identifikation einer Zeile verwendet werden. Die Ablage der zu speichernden Werte erfolgt in Ergänzung dazu über weitere Spalten, die in den folgenden Unterkapiteln näher beschrieben werden. Die Beschränkung [Einzigartig] bezieht sich immer nur auf die nicht gelöschten Elemente (`removed = 0`).

6.2.1.1 circulation

Tabelle 11: Datenbank: Tabelle `circulation`

Spalte	Einschränkungen	Beschreibung
<code>user_id</code>	[Fremdschlüssel]	Identifikationsnummer des ausführenden Nutzers
<code>subtype_id</code>	[Fremdschlüssel]	Identifikationsnummer des bearbeiteten <i>Untertyps</i>
<code>quantity</code>	[Nicht-negativ]	Bearbeitete Menge
<code>time</code>		Zeitpunkt der Transaktion
<code>op_type</code>	1 oder 2	Transaktionsart: 1 - Ausleihen und Zurückgeben, 2 - Bestandsänderungen

6.2.1.2 groups

Tabelle 12: Datenbank: Tabelle `groups`

Spalte	Einschränkungen	Beschreibung
<code>group_name</code>	[Einzigartig]	Gruppenname
<code>removed</code>	0 oder 1	0 - existent, 1 - gelöscht

6.2.1.3 group_type

Tabelle 13: Datenbank: Tabelle `group_type`

Spalte	Einschränkungen	Beschreibung
<code>group_id</code>	[Fremdschlüssel]	Identifikationsnummer der <i>Gruppe</i>
<code>type_id</code>	[Fremdschlüssel]	Identifikationsnummer des <i>Typs</i>

6.2.1.4 subtype

Tabelle 14: Datenbank: Tabelle **subtype**

Spalte	Einschränkungen	Beschreibung
type_id	[Fremdschlüssel]	Identifikationsnummer des <i>Typs</i>
subtype_name	[Einzigartig] innerhalb eines <i>Typs</i>	Untertypenname
quantity	[Nicht-negativ]	Bestandsmenge
removed	0 oder 1	0 - existent, 1 - gelöscht

6.2.1.5 type

Tabelle 15: Datenbank: Tabelle **type**

Spalte	Einschränkungen	Beschreibung
type_name	[Einzigartig]	Typname
removed	0 oder 1	0 - existent, 1 - gelöscht

6.2.1.6 user

Tabelle 16: Datenbank: Tabelle **user**

Spalte	Einschränkungen	Beschreibung
hash		Verhaschter Nutzername (notwendig für Cookies)
name	[Einzigartig]	Nutzername
status	admin, mod oder user	Nutzerrolle
password		Verhashtes Passwort
added_from		Nutzeridentifikationsnummer des Hinzufügenden
time_added		Zeitpunkt der Kontoeröffnung
time_current_logged		Zeitpunkt der letzten Anmeldung
time_previous_logged		Zeitpunkt der vorletzten Anmeldung
times_logged		Anzahl der Anmeldungen
removed	0 oder 1	0 - existent, 1 - gelöscht

6.2.2 Zugriff

Der Zugriff auf die Datenbank erfolgt, wie beschrieben, über das Package {DBI}[19] und soll hier exemplarisch vorgeführt werden. Die Verbindung der Datenbank und R könnte dabei folgendermaßen aussehen:

```
library(DBI)
library(RSQLite)

# Verbinde die Datenbank mit R
db <- dbConnect(SQLite(), "db/db.sqlite")
```

Tabelle 17: Beispielhafter Inhalt der Tabelle circulation

user_id	subtype_id	quantity	time	op_type
1	2	24	2021-02-18 18:38:00	2
1	3	12	2021-02-18 18:38:04	2
1	4	5	2021-02-18 18:38:07	2
1	5	8	2021-02-18 18:38:12	2
1	6	14	2021-02-18 18:38:16	2
1	11	4	2021-02-18 18:40:14	2
1	12	3	2021-02-18 18:41:09	2
1	17	112	2021-02-18 18:41:40	2
1	1	45	2021-02-18 19:38:35	2
4	7	6	2021-02-18 19:40:53	1

```
# Tabellennamen
dbListTables(db)

## [1] "circulation" "group_type"   "groups"       "subtype"      "type"        "user"

Beispielhaft wird die Tabelle circulation betrachtet. Diese Tabelle speichert alle Transaktionen, die von Nutzern durchgeführt worden sind, also zum Beispiel Ausleihen, Rückgaben und Mengenänderungen.

dbReadTable(db, "circulation")
```

Der Ordner `db/func/` enthält eine Sammlung von über 80 Hilfsfunktionen, die zum Erstellen, Abfragen und Modifizieren der Datenbank verwendet werden. Die Interaktion mit der Datenbank erfolgt ausschließlich über diese Hilfsfunktionen. Dadurch werden Redundanzen ausgeschlossen und es ist darüber hinaus einfacher, die korrekte Funktionalität zu gewährleisten.

Beispielhaft wird die Funktion `db_get_borrowed_quantity(db, subtype_id)` betrachtet. Unter Angabe der Datenbank `db` und einer oder mehrerer Untertypenidentifikationsnummern `subtype_id` wird die ausgeliehene Menge dieser Untertypen zurückgegeben. In `{DBI}` werden Abfragen an die Datenbank mithilfe von `dbGetQuery()` gestellt. Das Resultat der Abfrage wird anschließend weiterverarbeitet. Der Nutzen der Hilfsfunktion wird zusätzlich dadurch verdeutlicht, dass sowohl die abzufragende Tabelle sowie die beteiligten Zeilen und Spalten nicht gesondert angegeben werden müssen.

```
# Funktionsdefinition
db_get_borrowed_quantity <- function(db, subtype_id) {
  borrowed <- DBI::dbGetQuery(
    db,
    "SELECT SUM(quantity) AS borrowed FROM circulation
     WHERE subtype_id = ? AND op_type = 1",
    params = list(subtype_id)
  )$borrowed

  ifelse(is.na(borrowed), 0, borrowed)
}
```

```
# Abfrage der ausgeliehenen Menge für mehrere Untertypen
db_get_borrowed_quantity(db, 1:5)

## [1] 0 13 0 0 2
```

6.3 Cookies

Cookies sind Textdaten, die der Client mit jeder Anfrage an den Server mitsendet [37]. Die Web-Applikation setzt zwei Cookies. Die Cookies sind jeweils für einen Zeitraum von sieben Tagen gültig.

6.3.1 dark-mode

Dieses Cookie kann die Werte "true" oder "false" enthalten. Das Cookie wird gesetzt, wenn der Nutzer den Schalter für den Nachtmodus verwendet. Beim Start der Web-Applikation wird das Cookie verwendet, um den initialen Status für den Nachtmodus zu bestimmen. Falls das Cookie nicht vorhanden ist, wird die Einstellung des Browsers (`window.matchMedia('prefers-color-scheme: dark')`) ausgelesen und im Cookie gesetzt.

6.3.2 user

Dieses Cookie enthält einen verhaschten Nutzernamen und dient zur Identifikation eines angemeldeten Nutzers. Das Cookie wird gesetzt, wenn sich ein Nutzer anmeldet. Beim Start der Web-Applikation wird das Cookie verwendet, um den Nutzer automatisch anzumelden. Falls das Cookie nicht vorhanden ist, muss der Nutzer sich manuell anmelden.

6.3.3 Implementierung

Zur Verwaltung der Cookies wird die Bibliothek js-cookie [38] verwendet. Die Datei `www/js/cookies.js` enthält Hilfsfunktionen zum Lesen und Schreiben von Cookies und verbindet diese mit Inputwerten in Shiny. Die Hilfsfunktionen werden in R mithilfe von `shinyjs::extendShinyjs()` eingebunden.

6.4 Deployment

Der Shiny Server liegt auf einer virtuellen Maschine (VirtualBox). Auf der virtuellen Maschine ist das Betriebssystem *Ubuntu 20.04.2 LTS* ohne grafische Benutzeroberfläche installiert. Änderungen am Shiny Server werden erst wirksam, nachdem der Shiny Server neugestartet wurde. Dazu kann folgender Befehl verwendet werden: `sudo systemctl restart shiny-server`.

Nachfolgend wird auf Ordner verwiesen, in denen für die Web-Applikation relevante Dateien liegen.

- `/etc/shiny-server`

Dieser Ordner enthält die Datei `shiny-server.conf`, mit der der Shiny Server konfiguriert werden kann [39]. Hier kann der Port eingestellt werden, unter dem die Web-Applikation verfügbar gemacht wird. Der Port 3838 ist voreingestellt.

- `/srv/shiny-server/Sensotheke`

Dieser Ordner enthält das Git-Repository, das die tatsächliche Web-Applikation darstellt. Es verweist auf das GitHub-Repository <https://github.com/PFA-WebApp/App.git> als Remote mit Namen `origin`. In der Datei `app.yml` müssen die in Tabelle 18 aufgelisteten Einstellungen vorgenommen werden:

Tabelle 18: Einstellungen in `app.yml`

Einstellung	Beschreibung
<code>showcase</code>	<code>yes</code> für Showcasemodus (Passwörter für Standardnutzer werden eingeblendet, Standardnutzer können nicht gelöscht werden), <code>no</code> (voreingestellt) für Betriebsmodus
<code>url</code>	Basis-URL auf welche QR-Codes verweisen (hier sind Anpassungen zwingend notwendig)

- `/etc/rstudio`

Dieser Ordner enthält die Datei `rserver.conf`, mit der der RStudio Server konfiguriert werden kann [40]. Hier kann der Port eingestellt werden, unter dem der RStudio Server verfügbar gemacht wird. Der Port 8787 ist voreingestellt. Der RStudio Server kann zur nachhaltigen Pflege und Weiterentwicklung der Web-Applikation genutzt werden. Dazu muss der Port aufgerufen werden. In der Login-Maske kann sich mithilfe des Nutzers `pfa` und eines zugehörigen Passworts eingeloggt werden. Danach öffnet sich die Online-Version der Entwicklungsumgebung RStudio, deren Funktionsumfang mit dem der Desktopversion identisch ist.

Für weitere Informationen siehe <https://docs.rstudio.com/ide/server-pro/>. Es ist zu beachten, dass es sich bei der installierten Version um die Open-Source-Version und *nicht* um die Pro-Version handelt.

- `/var/log/shiny-server/Sensotheka`

Dieser Ordner enthält Log-Dateien, die vom Shiny Server erzeugt werden, wenn Komplikationen im Betrieb auftreten. Sollte das der Fall sein, ist es ratsam, diese zunächst zu konsultieren, bevor mit der Fehlerbehandlung begonnen wird.

7 Diskussion und Ausblick

Das Projekt wird mit einer Diskussion der Ergebnisse abgeschlossen. Dabei wird auch thematisiert, wie diese Ergebnisse erzielt wurden. Übergeordnetes Ziel des Projektes war es, eine Web-Applikation zu entwickeln, die die Sensorverwaltung des Fachgebietes Fahrzeugantriebe der Technischen Universität Berlin übernimmt. Dazu musste eine Vielzahl von Konzepten erstellt und umgesetzt werden, unter anderem die Abstraktion der Datenstruktur und deren Abbildung auf eine Datenbank. Die an die Web-Applikation nach Aufgabenstellung gestellten Anforderungen wurden allesamt erfüllt und konnten in Rücksprache mit den Ansprechpartnern am Fachgebiet sogar erweitert werden.

Rückblickend ist festzuhalten, dass der Umfang des Projektes - insbesondere durch zusätzliche Anforderungen - den verfügbaren zeitlichen Rahmen überstiegen hat. Während der Projektphase sind zusätzliche optionale Anforderungen aufgetreten, die mit einem realistischen Qualitätsanspruch nicht umgesetzt werden konnten. Das Hauptaugenmerk bei der Entwicklung der Web-Applikation

lag insbesondere auf der Gewährleistung der Funktionalität über den Rahmen des Projektes hinaus. Diesem Aspekt wurde die Implementierung zusätzlicher Funktionalität geringerer Qualität untergeordnet. Vor allen Dingen wurden das Layout, die Datenbankstruktur und Operationen auf der Datenbank optimiert. Das Layout ist zeitgemäß, konsistent und übersichtlich, sodass der Nutzer sich einfach zurechtfindet. Es überzeugt durch einen hohen Detailgrad. Die Datenbankstruktur ist redundanzfrei und konnte trotz gestiegener Anforderungen über das Projekt zielführend erweitert werden. Eine Vielzahl an Hilfsfunktionen sowie der kohärente Aufbau der Tabellen begünstigt darüber hinaus eine zukünftige Erweiterung auf eine effiziente Weise. Die logische Konsistenz der Daten wird des Weiteren durch den vielfältigen Einsatz von Datenbankbeschränkungen garantiert.

Die Struktur der Web-Applikation begünstigt daher nicht nur ihren zuverlässigen dauerhaften Einsatz sondern auch eine stetige Weiterentwicklung. Darüber hinaus wurde durch die sachgemäße Einrichtung der virtuellen Maschine der Grundstein für Folgeprojekte gelegt. Der Einstieg in das Projekt für mit R und der Entwicklung von Web-Applikationen unerfahrenen Personen ist sicherlich mit Hürden verbunden. Durch den vorliegenden Bericht und die durchdachte Struktur der Web-Applikation soll er jedoch so einfach wie möglich gemacht werden. Folgeprojekte könnten sich zum Beispiel mit den nachfolgenden Inhalten beschäftigen:

- Erweiterung der Datenstruktur um Stücklisten, um komplexere Beziehungen zwischen Elementen abbilden zu können
- E-Mail-Integration für Registrierung, Vergessen des Passworts und Benachrichtigungen
- Hinterlegen zusätzlicher Informationen für die Ausleihe (vorraussichtliches Rückgabedatum, Verwendungszweck etc.)
- Export von Daten
- Einstellen kritischer Bestände auf Nutzerebene
- Integration zusätzlicher fachgebetspezifischer Anwendungen (Kalender, Dateiverwaltung etc.)

Abschließend freuen wir uns, die fertige Web-Applikation dem Fachgebiet zu übergeben und sind gespannt, in welchem Umfang sie in Zukunft eingesetzt und sogar weiterentwickelt wird.

Literaturverzeichnis

- [1] Karl Hermann Fuchs Emmerich Fuchs und Christian H. Hauri. *Requirements-Engineering in IT effizient und verständlich*. 1. vieweg, 2002.
- [2] Christopher Pitt. „Deployment“. In: *The Definitive Guide to AdonisJs: Building Node.js Applications with JavaScript*. Berkeley, CA: Apress, 2018, S. 207–222. ISBN: 978-1-4842-3390-0. DOI: 10.1007/978-1-4842-3390-0_15. URL: https://doi.org/10.1007/978-1-4842-3390-0_15.
- [3] Bosse Küllenberg. *Wert und Zielbild eines transformativen Tech Stack im Rahmen einer digitalen Transformation*. 1. Springer Gabler, Wiesbaden, 2020.
- [4] Frank Thiesing und Sebastian Kortenmeyer. *Entwicklung moderner Web-Anwendungen mit Open-Source-Bausteinen*. 1. Springer-Verlag, Osnabrück, 2008.
- [5] Dietrich Baumgarten. *Kompakt im Doppelpack: HTML und JavaScript - Webdesign für Einsteiger*. 1. B. G. Teubner GmbH, Stuttgart/Leipzig/Wiesbaden, 2002.
- [6] Joachim Wietzke und Manh Tien Tran. *Automotive Embedded Systeme*. 1. Springer-Verlag Berlin/Heidelberg, 2005.
- [7] Jeff Allen. *SQLite Consortium: Appropriate Uses for SQLite*. Letzter Aufruf: 29.03.2021. URL: <https://shiny.rstudio.com/articles/shiny-server.html>.
- [8] John M. Chambers. *Object-Oriented Programming, FunctionalProgramming and R*. Letzter Aufruf: 29.03.2021. URL: <https://arxiv.org/pdf/1409.3531.pdf>.
- [9] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: <https://www.R-project.org>.
- [10] Kurt Hornik. *R FAQ*. 2020. URL: <https://CRAN.R-project.org/doc/FAQ/R-FAQ.html>.
- [11] David Granjon. *bs4Dash: A Bootstrap 4 Version of shinydashboard*. R package version 0.5.0. 2019. URL: <https://CRAN.R-project.org/package=bs4Dash>.
- [12] Yihui Xie, Joe Cheng und Xianying Tan. *DT: A Wrapper of the JavaScript Library DataTables*. R package version 0.17. 2021. URL: <https://github.com/rstudio/DT>.
- [13] Joe Cheng u. a. *htmltools: Tools for HTML*. R package version 0.5.1.1. 2021. URL: <https://github.com/rstudio/htmltools>.
- [14] Sébastien Bihorel. *rclipboard: Shiny/R Wrapper for clipboard.js*. R package version 0.1.3. 2021. URL: <https://github.com/sbihorel/rclipboard/>.
- [15] Dean Attali. *shinydisconnect: Show a Nice Message When a Shiny App Disconnects or Errors*. R package version 0.1.0. 2020. URL: <https://github.com/daattali/shinydisconnect>.
- [16] Dean Attali. *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*. R package version 2.0.0. 2020. URL: <https://deanattali.com/shinyjs/>.
- [17] John Coene. *waiter: Loading Screen for Shiny*. R package version 0.2.0. 2021. URL: <https://CRAN.R-project.org/package=waiter>.
- [18] Simon Urbanek und Jeffrey Horner. *Cairo: R Graphics Device using Cairo Graphics Library for Creating High-Quality Bitmap (PNG, JPEG, TIFF), Vector (PDF, SVG, PostScript) and Display (X11 and Win32) Output*. R package version 1.5-12.2. 2020. URL: <http://www.rforge.net/Cairo/>.
- [19] R Special Interest Group on Databases (R-SIG-DB), Hadley Wickham und Kirill Müller. *DBI: R Database Interface*. R package version 1.1.1. 2021. URL: <https://CRAN.R-project.org/package=DBI>.

- [20] Hadley Wickham u. a. *dplyr: A Grammar of Data Manipulation*. R package version 1.0.5. 2021. URL: <https://CRAN.R-project.org/package=dplyr>.
- [21] Jim Hester. *glue: Interpreted String Literals*. R package version 1.4.2. 2020. URL: <https://CRAN.R-project.org/package=glue>.
- [22] Vitalie Spinu, Garrett Grolemund und Hadley Wickham. *lubridate: Make Dealing with Dates a Little Easier*. R package version 1.7.10. 2021. URL: <https://CRAN.R-project.org/package=lubridate>.
- [23] Kirill Müller u. a. *RSQLite: SQLite Interface for R*. R package version 2.2.5. 2021. URL: <https://CRAN.R-project.org/package=RSQLite>.
- [24] Hadley Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*. R package version 1.4.0. 2019. URL: <https://CRAN.R-project.org/package=stringr>.
- [25] Kirill Müller und Hadley Wickham. *tibble: Simple Data Frames*. R package version 3.1.0. 2021. URL: <https://CRAN.R-project.org/package=tibble>.
- [26] Lionel Henry und Hadley Wickham. *purrr: Functional Programming Tools*. R package version 0.3.4. 2020. URL: <https://CRAN.R-project.org/package=purrr>.
- [27] Victor Teh. *qrcode: QRcode Generator for R*. R package version 0.1.1. 2015. URL: <https://CRAN.R-project.org/package=qrcode>.
- [28] Kevin Ushey. *renv: Project Environments*. R package version 0.13.1. 2021. URL: <https://rstudio.github.io/renv/>.
- [29] Jeremy Stephens u. a. *yaml: Methods to Convert R Data to YAML and Back*. R package version 2.2.1. 2020. URL: <https://github.com/viking/r-yaml/>.
- [30] Peter Bühler, Patrick Schlaich und Dominik Sinner. „Datenbanken“. In: *Webtechnologien: JavaScript – PHP – Datenbank*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, S. 68–91. ISBN: 978-3-662-54730-4. DOI: 10.1007/978-3-662-54730-4_4. URL: https://doi.org/10.1007/978-3-662-54730-4_4.
- [31] Kenneth C. Laudon, Jane Price Laudon und Detlef Schroeder. *Wirtschaftsinformatik - Eine Einführung*. 3. Pearson, 2015.
- [32] *SQLite Consortium: Appropriate Uses for SQLite*. Letzter Aufruf: 14.03.2021. URL: <https://www.sqlite.org/whentouse.html>.
- [33] Stefan Berndes, Klaus Kornwachs und Uwe Lünstroth. „Einführung“. In: *Softwareentwicklung: Erfahrung und Innovation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 1–10. ISBN: 978-3-642-56365-2. DOI: 10.1007/978-3-642-56365-2_1. URL: https://doi.org/10.1007/978-3-642-56365-2_1.
- [34] Frank Padberg und Walter Tichy. „Schlanke Produktionsweisen in der modernen Softwareentwicklung“. In: *WIRTSCHAFTSINFORMATIK* 49.3 (Juni 2007), S. 162–170. ISSN: 1861-8936. DOI: 10.1007/s11576-007-0046-1. URL: <https://doi.org/10.1007/s11576-007-0046-1>.
- [35] Mariot Tsitoara. *Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer*. Berkeley, CA: Apress, 2020. ISBN: 978-1-4842-5313-7. DOI: 10.1007/978-1-4842-5313-7_1. URL: https://doi.org/10.1007/978-1-4842-5313-7_1.
- [36] Winston Chang. *Modularizing Shiny app code*. 2020. URL: <https://shiny.rstudio.com/articles/modules.html>.
- [37] *Using HTTP cookies*. Mozilla Foundation. 2021. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>.

- [38] Klaus Hartl und Fagner Brack. *A simple, lightweight JavaScript API for handling browser cookies*. URL: <https://github.com/js-cookie/js-cookie>.
- [39] *Shiny Server Professional v1.5.16 Administrator's Guide*. RStudio. URL: <https://docs.rstudio.com/shiny-server/>.
- [40] *RStudio Server Professional Edition 1.4.1106-5 - Administration Guide*. RStudio. URL: <https://docs.rstudio.com/ide/server-pro/>.