Contents

1 Übersicht

2 Einführung

Die Verwaltung von Messmitteln 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 zu erstellen. 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 für Messmittel, bereitgestellt und der Bestand von Messmitteln 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, die Beschreibung des finalen Produktes hinsichtlich technischer Umsetzung und Funktionalität und zeigt zusätzlich Anknüpfungspunkte für Folgeprojekte auf.

Messmittel = Sensor

3 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 sind dabei aus dem in der Aufgabenstellung beschriebenen Funktionstest abgeleitet worden:

- Erstellen von Sensoren
- Erstellen von Nutzern
- Matching unterschiedlicher Sensortypen 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.

3.1 Funktionale Anforderungen an die Web-Applikation

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

3.1.1 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*

3.1.2 Dateiverwaltung

- 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

3.1.3 Nutzerverwaltung

Wie eingangs beschrieben, ist eine rechtebasierte Verwaltung 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. Eine genaue Aufschlüsselung der Rechte der drei Rollen - und somit der Anforderungen an die Nutzerverwaltung - kann Abbildung ... entnommen werden.

TODO: Rechte-Matrix

3.1.4 Ausleihverwaltung

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

3.1.5 Bestandsinformation

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

3.2 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.

3.3 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.

3.3.1 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.

3.3.2 Deployment

Als Deployment wird die Integration der Web-Applikation und der Datenbank in die bestehende Infrastruktur bezeichnet. 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*

4 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.

4.1 Framework

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

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 Browser 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 dem visuellen Part legen. Multipage-Frameworks enthalten mehrere Seiten, wohingegen Singlepage-Frameworks nur eine einzige Seite darstellen.

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. 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:

- Moderne Templates
- Fokus auf konkrete Funktionalität
- Reaktives Zustandsmodell
- Modularisierbarkeit
- Für Anwendungsfall ausreichende Performance
- Einfache Integration von Datenbanken

4.2 Programmiersprache

R ist eine Multiparadigmen-Programmiersprache. Je nach Anwendungsfall kann somit zum Beispiel objektorientiert oder funktional programmiert werden. R verfügt einen Pool an Standardbibliotheken und kann einfach durch selbstgeschriebene und frei verfügbare Packages erweitert werden. Das Comprehensive R Archive Network (CRAN) stellt eine Vielzahl von quelloffenen Bibliotheken zur Verfügung beispielsweise {shiny}, das die Funktionalitäten des Frameworks beinhaltet. Die folgenden Tabellen 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.

4.2.1 Packages für {shiny}

Package	Beschreibung
{bs4Dash}	AdminLTE-Template
{DT}	DataTables für {shiny}
{htmltools}	HTML-Repräsentation in R
{rclipboard}	Zwischenablage
{shinydisconnect}	Verbindungsverlustbildschirm
{shinyjs}	Integration von Custom-JavaScript

Package	Beschreibung
{waiter}	Ladebildschirm

4.2.2 Packages für die Programmierung

Package	Beschreibung
{Cairo}	PDF-/PNG-/SVG-Erstellung
{DBI}	Datenbankinterface
{dplyr}	Datentransformationen
{glue}	String-Erzeugung
{lubridate}	Datumsformat
{RSQLite}	SQLite-Datenbank
{stringr}	String-Manipulation
{tibble}	Tabellenformat
{purrr}	Funktionale Programmierung
{qrcode}	Erstellung von QR-Codes
{renv}	Packagemanagement
{yaml}	YAML-Dateiformat

4.3 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. Es existieren verschiedene Datenbankmodelle, unter anderem das Netzwerk-, das objektorientierte, das hierachische oder das relationale Datenbankmodell.[1] 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. 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. 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.

Es gibt eine Vielzahl verschiedener relationaler Datenbankmanagementsysteme, die sich hinsichtlich ihrer Anwendungsbereiche und Skalierbarkeit unterscheiden. Für Projekte kleinen und mittleren Umfanges (unter 100.000 Aufrufe / Tag[2]) 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.

5 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.

[1] Laudon, K.C.; Laudon, J.P.; Schroeder, D.: *Wirtschaftsinformatik - Eine Einführung*, Pearson, 2015, S. 295-300
[2] SQLite Consortium: *Appropriate Uses for SQLite*, URL: https://www.sqlite.org/whentouse.html, Letzter Aufruf: 14.03.2021

5.1 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. Im Kontrast dazu stehen die agilen Modelle, in denen alle Phasen wiederholt durchlaufen werden. Agile Modelle sind in der Regel deutlich flexibler, da zum Beispiel 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.

Wir haben uns 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.

5.2 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.

5.3 Versionskontrolle

Versionskontrolle ist notwendig,

6 Umsetzung

Neben der technischen Umsetzung ist insbesondere der Arbeitsprozess von herausragender Bedeutung.

7 Diskussion

8 Ausblick