FH MÜNSTER
University of Applied Sciences

Fachbereich Elektrotechnik und Informatik

ANGULAR WEB-APP ZUR MESSDATENVISUALISIERUNG

MASTERPROJEKT

Autoren:
Mark Kleine Vorholt
Robin Weiß

Betreuer:
Prof. Dr.-Ing. Peter Glösekötter
André Loechte, M. Sc.

Februar 2018

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

LISTINGS

1 EINLEITUNG

Der steigenden Speicherbedarf für elektrische Energie erfordert die stetige Weiterentwicklung von Batteriespeichersystemen. Die elektrochemische Reaktion von Zink mit Luftsauerstoff bietet ein beträchtliches Potenzial für wiederaufladbare Speicher für große Energiemengen, bei gleichzeitig hohen Energiedichten. Der besondere Vorteil von Zink-Luft-Akkumulatoren liegt neben der theoretisch dreifach größeren Energiedichte als der von Lithium-Ionen-Akkumulatoren in den niedrigen Kosten und der hohen Sicherheit.

Im Rahmen eines Verbundforschungsprojektes der Technischen Universität Dortmund und der Fachhochschule Münster wurde ein neuartiger Zink-Luft-Akku mit dem dazugehörigen Batterie-Management-System entwickelt. Um die bei der Entwicklung anfallenden Messdaten für zukünftige Auswertungen zentral zu speichern und sie anderen Projektpartnern zur Verfügung zu stellen, wurde im Rahmen eines Masterprojektes eine Web-App entwickelt, die die aus den unterschiedlichen Versuchsreihen anfallenden Messdaten visualisiert und zur weiteren Verarbeitung zum Download zur Verfügung stellt.

1.1 ANFORDERUNGEN

Nachfolgend werden die Anforderungen an die Web-App im Detail beschrieben:

- Messdaten sollten in einem einheitlichen Format hochladbar sein und automatisch in eine Datenbank übernommen werden. Metadaten und Zusatzinformationen zu einzelnen Messaufbauten und -abläufen sollten in den jeweiligen Files gespeichert und automatisch ausgelesen werden können.
- Die als Zeitreihen und Ortskurven vorliegenden Messdaten sollten visualisiert werden. Dabei sollten unterschiedliche Ortskurven und Zeitreihen eines Lade- bzw. Entladevorgangs gemeinsam dargestellt werden können. Das zeitliche Intervall der Darstellung einer Messreihe sollte veränderbar sein. Um eine ausreichend hohe Auflösung bei gleichzeitig optimiertem HTTP-Traffic zu erreichen, sollten Daten beim Zoomen “lazy” nachgeladen werden.
- Um Zukunftssicherheit und Skalierbarkeit zu gewährleisten, sollten zusätzlich zu einzelnen Zellen auch Stacks, das sind Verschaltungen von einzelnen Zellen, und Systeme, das sind Verschaltungen von einzelnen Stacks, angelegt werden können. Die

App sollte eine Filterfunktion nach unterschiedlichen Kategorien sowie eine Suchfunktion in den jeweiligen Kategorien bereitstellen.

- Die App sollte über einen Webserver im Labor ausgeliefert werden und im Webbrowser laufen. Um zu verhindern, dass Dritte die gespeicherten Daten einsehen und manipulieren können, sollte der Zugriff über einen Login geschützt werden.
- Das User Interface sollte responsive sein, damit die App auch auf mobilen Geräten nutzbar ist.

2 IMPLEMENTIERUNG

Um die entwickelte Anwendung plattformübergreifend einsetzen zu können, wurde eine Angular-Single-Page-App mit ASP.NET-Back-End entwickelt. Dieses Kapitel beschreibt die Implementierung und die dabei genutzten Komponenten im Detail. Der Source-Code [9] ist auf GitHub verfügbar

2.1 DOMAIN MODEL

Das Domain Model besteht aus den in Abbildung 1 dargestellten Klassen. Dabei wird eine Messung durch Messdaten sowie zugehörige Metadaten, wie Name und Beschreibung, repräsentiert. Eine Messung kann entweder zu einer Zelle, einem Stack oder einem System gehören. Ein Stack ist eine Verschaltung von mehreren Zellen, ein System eine Verschaltung von mehreren Stacks. Systeme werden intern als Batteries bezeichnet, um Konflikte mit C# Bibliotheken zu vermeiden. Die Enumeration `MeasurementType` beschreibt den Typ der Messung: Undefiniert, Zeitreihe, Ortskurve, oder Sonstige. `CircuitType` gibt die Schaltungsart an: Reihen- bzw. Parallelschaltung oder Sonstige. Neue Instanzen des jeweiligen Typs können zur Datenbank hinzugefügt, gelöscht, und bearbeitet werden. Es stehen jeweils List- und Detail-Views zur Verfügung. Die Messung hat eine Upload- und eine Download-Methode.

Abbildung 2 zeigt die entwickelten Angular-Components und das Routing. Die Components werden weiter unten näher beschrieben.

2.2 FRAMEWORKS UND TECHNOLOGIEN

Im Folgenden werden die verwendeten Frameworks und Technologien näher beschrieben.

2.2.1 *Angular*

Angular 4.x [3] ist ein TypeScript-basiertes Frontend-Webframework. Angeführt durch Google, wird es von einer Community aus Einzelpersonen und Unternehmen entwickelt und als Open-Source-Software über GitHub [4] publiziert.

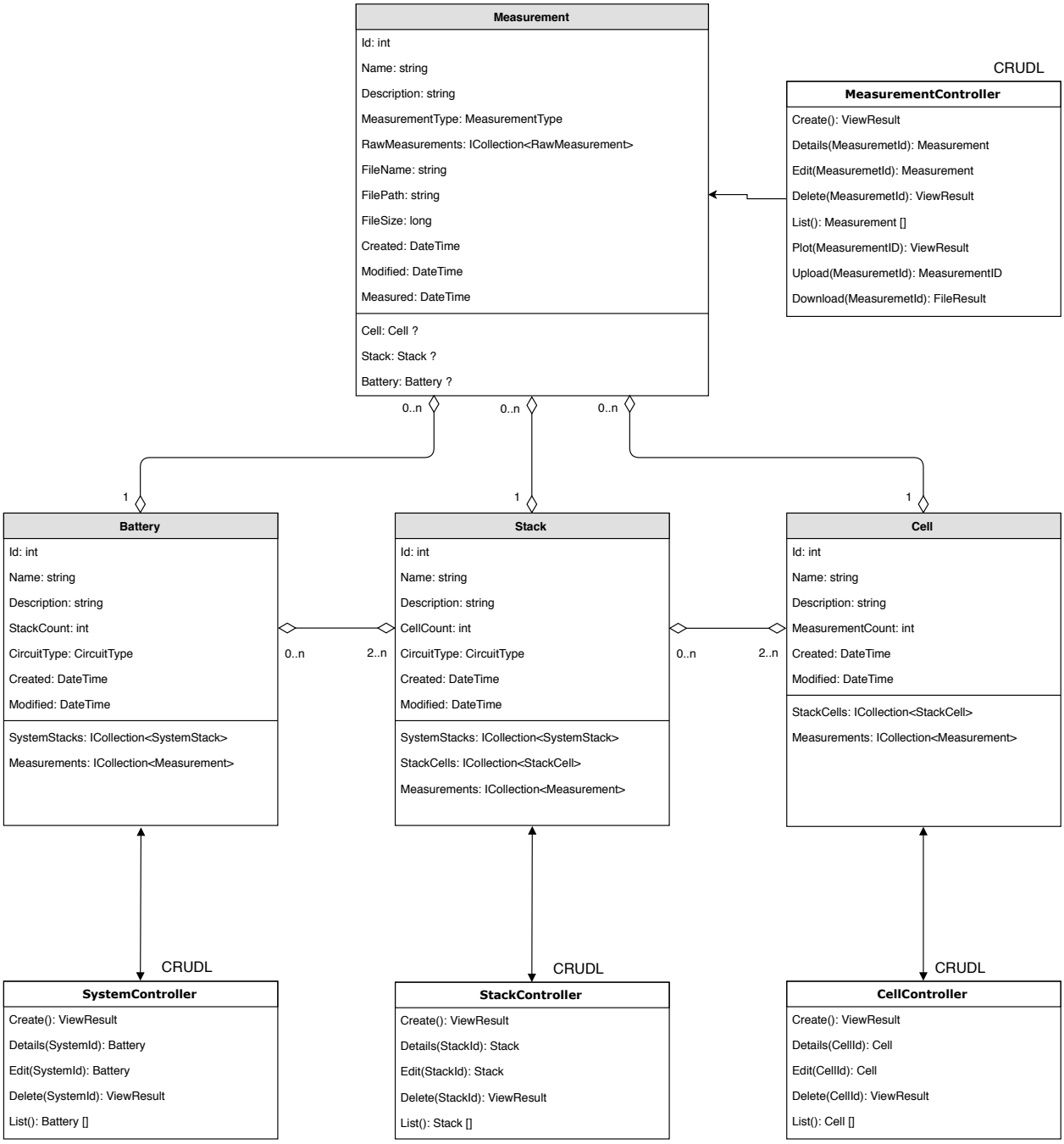

Abbildung 1: Domain Model als UML Klassenhierarchie

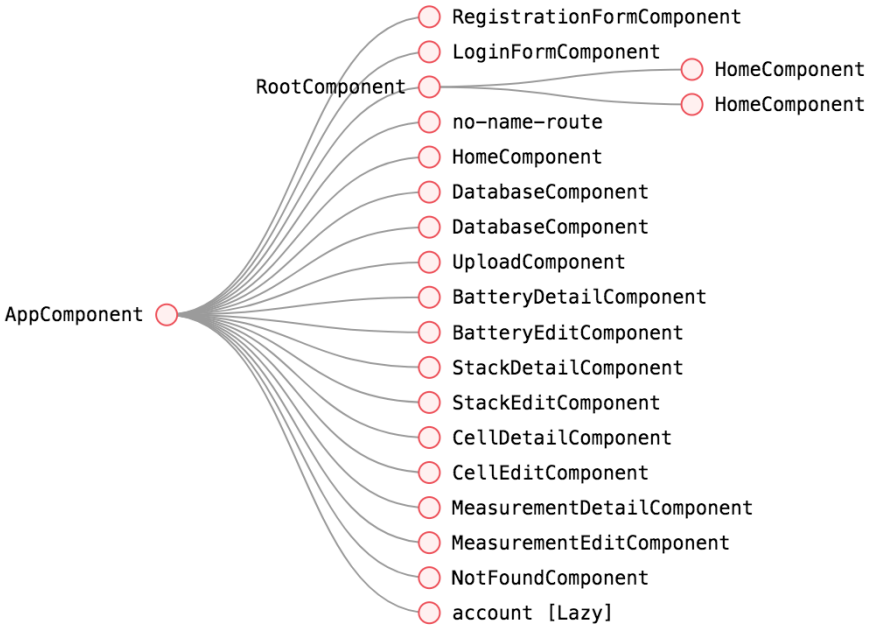

Abbildung 2: Angular-Components und Routing

2.2.2 *ASP.NET Core*

Die General-Purpose-Development-Platform .NET Core wird unter der Koordination von Microsoft zusammen mit der .NET Community entwickelt und ist als Open Source-Projekt über GitHub [7] verfügbar. Sie bietet eine platformübergreifende Lösung zur Entwicklung und Ausführung von Anwendungsprogrammen und läuft unter Windows, macOS and Linux. ASP.NET steht für Active Server Pages .NET und ist ein Web Application Framework von Microsoft, mit dem sich dynamische Webseiten, Webanwendungen und Webservices in C# entwickeln lassen. Im Hintergrund läuft der Kestrel Web Server für ASP.NET [1]. Als Entwicklungsumgebung wurde Visual Studio Code mit C#-Extension genutzt, ein kostenloser und quelloffener Texteditor.

2.2.3 *Entity Framework Core*

Entity Framework Core ist eine abgespeckte, erweiterbare, platformübergreifende Version des bekannten Entity Frameworks, ein Framework für objektrelationale Abbildung von Microsoft. Entity Framework Core ermöglicht es ausschließlich unter Benutzung von .NET-Objekten mit Datenbanken zu arbeiten und beseitigt so die Notwendigkeit eigenen Code zum Zugriff auf die Datenbank zu schreiben [8]. EF Core verwendet das aus Entity-Klassen bestehende C#-Domain-Model um daraus mit Hilfe von Migrations eine Datenbank zu erstellen. So kann das Domain Model während der Entwicklung einfach er-

weitert oder geändert werden, ohne händisch SQL-Befehle schreiben und testen zu müssen. Ein Datenzugriff auf eine Instanz einer Entity-Klasse wird durch Language Integrated Queries (LINQ) beschrieben [7].

2.2.4 *SQLite*

SQLite implementiert eine eigenständige, serverlose SQL-Database-Engine und ist dabei die weltweit meistverwendete ihrer Art [6]. Anders als die meisten anderen SQL Datenbanken schreibt und liest SQLite direkt in und aus herkömmlichen Files auf einem Datenträger. Dabei ist das Fileformat platformübergreifend und frei kopierbar zwischen 32-bit und 64-bit Systemen sowie zwischen Big-Endian and Little-Endian Architekturen.

2.2.5 *Bootstrap und Now UI Kit*

Bootstrap [10] ist ein freies CSS-Framework das responsive Webdesign vereinfacht. Es enthält auf HTML und CSS basierende Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen, Grid-Systeme, Navigations- und andere Oberflächengestaltungselemente. Bootstrap ist modular aufgebaut und besteht im Kern aus Less-Stylesheets. Als Erweiterung wurde das Now UI Kit [2] verwendet, was ein modernes, flaches und intuitives User Interface realisiert.

2.2.6 *Highcharts*

Highcharts [5] ist eine in pure JavaScript geschriebene Visualisierungslibrary der Firma Highsoft. Mit ihr lassen sich unter anderem interaktive Lineplots und Ortskurven erstellen. Auch lazy loading von Daten wird mit ihr vereinfacht.

2.3 JSON-FILEFORMAT

Das JSON-Format wurde gewählt, weil es in C# vorgefertigte JSON-to-Object Parser gibt und es flexibel und beliebig erweiterbar ist. Anders als beim ursprünglichen zum Logging genutzten CSV-Format, ist es einfach Metadaten und Zusatzinformationen zu speichern, ohne diese fest in gewissen Zeilen und Spalten zu kodieren. Weiterhin ist es menschenlesbar und portabel, anders als .mat-Files, die ursprünglich zur Speicherung der Ortskurven dienten. Listing 1 und 2 zeigen jeweils die typische Struktur eines JSON-Files für eine Ortskurve und eine Zeitreihe mit beschreibenden Kommentaren.

Listing 1: JSON Fileformat Ortskurven

```
{
        // 0 = Undefined, 1 = Zeitreihe,
        // 2 = Ortskurve, 3 = Sonstige
        "type": 2,
        "battery": null, // System ID
        "stack": 3, // Stack ID
        "cell": null, // Zellen ID
        "time": 1502702593000, // UNIX Timestamp
        "mode": "Discharge", // State of Charge: Laden oder Entladen
        "current": -1000, // Ladestrom
        "charge": -308843203.7, // Ladung
        "spectrum": {
                "frequency": [ // Frequenzen
                        10,
                        20,
                        30,
                        40,
                        50
                ],
                "impedance": {
                        "_ArrayType_": "double",
                        "_ArraySize_": [37, 1],
                        "_ArrayIsComplex_": 1,
                        "_ArrayData_": [        // Koplexe Impedanzen
                                [0.09263688244, 0.01274971625],
                                [0.09890474907, -0.00793708269],
                                [0.08428400943, -0.009960039118],
                                [0.08048473845, -0.01711342423],
                                [0.08200628526, -0.0163586119]
                        ]
                }
        },
        "rawdata": [{
                "frequency": 0.1,
                "timepoints": [  // Zeitstempel
                        0,
                        5.243e-06,
                        1.0486e-05,
                        1.5729e-05,
                        2.0972e-05,
                        2.6214e-05,
                        3.1457e-05
                ]
        }]
}
```

Listing 2: JSON Fileformat Zeitreihen

```
{
    "measurementType": 1, // 0 = Undefined, 1 = Zeitreihe, 2 = Ortskurve, 3 = Sonstige
    "stackId": 121,
    "data": [
        {
```

```
            "capacity": -13970,
            "current": 0,
            "mode": "discharging",
            "time": 1505127209192,
            "voltage": 1378
        },
        {
            "capacity": -25217,
            "current": -1419,
            "mode": "discharging",
            "time": 1505127219195,
            "voltage": 1106
        },
        {
            "capacity": -43090,
            "current": -1861,
            "mode": "charging",
            "time": 1505127229189,
            "voltage": 1100
        }
    ]
}
```


2.4 REST API

Representational State Transfer (REST) bezeichnet ein Programmierparadigma für verteilte Systeme, insbesondere Webservices. In einer Client-Server-Architektur stellt der Server einen Dienst bereit, der bei Bedarf vom Client angefragt werden kann. So stellt die REST API in unserer Single-Page-App Mess- und Metadaten aus der Datenbank bereit, die bei Bedarf dynamisch geladen bzw. nachgeladen werden.

2.5 ZUGRIFFSRECHTE

Der Authentifizierung von Usern wurde mit Hilfe von JSON Web Tokes implementiert. Typisches Einsatzgebiet für JWT ist die fortlaufende Authentifizierung bei Single-sign-on. Da REST API's zustandslos sind muss bei jeder Anfrage vom Client an den Server diese mit allen Informationen bestückt sein, die für die Verarbeitung der Anfrage benötigt wird. Damit man über die API Daten einsehen, hochladen, herunterladen und verändern kann, muss also immer ein vom Server zuvor generiertes JWT mitgesendet werden.

3 FUNKTIONALITÄTEN

Nachfolgend ist ein möglicher Workflow beschrieben um die Funktionalitäten der App zu erläutern. Abbildung 3 zeigt den Home-View, der den Ausgangspunkt dafür bildet.

3.1 LOGIN

Klickt man auf den Login-Button, oder auf einen anderen Navigationsbutton während kein User eingeloggt ist, wird man zum Login-View weitergeleitet, der in Abbildung 4 dargestellt ist.

3.2 DASHBOARD

Nach dem Login wird man zum Dashboard-View weitergeleitet. Dort wird der Name des Users sowie seine Rolle mit der er eingeloggt ist angezeigt, wie in Abbildung 5. Gleichzeitig ändern sich die Bedienelemente in der Navigationsleiste, über die man als Admin neue User hinzufügen, sich selbst ausloggen, oder die eigenen Nutzerdetails einsehen kann.

3.3 UPLOAD

Klickt man nun über die Navigationsleiste am obigen Bildschirmrand auf Upload, gelangt man zum Upload-View, wie in Abbildung 6 dargestellt, mit Hilfe dessen man per Drag-and-drop oder per Filebrowser Messdatenfiles zum Hochladen auswählen kann. Die Files werden in einer Warteschlange mit der jeweiligen Filegröße angezeigt und können einzeln oder alle nacheinander hochgeladen werden. Eine Progress-Bar zeigt jeweils Einzel- und Gesamtfortschritt des Uploads an.

3.4 DATENBANK

War der Upload erfolgreich wird man zur Datenbank weitergeleitet, in der, wie in Abbildung 7 zu sehen ist, durch Klicken auf die Seitenleiste eine Ansicht nach Systemen, Stacks, Zellen und Messungen ausgewählt werden kann. Die Ansicht mit den Messungen wird nach dem Upload automatisch ausgewählt. Hier finden sich alle Messungen in der Datenbank, sortiert in absteigender Reihenfolge, also mit den letzten Uploads ganz oben. Zusätzlich zum Namen der Messung,

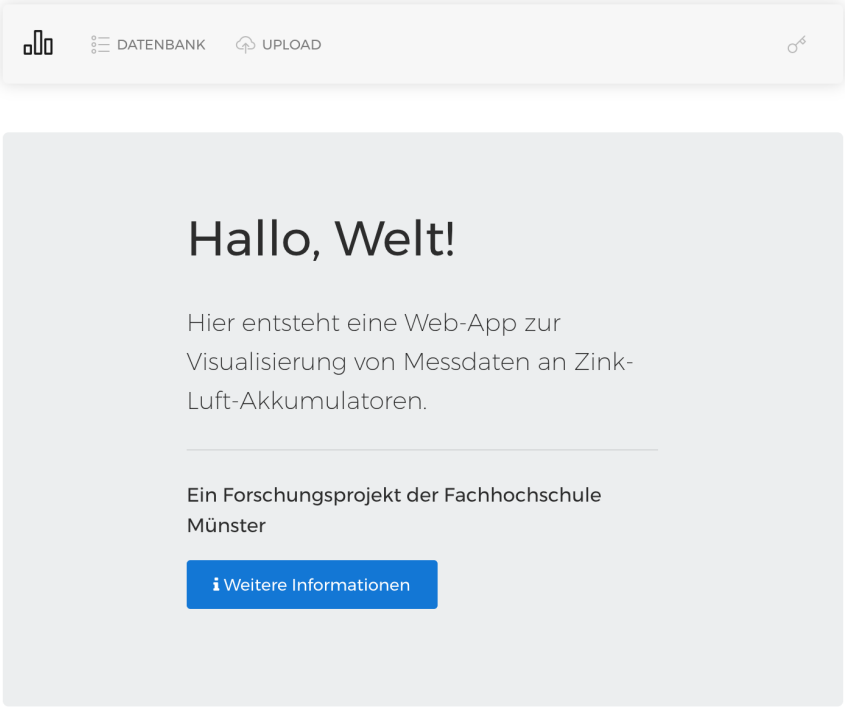

Abbildung 3: Home-View

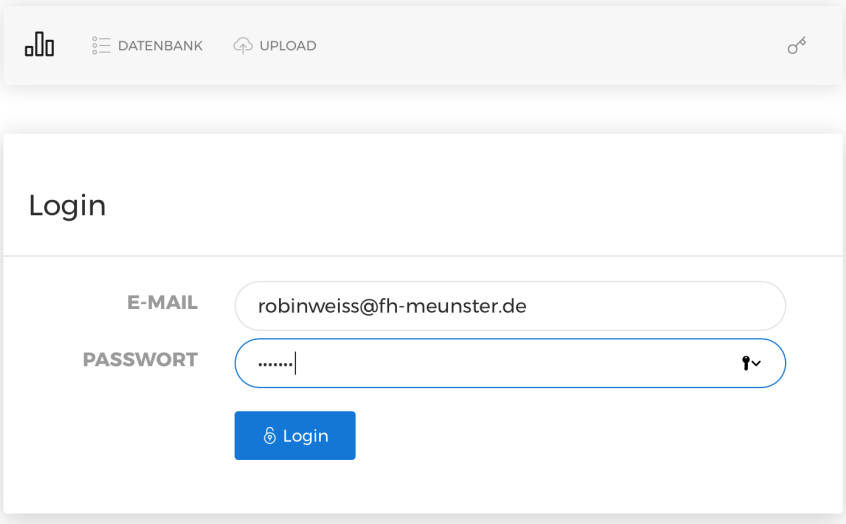

Abbildung 4: Login-View

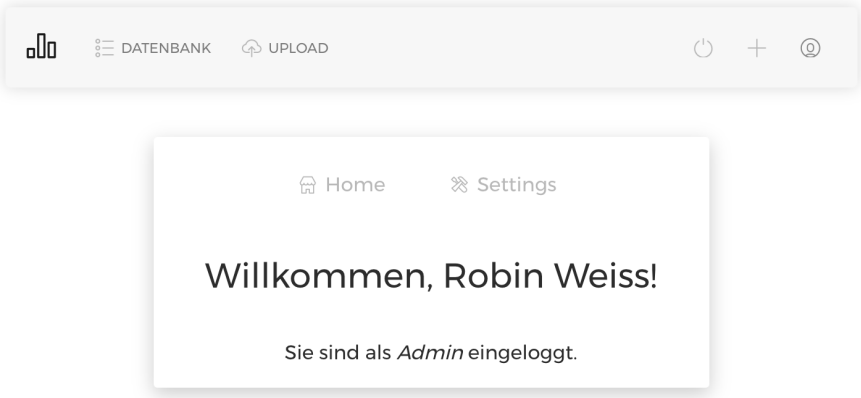

Abbildung 5: Dashboard-View

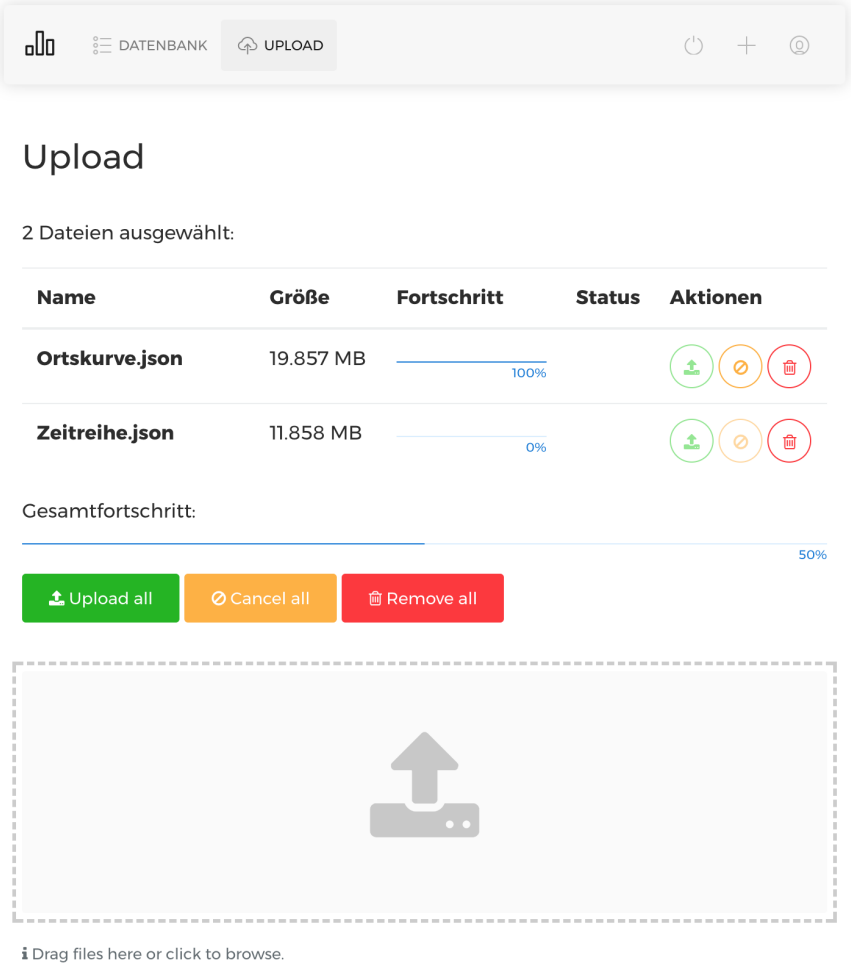

Abbildung 6: Upload-View

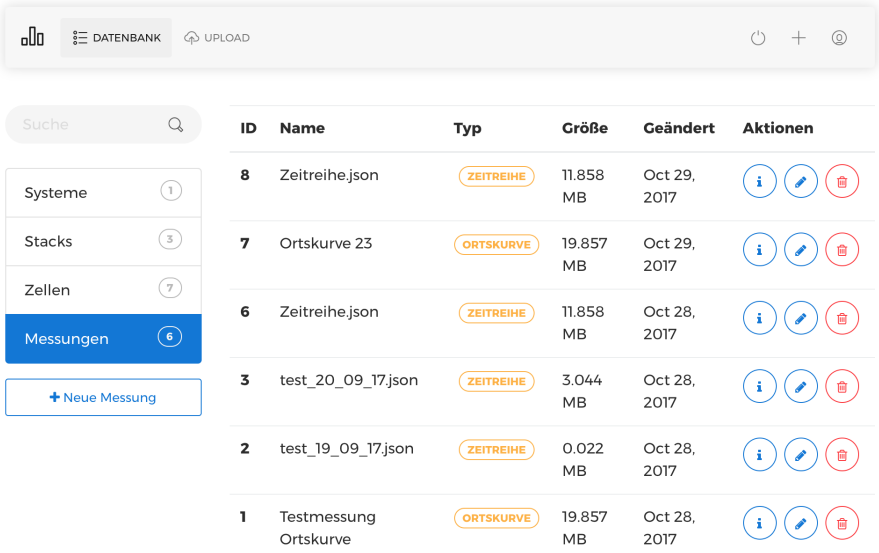

Abbildung 7: Datenbank View mit Messungs-Ansicht

der mit dem Dateinamen initialisiert wird wenn es nicht anders im JSON-File spezifiziert ist, sind der Typ der Messung, also Zeitreihe oder Ortskurve, die Größe und das Datum der letzten Änderung in einer Tabelle dargestellt. Über drei Aktionsbuttons kann man die Details der Messung einsehen, sie bearbeiten oder löschen. Es werden jeweils 10 Messungen pro Seite dargestellt. Sind mehr als 10 Messungen in der Datenbank vorhanden, wird unter den Messungen eine Navigation mit Seitenzahlen eingeblendet.

Wählt man beispielsweise die Ansicht Stacks, wie in Abbildung 8 dargestellt, so werden die Stacks kachelweise mit Zusatzinformationen wie der Art der Verschaltung – zum Beispiel Reihenschaltung – und der Anzahl der zugehörigen Zellen angezeigt. Außerdem wird der Beschreibungstext und das Erstellungs- sowie letzte Änderungsdatum angezeigt. Über die oben beschriebenen Aktionsbuttons kann der Stack geändert oder gelöscht werden. Klickt man auf Details, gelangt man in die Detailansicht.

3.5 DETIALANSICHT

Die Detailansicht in Abbildung 9 zeigt wiederum die Beschreibung, die Markdownbefehle unterstützt, und listet die zu dem jeweiligen Stack gehörenden Messungen, unterteilt in Zeitreihen und Ortskurven, in einer Tabellenansicht ähnlich der Ansicht der Messungen in der Datenbank, auf.

Klickt man nun auf eine Messung wird sie unter der entsprechenden Tabelle geplottet, wie in Abbildung 10 dargestellt. Der Klick auf eine der Messungen führt in die Detailansicht der jeweiligen Mes-

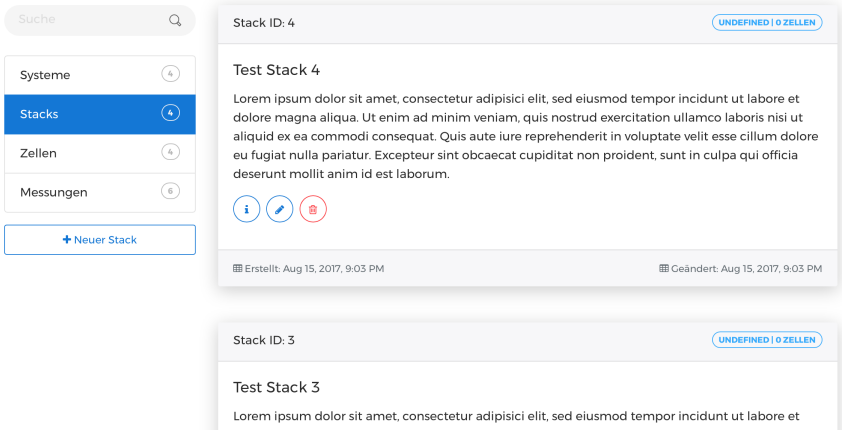

Abbildung 8: Datenbank View mit Stack-Ansicht

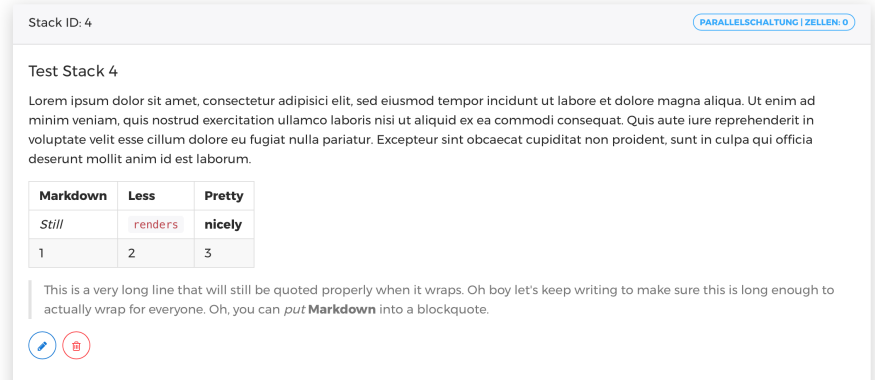

Abbildung 9: Stack-Detail-Ansicht

Abbildung 10: Stackmessung

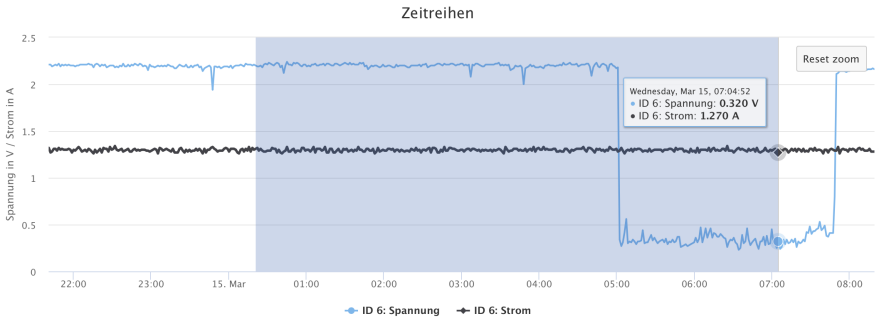

Abbildung 11: Zoom-Funktion vorher

sung. Dort werden neben Details zur Messung auch jedes Mal die Messung als Plot dargestellt.

3.6 PLOTS

Zeitreihen sind wie bei allen anderen Plots zoombar, wie in Abbildung 11 (vorher) und Abbildung 12 (nachher) dargestellt. Die Messdaten werden automatisch vom Server nachgeladen um den Plot mit einer höheren Auflösung darzustellen.

Bei Zeitreihen mit unterschiedlichen Größen können einzelne Zeitreihen durch Klick auf die Legende ein- und ausgeblendet werden um sie so vergrößert darzustellen, wie in Abbildung 13 (Ladung aktiviert) und 14 (Ladung deaktiviert) dargestellt. Die Skalierung erfolgt automatisch.

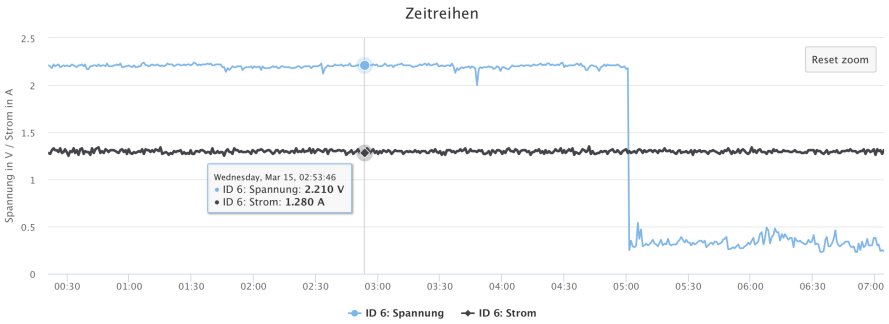

Abbildung 12: Zoom-Funktion nachher

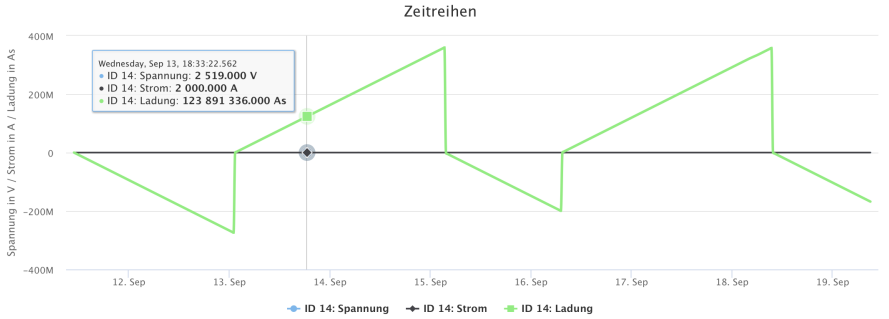

Abbildung 13: Ladungsplot aktiviert

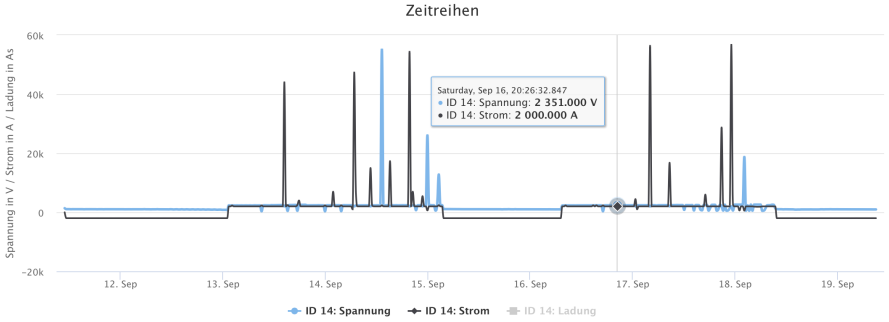

Abbildung 14: Ladungsplot deaktiviert

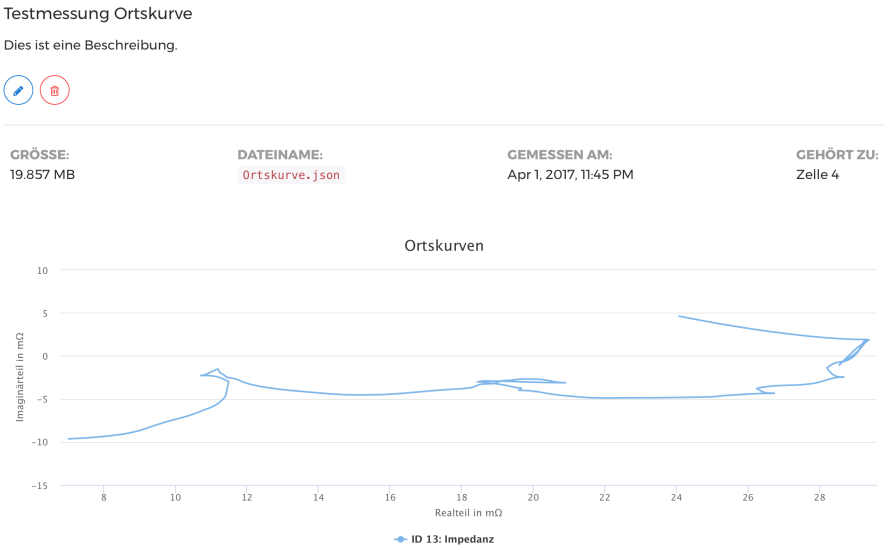

Abbildung 15: Detailansicht einer Ortskurve

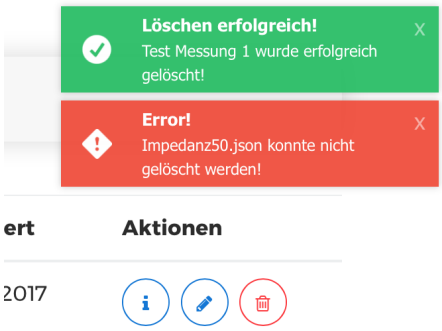

Abbildung 16: Notifications

Ein Plot einer Impedanzmessung ist in Abbildung 15 dargestellt. Die Koordinaten in der komplexen Ebene, wie auch die Frequenz sind am Cursor ablesbar.

3.7 NOTIFICATIONS

Bearbeitet oder löscht man Datensätze, oder fügt man neue durch Hochladen hinzu, wird das Ergebnis in Form von Notifications bzw. Fehlermeldungen im oberen rechten Bildschirmrand angezeigt. Dies ist in Abbildung 16 einmal für beide Fälle dargestellt.

3.8 REGISTRIERUNG

Neue Nutzer können nur durch bereits eingeloggte Nutzer mit Admin-Rechten hinzugefügt werden. Dafür kann man am oberen rechten Bildschirmrand auf das Plus klicken und gelangt zum Register-View, wie in Abbildung 17 dargestellt. Durch eintragen einer anderen Rolle,

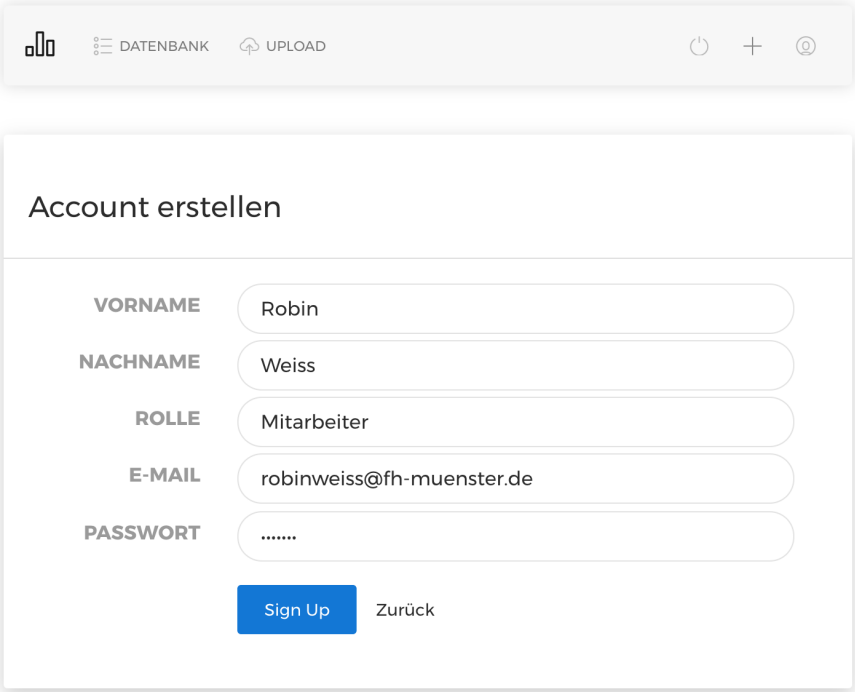

Abbildung 17: Register-View

wir zum Beispiel "Mitarbeiter", erhält der neue Nutzer keine Admin-Rechte und kann so auch keine neuen Nutzer hinzufügen.

4 ZUSAMMENFASSUNG

4.1 FAZIT

Das vorliegende Dokument beschriebt die Planung, die Implementierung und die Funktionalitäten einer Angular-Web-App zur Messdatenvisualisierung. Rückblickend war die Wahl der Frameworks eine geeignete. Eine Single-Page-Web-App bietet erstklassige User Experience und ist gleichzeitig effizient was HTTP-Traffic anbelangt. Angular ist hervorragend dokumentiert und es finden sich viele Codebeispiele und Tutorials im Internet. Die Programmiersprache C# ist eine elegante und leicht erlernbare Sprache für das Backend. Mit den während des Projekts erlernten Skills werden wir zukünftig weitere ähnliche Softwareprojekte realisieren.

4.2 AUSBLICK

Die App in ihrem jetzigen Zustand bietet grundlegende Funktionalitäten. Allerdings wären zahlreiche erweiternde Features möglich und hilfreich. So wäre die Löschung eines angelegten Users über das User Interface eine Erleichterung für den Administrator. Bislang müssen User, die keinen Zugriff mehr auf die App-Daten haben sollen, händisch aus der SQLite-Datenbank gelöscht werden. Die Modul- und Component-basierte Implementierung ermöglicht zukünftig eine einfache Erweiterung durch Drittentwickler.

LITERATUR

[1] ASP.NET. *Kestrel GitHub Repo*. [Accessed 01 October, 2017]. URL: https://github.com/aspnet/KestrelHttpServer.

[2] Creative Tim Design Agency. *Now UI Kit*. [Accessed 07 October, 2017]. URL: https://www.creative-tim.com/product/now-ui-kit.

[3] Google Inc. *Angular Frontend-Framework*. [Accessed 5 October, 2017]. URL: https://angular.io.

[4] Google Inc. *Angular GitHub Repo*. [Accessed 5 October, 2017]. URL: https://github.com/angular.

[5] Highsoft. *High Charts*. [Accessed 06 October, 2017]. URL: https://www.highcharts.com.

[6] Hwaci. *SQLite*. [Accessed 29 May, 2017]. URL: https://www.sqlite.org.

[7] Microsoft and the .NET Community. *Home repository for .NET Core*. [Accessed 28 April, 2017]. URL: https://github.com/dotnet/core.

[8] Microsoft. *Entity Framework Core*. [Accessed 29 May, 2017]. URL: https://docs.microsoft.com/en-us/ef/core/index.

[9] Robin Weiß. *Source Code Data Visualisation App*. [Accessed 29 October, 2017]. URL: https://github.com/rob-weiss/Data-Visualisation.

[10] Twitter Inc. *Bootstrap CSS Framework*. [Accessed 07 October, 2017]. URL: http://getbootstrap.com.