
Web-Benutzeroberfläche für Verteilnetze

Bachelorarbeit zur Erlangung des akademischen Grades

Bachelor of Science

im Studiengang Technische Informatik

an der Fakultät für Informations-, Medien- und Elektrotechnik

der Technischen Hochschule Köln

vorgelegt von: Parsa Amjadi

Adresse: parsa.amjadi@smail.th-koeln.de

eingereicht bei: Prof. Dr. Eberhard Waffenschmidt

Zweitgutachter: Markus de Koster

Frechen, 02.09.2025

Kurzfassung/Abstract

Ziel dieser Bachelorarbeit ist die Entwicklung einer intuitiven, webbasierten Benutzeroberfläche zur Visualisierung, Bearbeitung und Analyse von elektrischen Verteilnetzen.

Der Ausgangspunkt ist die Überlegung, dass viele bestehende Lösungen entweder stark auf Fachwissen angewiesen sind oder keine benutzerfreundliche Oberfläche bieten.

Die Anwendung richtet sich an Studierende und Dozenten, die mit OpenDSS-basierten Python-Programmen arbeiten. Mit dieser Web-Benutzeroberfläche können Verteilnetze in einen Browser geladen und angezeigt werden. Nutzer*innen können die Netze zudem analysieren, bearbeiten, speichern und Simulationsfunktionen über Zeit durchführen - ohne tiefgehende Programmierkenntnisse vorauszusetzen.

Die Arbeit beginnt mit den theoretischen Grundlagen verteilter Netze und graphbasierter Modellierung und einer Analyse vorhandener Visualisierungstools für Netzwerke.

Die Implementierung dieser Software basiert auf modernen Webtechnologien wie Vue, der Visualisierungsbibliothek D3 und der Firebase Datenbank. Der Fokus der Entwicklung lag auf einer modularen Architektur, einer benutzerfreundlichen Bedienoberfläche sowie der Möglichkeit, bestehende Netzprogramme unkompliziert über die standardisierte Schnittstelle anzubinden. Die Anwendung soll somit den Zugriff auf Simulationsdaten erleichtern und bestehende Workflows effizienter gestalten.

Inhalt

Kurzfassung/Abstract	I
Tabellenverzeichnis	III
Abbildungsverzeichnis	IV
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.2 Aufbau der Arbeit	4
2 Theoretischer Hintergrund	5
2.1 Elektrische Verteilnetze – Grundlagen und Herausforderungen.....	5
2.2 Digitalisierung und Simulationswerkzeuge im Energiesektor	6
2.3 Wichtige Netzelemente	7
2.4 Konzepte der Netzberechnung.....	9
2.5 Graphbasierte Modellierung	10
3 Stand der Technik	11
3.1 Überblick über bestehende Softwarelösungen	11
3.1.1 OpenDSS	13
3.1.2 DlgSILENT PowerFactory	14
3.1.3 pandapower.....	16
3.1.4 Power Grid Model.....	17
3.2 Vergleich bestehender Softwarelösungen	18
3.3 Anforderungen an Benutzeroberflächen.....	19
3.4 Eingesetzte Webtechnologien	21
4 Implementierung	21
4.1 Architektur der Anwendung	21
4.2 Seitenaufbau und Modellerung.....	26
4.3 Workflow.....	33
5 Bewertung der Anwendung	36
5.1 Vergleich zu bestehenden Lösungen	36
5.2 Benutzerfreundlichkeit.....	37
5.3 Bewertung anhand der „Goldenen Regeln“	38
5.4 Zusammenfassung der Bewertung.....	39
Literaturverzeichnis	40
Erklärung	41

Tabellenverzeichnis

Tabelle 1 – Netzelemente und einige ihrer Attribute.....	9
Tabelle 2 – Übersicht bestehender Softwarelösungen für Verteilnetze.....	19

Abbildungsverzeichnis

Abbildung 1: Ausschnitt aus dem OpenDSS Softwaretool.....	14
Abbildung 2: Schematisches Single Line Diagramm Ausschnitt aus dem PowerFactory Softwaretool.....	16
Abbildung 3: Aufbau von GridVisu (GridVisu in VSC.).....	25
Abbildung 4: Architekturübersicht der entwickelten Anwendung (erstellt.mit draw.io).....	26
Abbildung 5: Navigationsfluss und Beziehungen aller Komponenten der Anwendung (erstellt mit draw.io).....	27
Abbildung 6: Typischer Speicherungsablauf (Q.: GridVisu).....	28
Abbildung 7: Abbildung der JSON-Modellstruktur in eine interaktive Visualisierung mit D3 (Screenshots aus GridVisu und VSC.).....	29
Abbildung 8: Einige Knotentypen (Q.: GridVisu) 1.....	30
Abbildung 9: Element Dialog eines Generatorkno-tens (Q.: GridVisu).....	30
Abbildung 10: Messwert-Diagramm eines Knotenelements (Q.: GridVisu).....	31
Abbildung 11: Menüseite und Live-Änderung der „Node Voltage Over-load“ Farbe (Q.: GridVisu).....	32
Abbildung 12: Startseite der Anwendung und Auswahl eines Netzes (Q.: GridVisu).....	33
Abbildung 13: SimulationPage: Eingebblendete Labels und Farbanpassung des Netzes (Q.: GridVisu).....	33
Abbildung 14: ApplyDialog: Übersicht aller Änderungen am Grid vor dem persistenten Speichern (Q.: GridVisu).....	34
Abbildung 15: Visuell hervorgehobene Knoten mit kritischem Zustand (Q.: GridVisu).....	35

1 Einleitung

1.1 Motivation

Die Energieversorgung befindet sich aktuell in einem grundlegenden Umbruch. Durch die Energiewende werden immer mehr dezentrale Anlagen wie Photovoltaik- und Windkraftanlagen an das Stromnetz angeschlossen. Darunter fallen ebenso neue Verbraucher wie Ladeinfrastrukturen für Elektromobilität, Wärmepumpen oder Batteriespeicher, die flexibel und teilweise unvorhersehbar Strom beziehen oder einspeisen. Diese Entwicklungen führen zu einer deutlich höheren Komplexität in der Planung, Analyse und Steuerung von elektrischen Verteilnetzen.

Im Hinblick auf die Analyse von Verteilnetzen bedeutet dies folglich, dass zunehmend größere Datenmengen verarbeitet und verschiedenste Szenarien durchgespielt werden müssen. Dazu braucht es leistungsfähige und zugleich leicht zugängliche Werkzeuge. In der Praxis zeigt sich jedoch, dass viele der vorhandenen Softwarelösungen gewisse Einstiegshürden mit sich bringen.

Bewährte Programme wie beispielsweise Open-DSS sind hier ein gutes Beispiel. Sie setzen tiefergehende Kenntnisse in Programmierung und Simulation voraus. Selbst einfache Aufgaben wie das Visualisieren eines Netzes oder das Ausführen einer Standardsimulation erfordern Einarbeitungszeit und ein gutes Verständnis der internen Strukturen. Für Erfahrene mag das handhabbar sein, ist für Einsteiger*innen jedoch oft abschreckend. Hier zeigt sich eine Lücke: Es gibt einen Bedarf an benutzerfreundlichen, interaktiven Lösungen, die auch ohne umfangreiche Programmierkenntnisse genutzt werden können.

Darüber hinaus spielt auch die Forschung an Stromnetzen eine immer wichtigere Rolle. Die Integration erneuerbarer Energien oder die Entwicklung von Smart-Grid-Konzepten können nur dann praxisnah untersucht werden, wenn komplexe Netzzustände anschaulich genug visualisiert und interaktiv erfahrbar gemacht werden. Die steigende Systemkomplexität der Verteilnetze erfordert neue Werkzeuge, die über klassische Ansätze hinausgehen und flexible Betriebskonzepte unterstützen.¹ Damit sind leicht zugängliche Visualisierungs-

¹ <https://link.springer.com/article/10.1007/s12398-021-00316-x>

und Simulationsoberflächen nicht nur für Netzbetreiber und kleinere Versorger, sondern auch für Forschung und Lehre von zentraler Bedeutung.

Aus dieser Situation heraus entstand die Motivation für das vorliegende Projekt. Ziel dieser Bachelorarbeit ist es, eine webbasierte Benutzeroberfläche zu entwickeln, mit der elektrische Verteilnetze auf einfache Weise dargestellt, bearbeitet und simuliert werden können. Die Anwendung soll dazu beitragen, den Zugang zu etablierten Simulationswerkzeugen wie OpenDSS zu erleichtern und die Arbeit mit Netzdaten auch für Personen ohne tiefgehende Vorkenntnisse im Bereich der Programmierung greifbarer zu machen.

Ein weiterer Vorteil liegt darin, dass die entwickelte Lösung vollständig webbasiert ist. Dadurch kann sie plattformunabhängig auf jedem Computer mit einem aktuellen Browser genutzt werden – unabhängig davon, ob Windows, Mac oder Linux verwendet wird. Es ist keine Installation notwendig, was die Anwendung universell und einfach einsetzbar macht.

1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist die Entwicklung einer webbasierten Anwendung, die den Umgang mit elektrischen Verteilnetzen erleichtert und drei zentrale Anforderungen erfüllt.

Visualisierung

Netzmodelle sollen über eine modulare Schnittstelle in das Programm eingelesen werden können. Dabei wird vorausgesetzt, dass die Netze einer vorgegebenen Struktur entsprechen, um eine einheitliche und fehlerfreie Darstellung zu gewährleisten. Die eingelesenen Netze werden übersichtlich und interaktiv im Browser dargestellt. Dabei sollen alle relevanten Netzelemente wie Knoten, Leitungen, Lasten, Erzeuger und vieles mehr anschaulich sichtbar und klar voneinander unterscheidbar sein, einschließlich der Attributwerte einzelner Netzelemente.

Darüber hinaus wird den Anwendern die Möglichkeit gegeben, die Darstellung des Netzes nach ihren eigenen Vorstellungen und Vorlieben anzupassen. In den Einstellungen können beispielsweise Knotengrößen, Knotenfarben,

Kantenstärken oder auch Ereignisfarben individuell gewählt werden, wodurch sich die Visualisierung flexibel gestalten lässt.

Bearbeitung

Die Anwendung ermöglicht es Nutzern, bestehende Netzmodelle interaktiv zu bearbeiten. Der Schwerpunkt liegt dabei auf der Anpassung vorhandener Knoten, Leitungen, Lasten oder Erzeugern. So können beispielsweise Parameter wie Spannungen, Lastwerte oder andere Eigenschaften wie die X- und Y-Koordinaten der Netzelemente direkt verändert werden, ohne dass dafür externe Dateien editiert oder Programmcode angepasst werden muss.

Alle Änderungen am Netzmodell können flexibel gespeichert werden. Zum einen besteht die Möglichkeit, den aktuellen Stand der Designeinstellungen über den Browser im lokalen Speicher abzulegen und damit schnell wieder aufzurufen. Zum anderen können die bearbeiteten Netze persistent in einer eigenen Firebase-Datenbank gespeichert werden, bevor sie endgültig an den ursprünglichen Server übermittelt werden. Die Integration von Firebase ist modular gestaltet, sodass ein Wechsel auf alternative Lösungen unkompliziert und ohne großen Mehraufwand möglich ist. Auf diese Weise lassen sich Netzmodelle unabhängig vom jeweiligen Endgerät sichern und von verschiedenen Standorten aus abrufen. Dadurch können Anwender*innen sowohl kurzfristige Testszenarien durchführen als auch systematisch mit dauerhaft gespeicherten Netzständen arbeiten.

Simulation

Ein weiteres Ziel der Anwendung ist die Möglichkeit, Simulationsläufe direkt aus der Benutzeroberfläche zu starten. Dafür wird eine Schnittstelle integriert, über die regelmäßig aktualisierte Netzberechnungen abgerufen werden können. Diese Ergebnisse werden im zeitlichen Verlauf in Form von Diagrammen im Browser dargestellt.

Die Simulationsergebnisse wie beispielsweise Spannungsverläufe, Lastflüsse oder Netzverluste werden im Verteilnetz visualisiert. Alle jeweiligen Netzelemente werden unmittelbar visuell angepasst. Dadurch entsteht ein

direkter Bezug zwischen Modell und Ergebnis, was die Analyse deutlich erleichtert.

Die Simulationen werden zeitabhängig durchgeführt, um dynamische Entwicklungen im Netz (wie etwa Laständerungen oder Einspeiseschwankungen) zu untersuchen. Dies ermöglicht die Analyse von Szenarien über längere Zeiträume hinweg.

Durch diese Integration bildet die Anwendung den gesamten Ablauf von der Netzvisualisierung über die Bearbeitung bis hin zur Simulation in einer einzigen, durchgängigen Umgebung ab.

1.3 Aufbau der Arbeit

Die Arbeit ist in mehrere Kapitel gegliedert, welche den gesamten Entwicklungsprozess berücksichtigen:

- **Kapitel 2 - Theoretischer Hintergrund:** Zunächst werden die Grundlagen elektrischer Verteilnetze vorgestellt. Dazu gehören typische Strukturen, wichtige Netzelemente sowie grundlegende Konzepte wie Lastflüsse oder Einspeisungen. Außerdem wird die graphbasierte Modellierung als Methode zur Abbildung von Netzen erläutert, welche eine zentrale Rolle für die spätere Visualisierung spielt.
- **Kapitel 3 - Stand der Technik:** In diesem Kapitel werden bestehende Softwarelösungen und Visualisierungstools auf dem Markt, welche für die Analyse und Darstellung von Verteilnetzen genutzt werden, untersucht und verglichen. Dabei wird aufgezeigt, welche Funktionen bereits vorhanden sind und wo ihre Grenzen liegen. Anschließend folgt ein Überblick über die verwendeten Webtechnologien, welche u.a. Vue, D3 und Firebase beinhalten.
- **Kapitel 4 - Implementierung:** Auf Basis der vorherigen Kapitel wird das Konzept der entwickelten Anwendung beschrieben. Dazu gehört die Gesamtarchitektur, das Datenmodell, der Aufbau der Benutzeroberfläche, jegliche Speicherungen und typische Workflow Szenarien.

- **Kapitel 5 - Bewertung:** Im Anschluss an die Umsetzung erfolgt eine Bewertung der entwickelten Anwendung. Dabei wird untersucht, inwiefern die gestellten Anforderungen erfüllt wurden, wie benutzerfreundlich die Oberfläche gestaltet ist und wo technische Grenzen bestehen. Zusätzlich werden Erweiterungsmöglichkeiten diskutiert und das Tool im Vergleich zu bestehenden Lösungen, welche u.a. in Kapitel 3 erwähnt werden, eingeordnet.
- **Kapitel 6 - Fazit und Ausblick:** Abschließend werden die wichtigsten Erkenntnisse der Arbeit zusammengefasst. Außerdem wird ein Ausblick auf mögliche Erweiterungen und zukünftige Entwicklungen gegeben, wie beispielsweise zusätzliche Analysefunktionen oder die Erweiterung bestehender Funktionen.

2. Theoretischer Hintergrund

In diesem Kapitel werden die Grundlagen behandelt, die für das Verständnis der Arbeit notwendig sind. Dazu gehören zunächst die Eigenschaften elektrischer Verteilnetze, ihre typischen Strukturen und die Netzelemente. Anschließend wird auf grundlegende Konzepte wie Lastflüsse und Einspeisungen eingegangen. Schließlich wird die graphbasierte Modellierung vorgestellt, die eine wichtige Grundlage für die spätere Visualisierung und Bearbeitung darstellt.

2.1 Elektrische Verteilnetze – Grundlagen und Herausforderungen

Die Energieversorgung ist hierarchisch aufgebaut und besteht aus Übertragungsnetzen auf Hoch- und Höchstspannungsebene sowie Verteilnetzen, die Mittel- und Niederspannung umfassen. Verteilnetze bilden den letzten Abschnitt dieser Kette. Sie sorgen dafür, dass elektrische Energie von Umspannwerken zu den Endverbrauchern gelangt – also zu Haushalten, Gewerbe oder auch Ladeinfrastrukturen für Elektromobilität. Strukturell handelt es sich dabei um verzweigte Systeme mit Netzknoten, Leitungen, Transformatoren und Schaltanlagen.

Durch die Energiewende geraten diese Netze zunehmend unter Druck. Auf der einen Seite wächst die Zahl dezentraler Einspeiser wie Photovoltaik- und Windkraftanlagen, auf der anderen Seite steigt der Verbrauch durch neue Technologien wie Wärmepumpen und Elektromobilität. Dieses Zusammenspiel führt zu komplexeren Lastflüssen und erhöht die Wahrscheinlichkeit kritischer Zustände. Typisch sind zum Beispiel Überlastungen von Leitungen und Transformatoren oder Spannungsabweichungen außerhalb zulässiger Grenzen.

Die Verteilnetzstudie NRW kommt zu dem Ergebnis, dass allein in Nordrhein-Westfalen bis 2040 Investitionen von mehr als 15 Milliarden Euro nötig sein werden, um die wachsenden Anforderungen abzufedern. Gleichzeitig wird deutlich, dass digitale Technologien und intelligente Steuerung große Entlastungspotenziale bieten wie z.B. durch eine koordinierte Steuerung von Ladevorgängen bei Elektrofahrzeugen.²

2.2 Digitalisierung und Simulationswerkzeuge im Energiesektor

Die Energiewende verlagert viele Herausforderungen ins Verteilnetz. Dort sind die meisten dezentralen Erzeugungsanlagen angeschlossen und mit der zunehmenden Elektromobilität steigt die Belastung weiter. Um die gesteigerte Komplexität beherrschen zu können, ist eine stärkere Digitalisierung notwendig.³

Eine zentrale Rolle spielen dabei simulationsbasierte Werkzeuge. Sie erlauben es, Betriebsszenarien zu berechnen, kritische Zustände sichtbar zu machen und stellen verschiedene Handlungsoptionen zur Verfügung. In der Forschung werden diese Ansätze genutzt, um neue Konzepte wie z.B. Sektorenkopplung, Smart Grids oder resiliente Netzstrukturen zu untersuchen. Resilienz beschreibt die Fähigkeit eines Systems, nach Störungen schnell wieder in einen stabilen Betriebszustand zurückzukehren, und wird zunehmend als Schlüsselfaktor moderner Energiesysteme betrachtet.⁴

Für die Darstellung solcher Netze werden in der Praxis meist Einliniendiagramme (bzw. Single-Line-Diagrams) verwendet. Sie abstrahieren die Netzstruktur, indem

² (Verteilnetzstudie NRW 2021) - <https://www.wirtschaft.nrw/pressemitteilung/verteilnetzstudie-nrw>

³ (KIT – Institut für Elektroenergiesysteme und Hochspannungstechnik) - <https://www.ieh.kit.edu/2830.php>

⁴ <https://link.springer.com/article/10.1007/s12398-021-00316-x>

sie nur die wesentlichen Elemente wie Leitungen, Transformatoren oder Schaltstellen zeigen. Auch Kanten, die (hinsichtlich ihrer Phasen) normalerweise aus drei Leitungen bestehen, werden der Einfachheit halber durch eine einzige Linie dargestellt. Auch für diese Arbeit werden die Verteilnetze in Form von Single-Line-Diagrams vorgesehen. Dadurch entsteht eine kompakte und gut lesbare Form, die als Grundlage für klassische Berechnungen und Simulationen dient.

Hier setzen moderne Visualisierungsmethoden an. Sie bieten die Möglichkeit, Netze nicht nur in abstrahierter Form, sondern auch mit zeitabhängigen Zuständen und Interaktionen darzustellen. Dadurch wird es möglich, komplexe Abläufe wie Lastflussänderungen oder Netzüberlastungen anschaulich zu machen und sie im direkten Zugriff über eine grafische Benutzeroberfläche erfahrbar zu gestalten.

Die Digitalisierung der Energienetze bedeutet also nicht nur mehr Daten und neue technische Möglichkeiten, sondern erfordert auch Werkzeuge, die diese Daten verständlich aufbereiten. Interaktive, webbasierte Oberflächen können hier einen entscheidenden Beitrag leisten, da sie komplexe Netzzustände übersichtlich machen und gleichzeitig den Zugang zu Simulationen erleichtern.

2.3 Wichtige Netzelemente

Ein Verteilnetz besteht aus verschiedenen Komponenten, die in Modellen und Simulationen abgebildet werden:

Die **Knoten** sind die zentralen Verbindungspunkte bzw. Abzweige im Netz. An ihnen können Lasten oder Erzeuger angeschlossen sein. Sie sind über **Leitungen** verbunden, welche elektrische Energie zwischen den Knoten übertragen. In Netzen sind Leitungen oft zusätzlich mit **Schaltern** ausgestattet, die das temporäre Verbinden und Trennen bestimmter Netzabschnitte ermöglichen.

Die **Lasten** repräsentieren Verbraucher im Netz wie beispielsweise Haushalte, Gewerbebetriebe oder auch spezielle Verbraucher wie Ladestationen für Elektrofahrzeuge.

Erzeuger hingegen sind dezentrale Einspeiser wie z.B. Photovoltaikanlagen oder

Windkraftanlagen. Sie speisen elektrische Energie in das Netz ein und machen die Netzsituation dynamischer und schwerer vorhersehbar. **Transformatoren** verbinden Netzebenen mit unterschiedlichen Spannungen. Im Verteilnetz koppeln sie meist die Mittelspannungsebene mit der Niederspannungsebene und sorgen dafür, dass Energie effizient übertragen und an die Endverbraucher angepasst wird.

Diese Elemente bilden die Grundlage für die Netzmodelle, die später in der Anwendung visualisiert und bearbeitet werden. Jedes Netzelement wird durch bestimmte Parameter näher beschrieben, die sowohl für Simulation als auch für die Visualisierung eine Rolle spielen. Die wichtigsten Parameter sind in **Tabelle 1** zusammengefasst.

Netzelement	Attribut	Beschreibung
Knoten (Nodes)	ID	Eindeutige Bezeichnung zur Identifizierung eines Knotens
	Nennspannung („vn_kv“)	Typischerweise 0.4kV im Niederspannungsnetz
	Last („load“)	Bspw. Haus
	Erzeuger („generator“)	Bspw. PV, Wind, Speicher...
	Leistung	Wirkleistung (P) und Blindleistung (Q) am Knoten
Leitungen (Edges)	ID	(Unter allen Leitungen) einmaliger numerischer Wert

	Nennspannung	Spannung: Abhängig von angeschlossenen Knoten
	Schaltstellen („switch“)	Zum Öffnen & Schließen der Leitungen

Tabelle 1 – Netzelemente und einige ihrer Attribute

2.4 Konzepte der Netzberechnung

Die Lastflussberechnung gehört zu den grundlegenden Verfahren der Netzberechnung. Ziel ist es, die Spannungen an allen Netzknoten sowie Ströme und Leistungen in den Leitungen für einen bestimmten Betriebszustand zu bestimmen. Auf diese Weise lässt sich beurteilen, ob die Betriebsmittel innerhalb zulässiger Grenzen arbeiten oder ob kritische Zustände wie Spannungsabweichungen oder Überlastungen auftreten.

Da die Gleichungen zwischen Spannung, Strom und Leistung nichtlinear sind, werden in der Praxis numerische Näherungsverfahren eingesetzt. Häufig genutzt werden iterative Ansätze wie das Newton-Raphson-Verfahren.⁵ Sie ermöglichen es, auch große Netze mit vielen Knoten und Verbrauchern über die Zeit zu berechnen.

Neben dem Newton-Raphson-Verfahren wird auch das iterative Gauss-Seidel-Verfahren für Lastflussberechnungen eingesetzt. Letzteres ist rechnerisch einfacher, konvergiert aber langsamer und eignet sich daher eher für kleinere Netze. Newton-Raphson benötigt pro Iteration mehr Aufwand, erreicht dafür aber eine deutlich schnellere und stabilere Konvergenz.

Die Ergebnisse einer Lastflussberechnung sind insbesondere für Verteilnetze wichtig, da hier durch dezentrale Einspeisung und neue Verbraucher wie Elektromobilität veränderte Lastflüsse entstehen. Mithilfe der Berechnung

⁵ <https://de.wikipedia.org/wiki/Lastflussberechnung>

können Szenarien wie Spitzenlast, hohe PV-Einspeisung oder das gleichzeitige Laden von Fahrzeugen bewertet werden. Sie bildet damit die Grundlage für Netzplanung und Betriebsführung.

Neben der reinen Berechnung von Strömen und Spannungen ist die Lastflussanalyse auch ein wichtiges Werkzeug zur Beurteilung der Netzstabilität. Darunter versteht man die Fähigkeit eines Netzes, trotz schwankender Einspeisungen und Lasten innerhalb zulässiger Spannungs- und Frequenzgrenzen zu bleiben und nach Störungen wieder einen stabilen Betriebszustand einzunehmen. Gerade in modernen Verteilnetzen mit dezentralen erneuerbaren Energien und Elektromobilität spielt die Sicherstellung der Spannungsstabilität eine zentrale Rolle.

2.5 Graphbasierte Modellierung

Zur Abbildung elektrischer Verteilnetze wird in dieser Arbeit graphbasierte Modellierung verwendet. Ein Graph besteht allgemein aus Knoten (Nodes) und Kanten (Edges). Übertragen auf Verteilnetze repräsentieren die Knoten einzelne Netzelemente wie beispielsweise Lasten oder Erzeuger, während die Kanten die Leitungen bzw. Verbindungen zwischen diesen Elementen darstellen. Hinsichtlich der Darstellung von Transformatoren gibt es in bestehenden Softwarelösungen unterschiedliche Modellierungsarten. Während sie teils als eigene Netzelemente abgebildet werden, erfolgt in dieser Arbeit die Modellierung einheitlich als Kanten

Eine Art der Modellierung von Verteilnetzen ist die Baumstruktur. Ein Baum ist ein spezieller Graph ohne Zyklen, bei dem es einen eindeutigen Startpunkt (Wurzelknoten) gibt, von dem aus sich alle weiteren Elemente verzweigen. Diese Struktur passt gut zu typischen Verteilnetzen, die in der Praxis häufig strahlenförmig aufgebaut sind und von einer zentralen Spannungsquelle über mehrere Abzweige bis zu den Endverbrauchern reichen.

Ein wesentlicher Vorteil dieser Modellierung liegt in der guten Verknüpfbarkeit mit Visualisierungsbibliotheken wie D3. Auch wenn Knoten keine festen Koordinaten besitzen, können ihre Positionen anhand der zugrunde liegenden Graphstruktur automatisch berechnet werden. Dadurch entsteht eine übersichtliche

Darstellung, die an der „Wurzel“ beginnt und sich hierarchisch in Kinderknoten aufgliedert.

Auch wenn in dieser Arbeit für die automatische Berechnung der Netzkoordinaten eine Baumstruktur angenommen wird, lässt sich das Modell grundsätzlich auf allgemeinere Graphen mit Zyklen erweitern. Bibliotheken wie D3 unterstützen auch die Visualisierung zyklischer Netzwerke, sodass sich zukünftig beispielsweise vermaschte Netze oder komplexere Topologien darstellen lassen können.

Während **Kapitel 2** die theoretischen Grundlagen und konzeptionellen Methoden behandelt hat, richtet sich der Blick in Kapitel 3 auf den Stand der Technik. Dort werden bestehende Softwarelösungen und Visualisierungstools vorgestellt und miteinander verglichen, um einzuordnen, welche Funktionen bereits etabliert sind und wo die in dieser Arbeit entwickelte Weboberfläche neue Impulse setzen kann.

3. Stand der Technik

In diesem Kapitel werden zunächst gängige Softwarelösungen vorgestellt, die in Forschung und Praxis zur Modellierung, Analyse und Visualisierung elektrischer Netze eingesetzt werden. Anschließend folgt ein Überblick über die im Rahmen dieser Arbeit verwendeten Webtechnologien, die eine moderne und interaktive Umsetzung ermöglichen.

3.1 Überblick über bestehende Softwarelösungen

Die heute verfügbaren Programme zur Netzmodellierung und -simulation lassen sich grob in zwei Gruppen einteilen: kommerzielle Systeme, die vor allem in industriellen Anwendungen genutzt werden, und Open-Source-Werkzeuge, die insbesondere in Forschung und Lehre verbreitet sind. Typische Vertreter kommerzieller Lösungen sind etwa [DIgSILENT PowerFactory](#) oder [PSS®E](#) von Siemens, die eine breite Funktionalität, aber auch Lizenzkosten mit sich bringen.

Zu den Open-Source-Werkzeugen zählen vor allem [OpenDSS](#), [pandapower](#) und neuere Projekte wie das [Power Grid Model](#).

- **OpenDSS** wurde von der Electric Power Research Institute (EPRI) entwickelt und ist ein etabliertes, frei verfügbares Tool, das unter anderem für Lastfluss- und Kurzschlussberechnungen eingesetzt wird. Es ist leistungsfähig, erfordert jedoch tiefere Kenntnisse in Skriptsprachen und Simulation, da eine komfortable grafische Oberfläche fehlt.
- **DlgSILENT PowerFactory** ist ein kommerzielles Werkzeug, das sich durch eine breite Modellbibliothek und eine moderne grafische Benutzeroberfläche auszeichnet. Es ist in der industriellen Praxis weit verbreitet und unterstützt neben klassischen Lastfluss- und Stabilitätsanalysen auch komplexe Simulationen. Nachteilig ist zum einen die Lizenzpflicht und zum anderen die vergleichsweise hohe Einstiegshürde für neue Nutzer.
- **pandapower** ist ein Python-basiertes Open-Source-Tool, das 2016 an der Universität Kassel und am Fraunhofer IEE entwickelt wurde. Es kombiniert die Bibliotheken *pandas* und *pypower* und ermöglicht damit eine effiziente Zustandsberechnung, Netzplanung und Optimierung. Es hat sich inzwischen als ein leistungsstarkes und anerkanntes Werkzeug in Forschung und Praxis etabliert.⁶
- Das **Power Grid Model** ist ein Projekt der Linux Foundation (LF Energy), das speziell auf Offenheit, Modularität und Skalierbarkeit ausgelegt ist. Es bietet eine moderne C++-basierte Kernarchitektur und zielt darauf ab, flexible Schnittstellen für Simulation und Integration in andere Softwareumgebungen bereitzustellen.⁷

Allen Systemen gemeinsam ist, dass sie grundlegende Verfahren wie Lastfluss- und Kurzschlussberechnungen oder Stabilitätsanalysen ermöglichen. Unterschiede bestehen jedoch in der Benutzerfreundlichkeit, in den Visualisierungsmöglichkeiten und in der Offenheit gegenüber Schnittstellen. Diese Unterschiede sind entscheidend für die Wahl einer geeigneten Lösung.

⁶ (50komma2 - 2024). <https://www.50komma2.de/digitalisierung/pandapower-rekord/>

⁷ <https://lfenergy.org/projects/power-grid-model/>

3.1.1 OpenDSS

OpenDSS (Open Distribution System Simulator) ist eine vom Electric Power Research Institute (EPRI) entwickelte Simulationsumgebung, die speziell für Verteilnetze konzipiert wurde. Das Programm zeichnet sich durch eine offene Architektur und die Möglichkeit aus, detaillierte Modelle von Netzelementen zu erstellen. Typische Anwendungsfälle sind Lastflussberechnungen, Zeitreihenanalysen, Untersuchungen zur Integration erneuerbarer Energien sowie Bewertungen der Netzstabilität im Nieder- und Mittelspannungsbereich.⁸

Ein großer Vorteil von OpenDSS liegt in seiner Flexibilität. Netzmodelle werden textbasiert beschrieben und können über COM- oder Python-Schnittstellen in andere Programme eingebunden werden. Damit eignet sich das Tool besonders für Forschungsprojekte, bei denen eigene Erweiterungen oder Automatisierungen implementiert werden sollen.

Allerdings bringt dieser Ansatz auch Nachteile mit sich: Die Bedienung erfolgt vollständig skriptbasiert und erfordert einen hohen Einarbeitungsaufwand. Netzstrukturen müssen manuell in Textdateien erstellt werden, Ergebnisse erscheinen überwiegend in Tabellenform, und die vorhandene grafische Benutzeroberfläche ist nur rudimentär und nicht auf moderne Benutzerfreundlichkeit ausgelegt. Ein Ausschnitt dieser (veralteten) Oberfläche ist in *Abbildung 1* zu sehen.

⁸ (EPRI 2024) - <https://opendss.epri.com/IntroductiontoOpenDSS.html>

- Bild für öffentliche Version aus urheberrechtlichen Gründen entfernt. -

*Abb. 1 – Ausschnitt aus dem OpenDSS Softwaretool
(<https://sourceforge.net/projects/electricdss/>)*

3.1.2 DIgSILENT PowerFactory

DIgSILENT PowerFactory ist eines der am weitesten verbreiteten kommerziellen Softwarewerkzeuge zur Modellierung und Analyse elektrischer Energieversorgungssysteme. Es wird sowohl in Forschung als auch in der industriellen Praxis eingesetzt und deckt ein breites Spektrum von Funktionen ab – von klassischen Lastfluss- und Kurzschlussberechnungen bis hin zu erweiterten Simulationen wie dynamischen Stabilitätsanalysen, Schutzprüfungen oder Marktmodellen für Energiesysteme.⁹ PowerFactory eignet sich nicht nur für

⁹ (DIgSILENT 2024) - <https://www.digsilent.de/de/powerfactory.html>

Übertragungsnetze, sondern auch speziell für Mittel- und Niederspannungsnetze, womit es eine direkte Konkurrenz zu OpenDSS darstellt.

Ein zentrales Merkmal von PowerFactory ist seine umfangreiche Modellbibliothek, die es erlaubt, nahezu alle relevanten Netzelemente detailliert nachzubilden – von einfachen Leitungen und Transformatoren bis zu komplexen Modellen von Generatoren, Umrichtern und Schutzsystemen. Ergänzt wird dies durch leistungsfähige Simulationsverfahren, die über klassische stationäre Analysen hinausgehen und unter anderem transiente Vorgänge und Langzeitsimulationen abbilden können.

In technischer Hinsicht basiert PowerFactory auf einer integrierten Datenbankarchitektur, die eine konsistente Verwaltung von Netzmodellen ermöglicht. Die Bedienung erfolgt über eine moderne grafische Benutzeroberfläche. Die Netzelemente können per Drag-and-Drop modelliert werden, was in [Abbildung 2](#) weiter unten gut zu erkennen ist. Ergebnisse wie Spannungsprofile, Leistungsverläufe oder Fehlerströme werden visuell aufbereitet und erlauben eine intuitive Interpretation. Zusätzlich stellt PowerFactory Schnittstellen zu den bekannten Programmiersprachen Python, MATLAB und C++ bereit, sodass Automatisierungen und eigene Skripte in bestehende Workflows eingebunden werden können.

Ein wesentlicher Vorteil von PowerFactory liegt in seiner Benutzerfreundlichkeit. Die grafische Oberfläche erleichtert die Modellierung erheblich im Vergleich zu rein skriptbasierten Umgebungen wie OpenDSS. Allerdings bringt diese Stärke auch Nachteile mit sich: Als kommerzielle Software ist PowerFactory lizenzpflichtig, was den Zugang für Studierende oder kleinere Forschungseinrichtungen einschränken kann. Zudem ist der Funktionsumfang zwar sehr groß und gleichzeitig auch komplex, was eine gewisse Einarbeitung erfordert.

Zusammenfassend lässt sich festhalten, dass DigSILENT PowerFactory durch seine große Funktionsbreite und die professionelle Visualisierung eine etablierte Lösung für Netzberechnungen ist. Für praxisnahe Anwendungen, in denen Effizienz und Benutzerfreundlichkeit im Vordergrund stehen, bietet PowerFactory klare Vorteile gegenüber OpenDSS. Jedoch sollte man anmerken, dass interaktive, webbasierte Visualisierungen (wie in modernen Tools mit D3) nicht

vorgesehen sind und die Software an Windows gebunden ist. Die Software richtet sich primär an professionelle Anwender mit entsprechender Erfahrung.

- Bild für öffentliche Version aus urheberrechtlichen Gründen entfernt. -

Abb. 2 – Schematisches Single Line Diagramm Ausschnitt aus dem PowerFactory Softwaretool¹⁰

3.1.3 pandapower

pandapower ist ein Open-Source-Werkzeug, das 2016 an der Universität Kassel und am Fraunhofer IEE entwickelt wurde. Ziel war es, eine leicht zugängliche, aber dennoch leistungsstarke Umgebung für die Modellierung, Analyse und Optimierung elektrischer Netze bereitzustellen. Die Software kombiniert die Python-Bibliotheken *pandas* und *pypower*, wodurch sowohl eine effiziente Datenverarbeitung als auch hohe Rechengeschwindigkeiten erreicht werden. Durch diese Architektur ist *pandapower* plattformunabhängig einsetzbar und lässt sich flexibel in bestehende Workflows integrieren.

In den letzten Jahren hat sich *pandapower* zu einem der bekanntesten Open-Source-Tools für Stromnetze entwickelt. 2024 überschritt es die Marke von über 500.000 Downloads, was die breite Nutzung in Forschung und Praxis

¹⁰ <https://www.digsilent.de/de/fachbeitraege-reader-pf-de/network-model-diagrams-in-powerfactory-2.html>

unterstreicht.¹¹ Eingesetzt wird es unter anderem von Netzbetreibern wie 50Hertz in Deutschland oder UK Power Networks in Großbritannien, beispielsweise zur Zustandsberechnung, Netzplanung und zur Integration kohlenstoffarmer Technologien.¹²

Besonders hervorgehoben wird die einfache Bedienung kombiniert mit einem hohen Automatisierungsgrad. Neben klassischen Lastfluss- und Kurzschlussanalysen unterstützt pandapower auch automatisierte Netzplanungen und sektorenübergreifende Optimierungen, beispielsweise durch die Kopplung von Strom-, Wärme- und Gasnetzen. Zusätzlich wurden Converter entwickelt, um Modelle aus anderen Umgebungen wie PowerFactory oder CIM-Formaten zu übernehmen.

Ein weiterer Vorteil liegt in der aktiven Community und den kontinuierlichen Weiterentwicklungen. Regelmäßige Workshops und Schulungen fördern den Wissensaustausch, und mit der Gründung des Spin-offs *retoflow* im Jahr 2021 wurde die Entwicklung zusätzlicher Funktionen wie ein interaktiver Netzeditor vorangetrieben. Damit positioniert sich pandapower als vielseitiges Werkzeug, das sowohl für Forschung und Lehre als auch für praktische Anwendungen in der Netzplanung und -optimierung geeignet ist.

3.1.4 Power Grid Model

Das Power Grid Model ist ein junges Open-Source-Projekt, welches von der LF Energy (Linux Foundation) entwickelt wird. Ziel der Software ist es, eine flexible, modulare und skalierbare Umgebung für die Modellierung und Analyse von Stromnetzen bereitzustellen. Die Implementierung erfolgt in C++ und ist damit auf hohe Rechengeschwindigkeit und Effizienz ausgerichtet.

Im Unterschied zu vielen etablierten Tools setzt Power Grid Model von Beginn an auf Offenheit und Interoperabilität. Die Architektur ist so gestaltet, dass Schnittstellen zu externen Anwendungen einfach integriert werden können. Damit eignet sich das Werkzeug nicht nur für klassische Netzberechnungen,

¹¹ (Fraunhofer IEE, 2024) - <https://www.iee.fraunhofer.de/de/presse-infothek/Presse-Medien/2024/pandapower-open-source-tool-erreicht-500k-downloads.html>

¹² (50Komma2, 2024) - <https://www.50komma2.de/digitalisierung/pandapower-rekord/>

sondern auch als Baustein in größeren Softwaresystemen, beispielsweise in Forschungsprojekten oder bei der Kopplung verschiedener Energiesektoren.

Ein Vorteil liegt in der Modularität: Anwender können einzelne Funktionen oder Berechnungsroutinen gezielt einsetzen und erweitern. Dadurch ist die Software auch für zukünftige Entwicklungen interessant, bei denen Flexibilität und Erweiterbarkeit eine zentrale Rolle spielen. Allerdings befindet sich das Projekt noch im Aufbau, sodass der Funktionsumfang derzeit nicht mit etablierten Werkzeugen wie pandapower oder PowerFactory vergleichbar ist.

Trotzdem zeigt sich Power Grid Model als vielversprechende Ergänzung im Bereich der Open-Source-Tools für Stromnetze, vor allem durch seine moderne Codebasis und den starken Fokus auf Community-getriebene Weiterentwicklung.

13

3.2 Vergleich bestehender Softwarelösungen

Softwaretool	Typ	Stärken	Schwächen	Verbreitung
OpenDSS	Open Source (EPRI)	Flexibel, detaillierte Modellierung, Schnittstellen (COM, Python)	GUI altmodisch, nicht komfortabel, rein skriptbasiert	Frei verfügbar, in Forschung genutzt, weniger bekannt
PowerFactory	Kommerziell (DlgSILENT)	Modellbibliothek, moderne GUI	Lizenzkosten, steile Lernkurve, wenig Offenheit für Integration in Python-Workflows	Weltweit sehr verbreitet
pandapower	Open Source (Fraunhofer IEE)	Validierte Modelle, Python-basiert, Automatisierung, Community	Fehlende GUI, Nutzung ausschließlich über Python-Code	Open Source: wachsende Community

¹³ <https://fenergy.org/projects/power-grid-model/>

Power Grid Model	Open Source (LF Energy)	Moderne C++ Architektur, modular, offen, leistungsstark	Funktionsumfang begrenzt	Open Source: wachsende Community
------------------	-------------------------	---	--------------------------	----------------------------------

Tabelle 2 –Übersicht bestehender Softwarelösungen für Verteilnetze

Tabelle 2 fasst die wesentlichen Eigenschaften der vier betrachteten Werkzeuge zusammen und verdeutlicht ihre jeweiligen Stärken und Schwächen. Dadurch wird sichtbar, dass jedes Tool spezifische Einsatzfelder adressiert und kein System alle Anforderungen gleichermaßen abdeckt. Letztlich spielt bei der Auswahl auch immer die persönliche Präferenz und der konkrete Anwendungskontext eine Rolle.

3.3 Anforderungen an Benutzeroberflächen

Neben den funktionalen Fähigkeiten spielt die Gestaltung der Benutzeroberfläche eine zentrale Rolle. Gerade bei komplexen technischen Tools entscheidet die Usability darüber, ob Anwender effizient arbeiten können oder durch hohe Einstiegshürden abgeschreckt werden.

Ein grundlegender Referenzrahmen wird durch die ISO-Norm 9241-11 bereitgestellt. Sie definiert Usability anhand von drei Dimensionen:¹⁴

- **Effektivität:**
Die Aufgaben können vollständig und korrekt erledigt werden.
- **Effizienz:**
Die Nutzung geschieht mit akzeptablem Aufwand und ohne unnötige Umwege.
- **Zufriedenheit:**
Die Interaktion wird von den Anwendern positiv erlebt, ohne Frustration oder Überforderung

Diese Kriterien lassen sich unmittelbar auf Netztools übertragen. Ein Werkzeug zur Visualisierung von Verteilnetzen muss also ermöglichen, Netze zuverlässig

¹⁴ https://de.wikipedia.org/wiki/ISO_9241

zu modellieren, Zustände effizient zu berechnen und die Ergebnisse so darzustellen, dass sie für die Anwender nachvollziehbar sind.

Darüber hinaus liefern die von **Shneiderman (1992)** formulierten „Eight Golden Rules“ bis heute zentrale Prinzipien für die Gestaltung interaktiver Systeme. Besonders relevant sind:¹⁵

- **Konsistenz:** gleiche Aktionen führen zu gleichen Ergebnissen,
- **klare Rückmeldungen:** das System zeigt sofort, welche Eingabe verarbeitet wurde,
- **Fehlerfreundlichkeit:** fehlerhafte Eingaben können einfach korrigiert werden,
- **kontrollierbare Abläufe:** Nutzer behalten die Kontrolle und können Prozesse jederzeit abbrechen oder anpassen.

Übertragen auf Netztools bedeutet das: eine klare, konsistente Bedienung ist essenziell, um komplexe Zusammenhänge wie Lastflüsse oder kritische Zustände verständlich zu machen. Ebenso wichtig ist, dass die Bedienoberfläche nicht unnötig komplex wird. Fehler sollen so früh wie möglich abgefangen werden, sodass der Nutzer kaum die Möglichkeit hat, kritische Eingaben falsch zu machen – das System unterstützt und hält ihn aktiv davon zurück. Dialoge sind dabei klar strukturiert, geben freundliche und präzise Rückmeldungen und helfen, sich jederzeit schnell zurechtzufinden. Übersichtlichkeit und eine einfache Navigation sind zentrale Voraussetzungen, damit auch komplexe Netzanwendungen sicher und effizient genutzt werden können.

3.4 Eingesetzte Webtechnologien

In dieser Arbeit wird ein Technologie-Stack verwendet, der vor allem drei zentrale Komponenten umfasst: Vue, D3 und Firebase.

- **Vue** ist ein progressives JavaScript-Framework, das für die Entwicklung modularer und dynamischer Benutzeroberflächen genutzt wird. Durch die

¹⁵ <https://www.cs.umd.edu/~ben/goldenrules.html>

klare Trennung von Logik, Daten und Darstellung lassen sich komplexe Anwendungen strukturiert und nachhaltig umsetzen.

- **D3** ist eine leistungsfähige Bibliothek zur Datenvisualisierung im Browser. Sie bietet flexible Möglichkeiten zur Darstellung von Graphen, Diagrammen und interaktiven Strukturen. Für die in dieser Arbeit entwickelte Anwendung ist D3 besonders relevant, da sich damit die baumartige Struktur von Verteilnetzen anschaulich und interaktiv abbilden lässt. Funktionen wie Drag-and-Drop, dynamisches Ein- und Ausblenden von Attributen oder die direkte Visualisierung von Netzparametern sind so einfach umsetzbar.
- **Firestore** dient als cloudbasierte Plattform für das Zwischenspeichern und Synchronisieren von Verteilnetzen. Dadurch können Netzmodelle plattformunabhängig gespeichert, geladen und mit verschiedenen Endgeräten genutzt werden.

Zusammengenommen schaffen diese Technologien eine moderne, interaktive und plattformunabhängige Lösung, die typische Schwächen klassischer Netztools adressiert und Anwendern einen deutlich einfacheren Einstieg ermöglicht.

4. Implementierung

4.1 Architektur der Anwendung

Die entwickelte Software, nachfolgend GridVisu genannt, besteht aus einem klassischen **Frontend** und einem angebundenen **Backend**. Das Frontend wurde mit Vue umgesetzt und enthält alle zentralen Module für Darstellung, Logik und Speicherung. Das Backend ist in Form eines bereitgestellten Dummy-Servers realisiert, der eine REST-Schnittstelle anbietet und vollständig in Python geschrieben ist. Darüber können typische CRUD-Operationen auf Netzmodellen ausgeführt und zeitabhängige Messwerte für Knoten und Kanten abgefragt werden.

Im Frontend kommen verschiedene Bausteine zum Einsatz, welche weiter unten in [Abbildung 3](#) zu sehen sind. Dies ist ein Ausschnitt aus der genutzten Entwicklungsumgebung Visual Studio Code.

Pinia Stores bilden die interne, lokale Datenhaltung des Frontends:

- Der **gridStore** bildet die zentrale Schnittstelle für das Laden, Speichern und Verwalten von Netzen. Für die Startseite steht eine Methode zur Verfügung, die lediglich Vorschaudaten eines Netzes lädt. Im Gegensatz dazu wird für die Simulationsansicht die vollständige Topologie benötigt, welche über die initiale *loadGrid* Methode bereitgestellt wird. Beim Aufruf von *loadGrid* wird die vollständige Topologie (*gridTopology*) zunächst vom Server geladen und anschließend im *gridStore* weiterverarbeitet. Dazu gehören das Ergänzen zusätzlicher Attribute wie *meas_voltage* oder *meas_current*, in denen aktuelle Messwerte abgelegt werden, um sie unmittelbar im Grid darstellen zu können. Zudem überprüft der *gridStore* die Koordinaten aller Knoten: Fehlen diese, werden sie mithilfe eines D3-Hierarchy-Layouts automatisch berechnet. Für diese Methode wird auf eine Composable Datei zugegriffen. Auf Grundlage dieser Positionen wird der Viewport bestimmt, sodass das Netz vollständig und übersichtlich im Frontend angezeigt werden kann. Damit die Daten nicht mehrfach vom Server abgerufen werden müssen, werden sowohl die *gridId* als auch die *gridTopologyId* intern zwischengespeichert.

Bei der Speicherung unterscheidet der *gridStore* zwischen zwei Varianten: Einerseits kann ein Grid in einer persönlichen Firebase-Instanz abgelegt werden, andererseits lässt es sich dauerhaft auf dem Hauptserver sichern. Wird ein Grid an den Server übertragen, wird die zugehörige Firebase-Instanz automatisch gelöscht, sodass die Datenbestände jederzeit synchron bleiben. Zusätzlich besteht die Möglichkeit, Netze auf dem Server zu löschen oder über *resetGrid* sowie durch das manuelle Entfernen von Firebase-Einträgen den synchronen Zustand wiederherzustellen.

Neben diesen Kernfunktionen sind auch Methoden zum Abruf von Messdaten integriert, wodurch die im Grid hinterlegten Parameter für Auswertungen genutzt werden können. Abschließend bietet der

gridStore Mechanismen zum Import und Export von Netzen in Form von JSON-Dateien, sodass Grids unkompliziert in der Simulationsansicht hochgeladen oder extern weitergegeben werden können.

- Der **settingsStore** hingegen verwaltet alle Initialwerte für die Designeinstellungen der Visualisierung. Hier lassen sich zentrale Aspekte wie Knotenfarbe und -größe, Kantendicke und -farbe oder auch die Darstellung von Generatoren individuell anpassen. Über Colorpicker und Slider ist die Bedienung einfach und flexibel. Zusätzlich können funktionale Parameter gesetzt werden, etwa ein Schwellenwert für Node-Overloads in Prozent oder ein Referenzwert für die Spannung (Standard: 230 V).

Alle Einstellungen werden direkt nach einer Änderung im localStorage gespeichert und unmittelbar in der GridVisu-Anzeige übernommen. Damit bleibt jede Anpassung jederzeit sichtbar und nachvollziehbar. Zudem besteht die Möglichkeit, Einstellungsprofile zu exportieren und später wieder zu importieren, sodass wiederholte manuelle Eingaben entfallen. Falls gewünscht, können die Einstellungen auch jederzeit auf die Standardwerte zurückgesetzt werden. Auf diese Weise bietet der settingsStore maximale Kontrolle und gleichzeitig Komfort bei der Gestaltung der Visualisierung.

Der **Service-Ordner** mit der **RESTfulService** Datei ist die einzige Schnittstelle zum Backend; der gridStore greift ausschließlich darüber zu. Die Services kapseln alle HTTP-Requests und liefern normalisierte Datenstrukturen zurück. Da hierüber alle API-Aufrufe laufen, bleibt das Backend sauber entkoppelt und der *gridStore* arbeitet nur mit konsistenten, geprüften Antworten.

Der **Vue Router** steuert die Navigation zwischen den Hauptseiten und definiert eindeutige Pfade. **HomePage** listet alle Netze; bei Auswahl wird per Route-Parameter (*gridId* und *topologyId*) zur **SimulationPage** gewechselt. Dort wird auf die meisten Composables zugegriffen, insbesondere für die graphische Darstellung und UI-Updates. Die **GlobalMenu** Komponente stellt die App-Einstellungen dar, welches direkt am **settingsStore** hängt.

Composables sind Vue-typische JavaScript-Dateien, in denen wiederverwendbare Logik gekapselt wird. Dazu gehören Rendering-Funktionen mit D3, Event-Handling, Berechnungen, Parser und Formatter, ebenso wie Zoom- und Positionierungslogik oder Dialog-Utilities. Alle Composables greifen auf einen gemeinsamen Kontext zu, der nach dem Prinzip des Shared State organisiert ist. Dieser Kontext wird einmalig in der *SimulationPage* erzeugt und anschließend allen Komponenten zur Verfügung gestellt, sodass sämtliche Teile der Anwendung konsistent auf dieselben Zustände und Methoden zugreifen können.

In der Anwendung kommen mehrere zentrale Composables zum Einsatz, die jeweils spezifische Aufgaben abdecken und so eine klare Trennung der Logik ermöglichen. Das *useGridRenderer* ist für das Rendern der Netzstruktur zuständig und übernimmt insbesondere das Zeichnen von Knoten, Kanten und Labels mit D3. Ergänzend dazu steuert das *useGridEvents* die Benutzerinteraktionen wie Klicks, Drag-and-Drop oder Hover-Effekte. Allgemeine Rechenoperationen zur Verarbeitung, Filterung und Formatierung von Daten sind im *useGridUtils* gebündelt. Für die Positionierung der Knoten sorgt das *useGridNodePositioner*, das unter anderem Layout-Algorithmen wie D3-Hierarchy verwendet. Die Navigation innerhalb des SVG-Viewports wird durch das *useGridZoomController* ermöglicht, welches Zoom- und Pan-Funktionen bereitstellt. Darüber hinaus verwaltet das *useElementAttrState* den Zustand der Attribute von Knoten und Kanten, etwa die Auswahl über Checkboxes oder die Eingabe von Werten. Schließlich ist *useElementDialogUtils* eine speziell für Dialogkomponenten entworfene Hilfsklasse, die notwendigen Funktionen zur Steuerung von Dialogen bereitstellt, wie beispielsweise die Änderung von Attributwerten von Elementen und deren Komponenten (Lasten bzw. Generatoren).

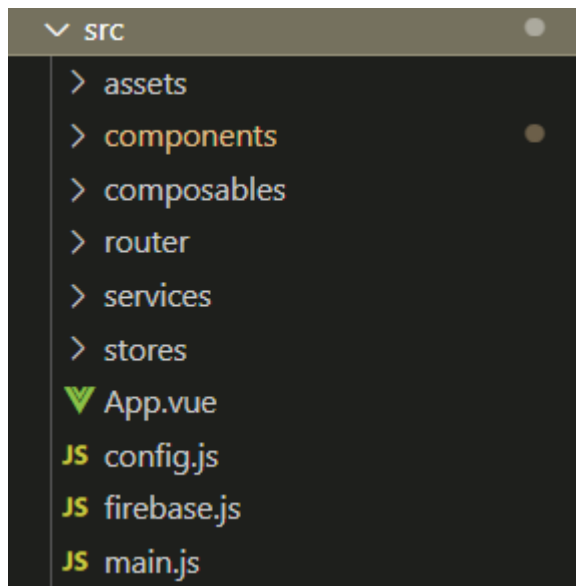


Abb. 3 – Aufbau von GridVisu (GridVisu in VSC.)

Die **Components** repräsentieren größtenteils die einzelnen Seiten sowie spezifische UI-Elemente. Sie definieren die HTML-Struktur und das Styling und greifen dabei auf die zugrunde liegenden Composables und Stores zurück.

Als Einstiegspunkt dient die Hauptkomponente *App*, über die der Vue Router eingebunden wird. Von hier aus werden die jeweiligen Seiten

geladen, und die Komponenten können sich gegenseitig aufrufen. Neben den drei zentralen Hauptkomponenten existieren zudem entkoppelte und wiederverwendbare Teilkomponenten. Dazu zählen insbesondere modulare Dialog- und Panel-Komponenten wie etwa der *ElementDialog* oder der *ApplyDBDialog*.

Der Ordner **Assets** enthält alle statischen Ressourcen wie Icons, Bilder oder Schriftarten, die für die Gestaltung und visuelle Konsistenz der Benutzeroberfläche genutzt werden.

Abbildung 4 zeigt die Architekturübersicht von GridVisu. Das Frontend gliedert sich in Stores, Services, Router, Composables, Components und Assets. Über den REST-Client erfolgt die Kommunikation mit dem Backend, während Netzmodelle zusätzlich in Firebase gespeichert werden können. Anzeige- und Theme-Einstellungen werden ausschließlich im Browser im localStorage gesichert. Der Nutzer interagiert über die Components mit dem System, die wiederum auf Stores, Composables und Router zurückgreifen.

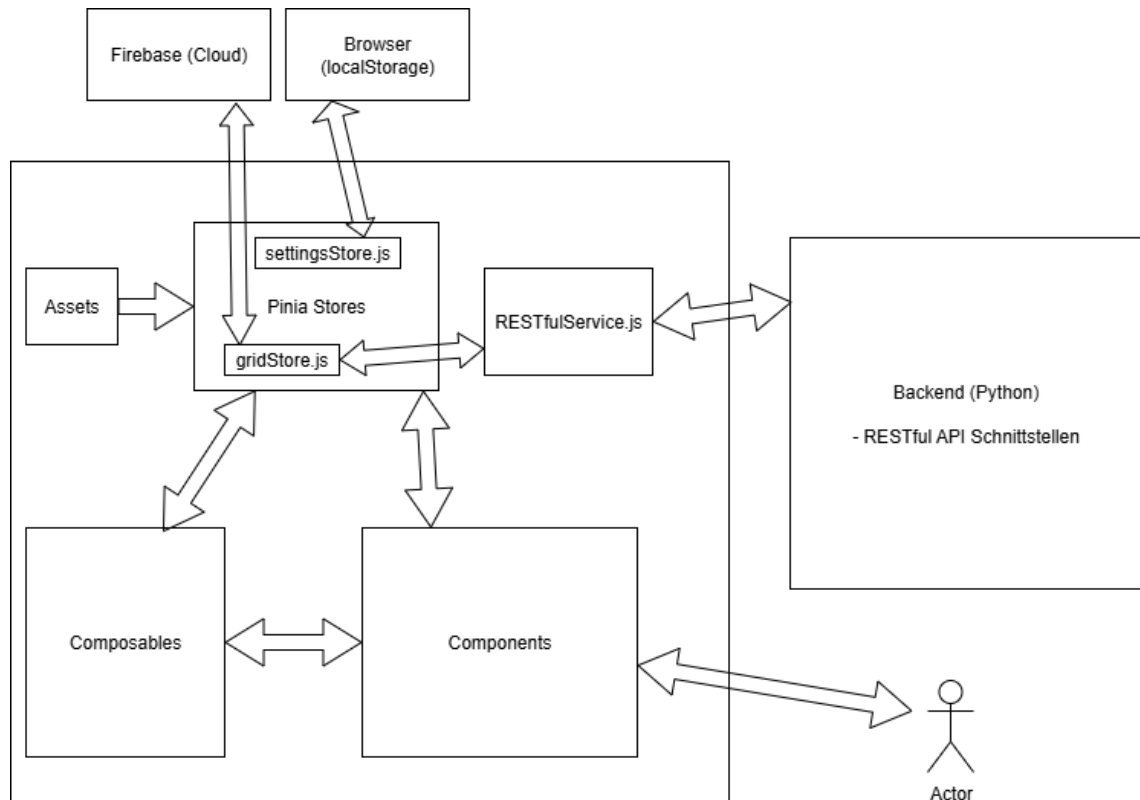


Abb. 4 – Architekturübersicht der entwickelten Anwendung (erstellt mit draw.io)

4.2 Seitenaufbau und Modellierung

In der nachfolgenden [Abbildung 5](#) wird die Navigation zwischen den Hauptseiten und die Verbindung zu den wichtigsten Dialogen ersichtlich. Von der HomePage gelangt der Nutzer nach Auswahl eines Netzes zur SimulationPage. Auf dieser Seite stehen verschiedene Dialoge für Bearbeitung, Analyse und Fehlerbehandlung zur Verfügung. Das ConfirmDialog und ApplyDialog werden genutzt, wenn kritische Aktionen wie das Zurücksetzen eines Netzes oder das Speichern auf dem Server ausgeführt werden sollen. Der ElementDialog öffnet sich beim Anklicken eines Knotens und erlaubt die Einsicht und Bearbeitung aller Attribute. Für zugehörige Komponenten wie Lasten oder Generatoren dient der DialogComponent, der die Detailansicht und Anpassung dieser Elemente ermöglicht. Über den SimPanelDialog können Simulationswerte über die Zeit geladen und in Diagrammen dargestellt werden. Das GlobalMenu im Header der Anwendung ist von überall aus erreichbar, über das globale Designeinstellungen angepasst werden können.

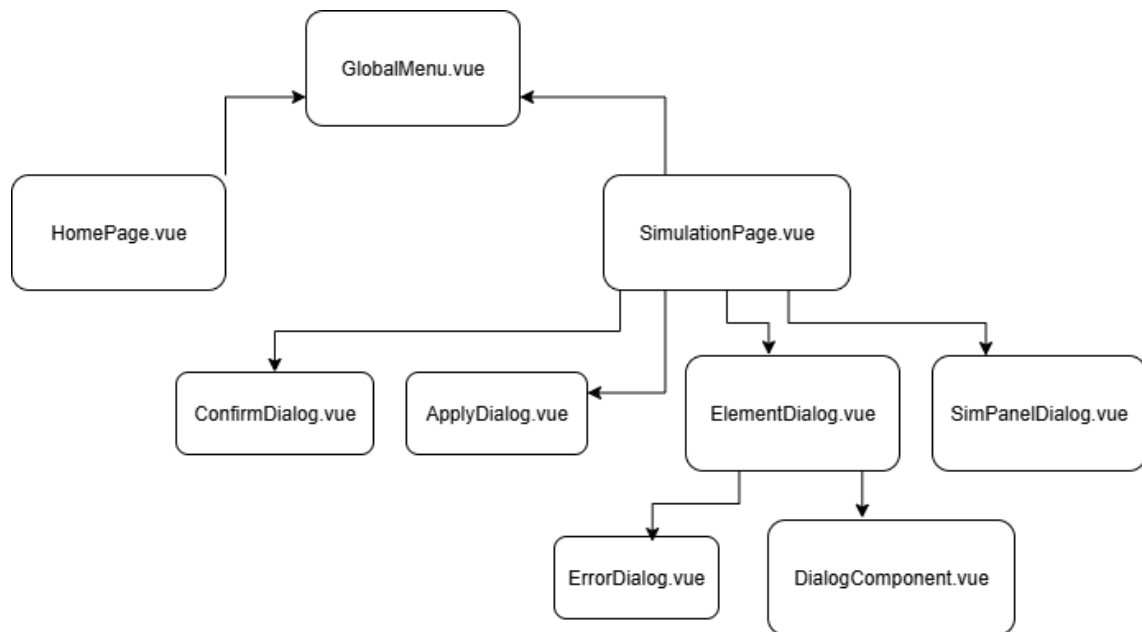


Abb. 5 – Navigationsfluss und Beziehungen aller Komponenten der Anwendung (erstellt mit draw.io)

HomePage

Die *HomePage* bildet die Startseite der Anwendung und zeigt eine Übersicht aller verfügbaren Netze. Jedes Netz wird in einem kompakten Preview-Format als kleines SVG dargestellt, sodass die grobe Struktur und die Verbindungen der Knoten auf einen Blick erkennbar sind. Für diese Previews wird ein skalierbares Grid genutzt, das über CSS-Eigenschaften horizontal flexibel auf unterschiedliche Bildschirmbreiten reagiert. Dadurch werden die Darstellungen weder zu groß noch zu klein, sondern passen sich stets den Dimensionen des Endgeräts an. Falls ein Netz noch keine Koordinaten für Knoten enthält, werden diese automatisch generiert und direkt für die Vorschau verwendet, um eine konsistente Darstellung sicherzustellen. Ergänzend ermöglicht eine Suchfunktion das Filtern nach Namen oder IDs, und über den Header kann global zwischen hellem und dunklem Theme (Tag- bzw. Nachtmodus) gewechselt werden. Von hier aus lässt sich zudem auf die globalen Einstellungen zugreifen. Die *HomePage* stellt somit den Ausgangspunkt dar, von dem aus ein ausgewähltes Netz zur weiteren Analyse und Bearbeitung in die *SimulationPage* überführt wird.

SimulationPage - Seitenaufbau

Die SimulationPage stellt den zentralen Arbeitsbereich der Anwendung dar. Im SVG-Bereich wird die vollständige Grid-Topologie angezeigt, dazu folgt später mehr. In der oberen Leiste sind grundlegende Metainformationen wie die Grid-ID, der Name sowie der aktuelle Server-Status (z. B. *synced*, *unsaved* oder *saved*) jederzeit sichtbar. Änderungen am Grid wirken sich unmittelbar auf diesen Status aus: Sobald ein Element editiert wird, erfolgt eine farbliche Markierung und der Status wechselt automatisch auf *unsaved* (s. [Abbildung 6](#)). Dadurch ist jederzeit klar erkennbar, ob noch nicht gespeicherte Änderungen vorliegen. Um unbeabsichtigten Datenverlust zu vermeiden, erscheint beim Verlassen der Seite ein Dialogfenster, das die Aktion abfängt und so ein versehentliches Löschen oder Überschreiben verhindert.



Abb. 6 – Typischer Speicherungsablauf (Q.: GridVisu)

Am unteren Rand befindet sich eine kompakte Werkzeugleiste, die alle wichtigen Steuerungsoptionen bündelt. Hier lassen sich Zoom- und Pan-Modus aktivieren oder deaktivieren, ebenso Edit-Mode und Show-Labels. Ergänzend stehen Schaltflächen für Save und Apply zur Verfügung. Über ein Menü mit drei Punkten können zudem weniger häufig verwendete Funktionen wie Import/Export, Reset Grid oder Delete Grid aufgerufen werden. Die Gestaltung ist bewusst kompakt und übersichtlich gehalten, wobei Wert auf eine klare und moderne Benutzeroberfläche gelegt wurde. Sichtbar wird dies unter anderem durch den Einsatz großflächiger, farblich markierter Checkboxes, die nicht als kleine Symbole, sondern als vollständige farbige Kästen dargestellt sind. Dadurch wird die Interaktion hervorgehoben und die Bedienung insgesamt intuitiver.

SimulationPage - Modellierung

Die Grundlage für die Visualisierung der Netze bildet eine JSON-Struktur, in der Knoten und Kanten definiert sind. Jeder Knoten besitzt eine eindeutige ID sowie Attribute wie Spannungsebene, Knotentyp oder zugeordnete Last- und Erzeugungselemente. Die Kanten verweisen auf die IDs der Knoten, die sie verbinden, und beinhalten zusätzlich elektrische Parameter wie Widerstand, Nennfrequenz oder Leitungslänge.

Für die grafische Umsetzung wird D3 verwendet. Die Bibliothek ermöglicht es, die JSON-Daten direkt als Objekte zu übergeben und daraus die Elemente im SVG-Bereich zu generieren. Knoten werden als Kreise, Kanten als Linien dargestellt; Beziehungen zwischen Knoten und Kanten können dabei konsistent abgebildet werden, da D3 das zugrunde liegende Datenmodell mit den grafischen Elementen verknüpft. Anpassungen wie Farbgebung, Hervorhebung des Vaterknotens oder Interaktionen (z. B. Klick, Hover, Drag-and-Drop) lassen sich unmittelbar in den Rendering-Prozess integrieren. Damit entsteht aus einer abstrakten Datenstruktur eine intuitive, interaktive Visualisierung des Netzes.

Abbildung 7 zeigt exemplarisch den Zusammenhang zwischen JSON-Struktur und grafischer Darstellung am Beispiel von Bus 13, Bus 14 und der Verbindung Line 13–14.

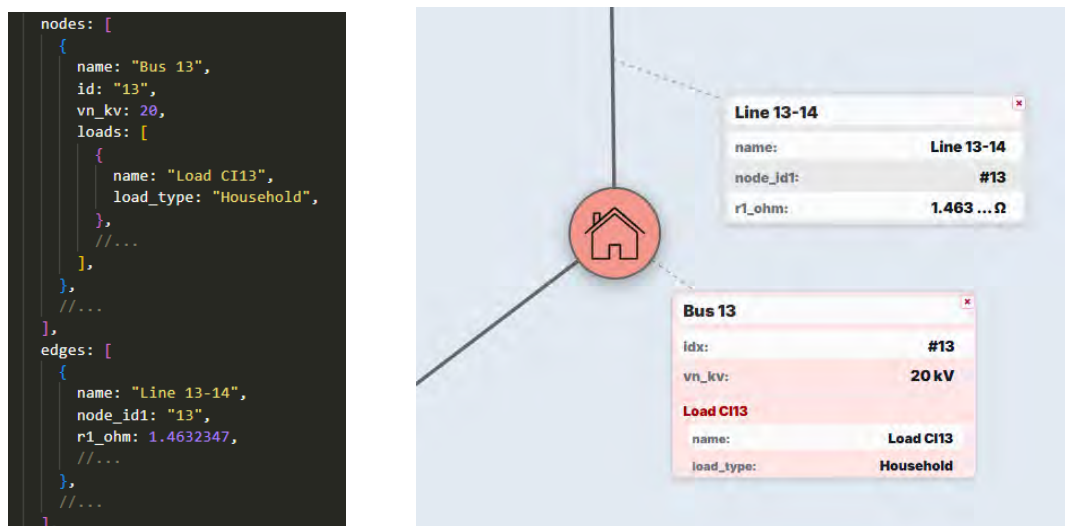


Abb. 7: Abbildung der JSON-Modellstruktur in eine interaktive Visualisierung mit D3 (Screenshots aus GridVisu und VSC.)

Die Netzstruktur wird detailliert dargestellt und ist farblich so aufbereitet, dass Lasten und Erzeuger sofort erkennbar sind. Es kann durchaus vorkommen, dass

sich an einem Knoten gleich mehrere Lasten bzw. Generatoren befinden. Hier gibt es eine unterstützende, simple Darstellungsweise: Die Knotenkreise werden leicht versetzt übereinander gerendert, sodass deren Lage auf einen Blick

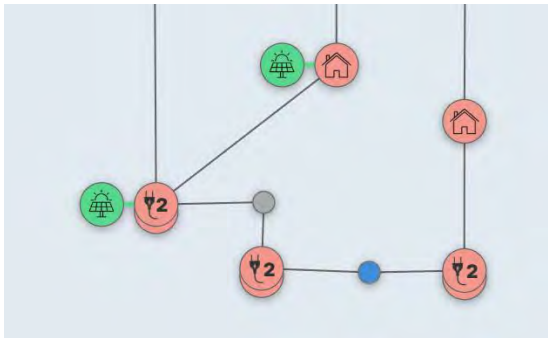


Abb. 8 – Einige Knotentypen (Q.: GridVisu)

erkennbar ist (s. dazu [Abbildung 8](#)). Außerdem werden Generatoren dabei nicht direkt auf den Knotenpunkten platziert, sondern links daneben dargestellt und über eine Verbindungslinie eindeutig mit dem jeweiligen Knoten verknüpft.

An den Knoten werden zudem spezifische Icons angezeigt, die den Typ der angeschlossenen Last- bzw. Generator-Elemente kennzeichnen, beispielsweise Haustypen für Lasten oder PV-Symbole für Generatoren. Auf den Kanten werden zusätzliche Symbole für Betriebsmittel wie Schalter oder Transformatoren eingeblendet. Schalter können dabei mit einem Klick direkt ein- oder ausgeschaltet werden.

Mit einem Klick auf einen Knoten wird (mit [ElementDialog](#)) eine vollständige Attributliste angezeigt, die übersichtlich und benutzerfreundlich gestaltet ist (s. Abb. 9). Durch Auswahl einzelner Attribute erscheinen entsprechende Labels direkt im Netz, welche sich zudem frei verschieben lassen. Einige Labels sind in [Abbildung 6](#) zu sehen. Die Anzahl der darzustellenden Attribute ist beliebig; sie werden sortiert und strukturiert in den Labels angeordnet. Dieses Prinzip gilt gleichermaßen für Komponenten wie Lasten oder Generatoren, deren Attribute nicht nur eingesehen, sondern auch direkt in Eingabefeldern angepasst werden können.

Abb. 9 - Element Dialog eines Generatorknotens (Q.: GridVisu)

Darüber hinaus lassen sich im Netz selbst die Elemente interaktiv per Drag-and-Drop an neue Positionen verschieben, was die Bedienung besonders flexibel und vorteilhaft macht. Schalter, welche ebenfalls automatisch an den jeweiligen Kanten gerendert werden, lassen sich mit einem Klick an und ausschalten.

SimPanelDialog

Das SimPanel erweitert die Komponente um die Möglichkeit, zeitabhängige Messdaten direkt im System zu analysieren. Nutzerinnen und Nutzer können hierzu einen Zeitraum angeben und anschließend über einen Klick die zugehörigen Messwerte laden. Wurde zuvor eine Kante ausgewählt, werden die Strommessungen im gewählten Intervall abgerufen; bei Knoten die Spannungswerte.

Die Daten werden dann in einem Diagramm über die Zeitachse dargestellt (s. [Abb. 10](#)). Über Mausinteraktionen oder Pfeiltasten lassen sich spezifische Zeitabschnitte markieren, wobei die zugehörigen Werte direkt angezeigt werden. Parallel zur Diagramm-Darstellung



Abb. 10 – Messwert-Diagramm eines Knotenelements (Q.: GridVisu)

werden kritische

Zustände auch im Grid selbst unmittelbar sichtbar: So wird beispielsweise eine Überlastung eines Knotens durch einen (standardgemäß) rot leuchtenden Glüheffekt im Hintergrund hervorgehoben. Auf diese Weise verknüpft das SimPanel numerische Messwerte mit einer visuellen Rückmeldung im Grid und unterstützt damit sowohl die Analyse als auch die intuitive Erkennung kritischer Situationen.

GlobalMenu

Das GlobalMenu bündelt den Zugriff auf sämtliche globalen Designeinstellungen und stellt diese in einer übersichtlichen Benutzeroberfläche bereit. Über Selektoren, Colorpicker und Slider können zentrale Parameter wie Farben, Größen oder Kantendicken direkt angepasst werden. Dabei werden alle Änderungen live ins Grid übertragen, sodass das Ergebnis unmittelbar sichtbar ist und ohne zusätzlichen Bestätigungsschritt übernommen wird, was in der folgenden *Abbildung 11* zu sehen ist.

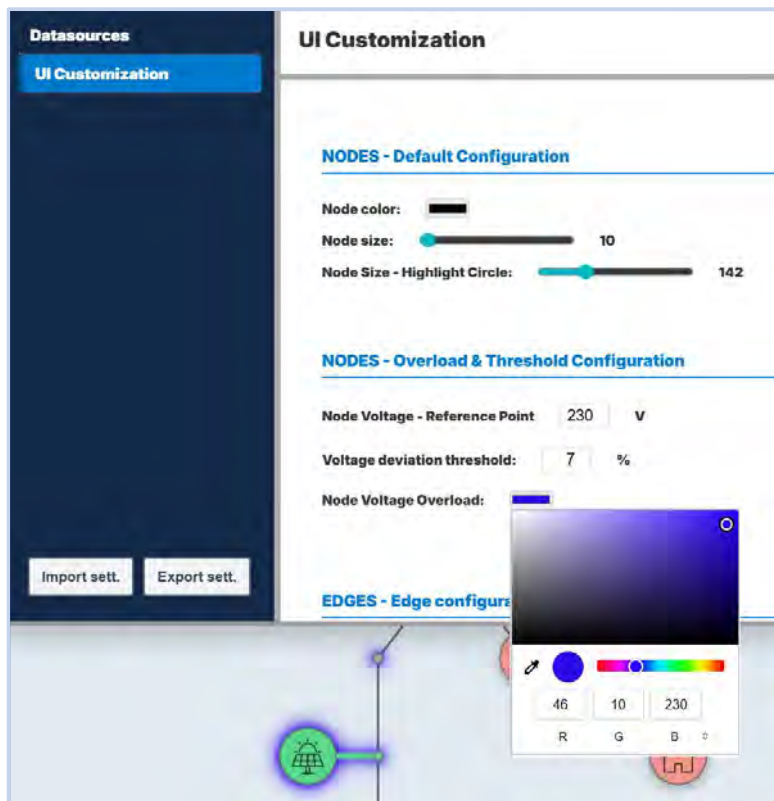


Abb. 11 – Menüseite und Live-Änderung der „Node Voltage Overload“ Farbe (Q.: GridVisu)

Neben diesen direkten Anpassungen bietet das GlobalMenu auch die Möglichkeit, Einstellungsprofile zu exportieren oder zu importieren. Dadurch lassen sich individuell konfigurierte Layouts sichern und bei Bedarf wieder abrufen, ohne die Parameter erneut manuell eingeben zu müssen. Ergänzend steht ein Reset-Button zur Verfügung, mit dem jederzeit auf die

Standardwerte zurückgekehrt werden kann. Da die Konfigurationen im *settingsStore* hinterlegt sind, gelten sie netzübergreifend und bleiben auch bei einem Wechsel des Netzes erhalten. Auf diese Weise unterstützt das GlobalMenu nicht nur die konsistente Gestaltung der Visualisierung, sondern erleichtert zugleich die Bedienung durch eine intuitive und zentrale Steuerung.

4.3 Workflows

Der typische Workflow mit der Anwendung lässt sich anhand eines Beispiels verdeutlichen. Ausgangspunkt ist die *HomePage*, auf der zunächst ein gewünschtes Grid ausgewählt wird (s. *Abb. 12*).

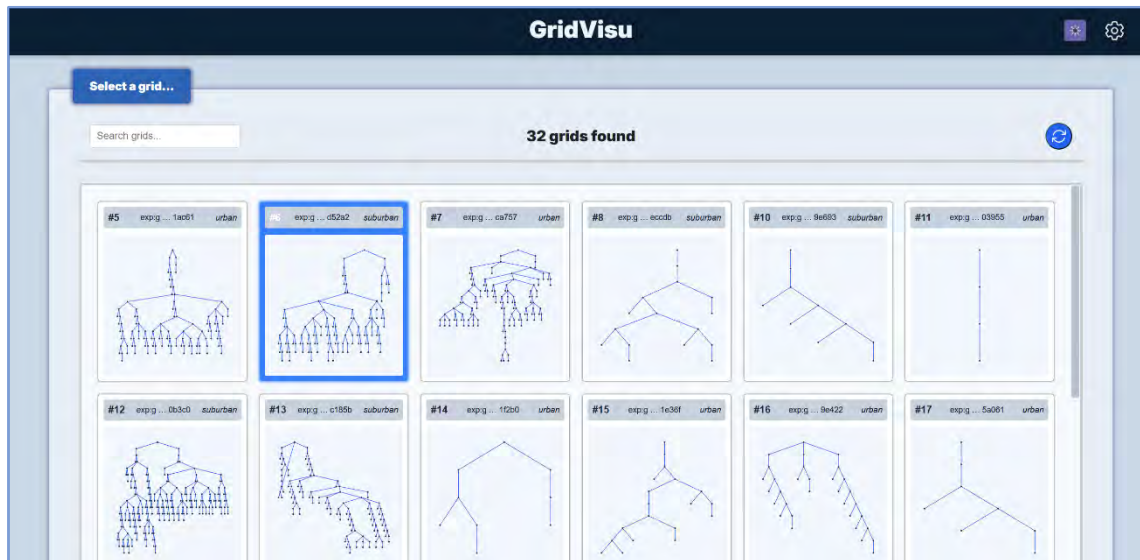


Abb. 12 – Startseite der Anwendung und Auswahl eines Netzes (Q.: GridVisu)

Nach dem Wechsel in die *SimulationPage* wird die Netzstruktur im SVG-Bereich dargestellt. Hier können beliebige Labels eingeblendet werden. Im nächsten Schritt lassen sich globale Einstellungen verändern, zum Beispiel die Farbgebung der Knoten über das GlobalMenu, was in *Abbildung 13* bereits getan wurde.



Abb. 13 – SimulationPage: Eingeblendete Labels und Farbanpassung des Netzes (Q.: GridVisu)

Anschließend kann die Positionierung der Knoten per Drag-and-Drop geändert werden. Sind die Anpassungen abgeschlossen, wird das Grid über die Funktion `ApplyToServer` dauerhaft gespeichert. Dadurch ist sichergestellt, dass der aktuelle Stand synchron mit dem Server vorliegt und die lokale Firebase-Instanz zurückgesetzt wird. Die *ApplyDialog* Komponente ist besonders nutzerfreundlich gestaltet, da sie vor der Bestätigung alle Unterschiede zwischen der aktuellen lokalen Version und dem auf dem Server gespeicherten Stand übersichtlich darstellt. Auf diese Weise sind die bevorstehenden Änderungen für die Nutzerinnen und Nutzer klar nachvollziehbar, bevor sie übernommen werden. Erst nach einer expliziten Bestätigung wird der Vorgang ausgeführt, wodurch unbeabsichtigte Änderungen oder Datenverluste zuverlässig vermieden werden (s. Abb. 14).

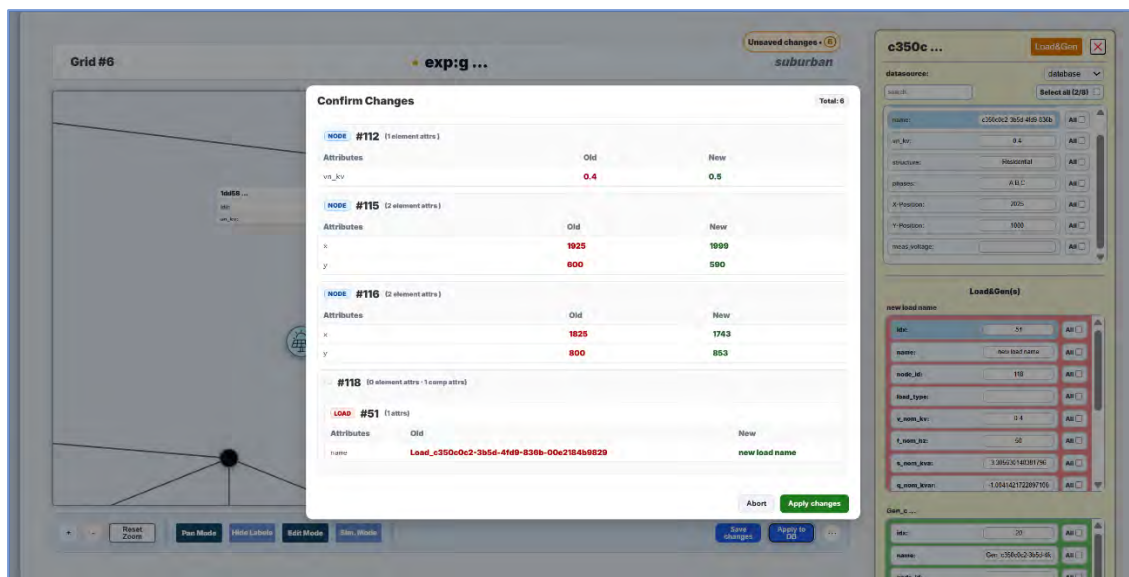


Abb. 14 - *ApplyDialog*: Übersicht aller Änderungen am Grid vor dem persistenten Speichern (Q.: GridVisu)

Abschließend können im SimPanel zeitabhängige Messwerte geladen und analysiert werden. Hierbei zeigt das Diagramm die Werte im Verlauf, während kritische Zustände zusätzlich direkt im Grid visuell hervorgehoben werden, was in *Abbildung 15* zu erkennen ist.



Abb. 15 - Visuell hervorgehobene Knoten mit kritischem Zustand (Q.: GridVisu)

Kapitel 5 – Bewertung der Anwendung

Die entwickelte Anwendung wird im Folgenden hinsichtlich ihrer Funktionalität, Benutzerfreundlichkeit sowie im Vergleich zu bestehenden Softwarelösungen bewertet. Dabei werden sowohl allgemeine Kriterien als auch die „Goldenen Regeln“ der Interaktionsgestaltung nach Shneiderman berücksichtigt.

5.1 Vergleich zu bestehenden Lösungen

Am Markt verfügbare Werkzeuge zur Simulation und Analyse elektrischer Verteilnetze bieten in der Regel eine sehr große Funktionsvielfalt, sind jedoch häufig auf komplexe Desktop-Umgebungen ausgelegt und erfordern umfangreiche Fachkenntnisse. Die Einarbeitung ist für unerfahrene Anwenderinnen und Anwender entsprechend anspruchsvoll.

Die hier vorgestellte Anwendung verfolgt hingegen bewusst einen reduzierten, webbasierten Ansatz. Sie stellt die grundlegende Basis bereit, die für die

Modellierung, Visualisierung und grundlegende Analyse von Netzen notwendig sind. Hierbei wird eine ausreichende Tiefe in der Funktionalität gewährleistet, sodass auch fortgeschrittene Szenarien (z. B. zeitabhängige Analyse von Messdaten) unterstützt werden. Durch den modularen Entwicklungsstil ist sie entsprechend einfach zu erweitern. Sie ist insbesondere für den Einsatz in Lehre und Ausbildung geeignet, wo eine intuitive Bedienung und eine niedrige Einstiegshürde im Vordergrund stehen.

Ein zusätzlicher Vorteil liegt in der Schnittstellengestaltung der Software, die über eine einheitliche RESTful API realisiert ist. Dieser Ansatz ist in modernen Softwareumgebungen weit verbreitet und wird durch eine große Zahl an Tools, Bibliotheken und Frameworks unterstützt. Dadurch ist die Lösung sowohl technologisch anschlussfähig als auch langfristig erweiterbar und kann problemlos in bestehende Systeme integriert werden.

5.2 Benutzerfreundlichkeit

Die Benutzeroberfläche ist modular aufgebaut und folgt einer klaren Navigationsstruktur. Über die HomePage erfolgt zunächst die Auswahl und Übersicht der Netze. Anschließend bildet die SimulationPage das zentrale Arbeitsumfeld, in dem die eigentliche Bearbeitung und Analyse stattfindet. Ergänzend stellt das GlobalMenu übergreifende Einstellungen bereit, die gridunabhängig wirken und konsistent übernommen werden.

Durch diesen Aufbau ist der Workflow klar gegliedert und reduziert sich auf wenige Kernschritte: Netz auswählen, visualisieren, bearbeiten, speichern und analysieren. Dabei treten mehrere Aspekte besonders hervor. Änderungen an Designeinstellungen oder am Datenmodell werden unmittelbar im Grid sichtbar und bieten so eine direkte Rückmeldung. Der aktuelle Serverstatus eines Grids (wie *synced* oder *saved*) wird permanent angezeigt und zusätzlich farblich hervorgehoben, sodass jederzeit Transparenz über den Bearbeitungsstand besteht. Zudem ist die Bedienung durch Bestätigungsdialoge für kritische Aktionen wie *Apply to Server* oder *Reset Grid* abgesichert. Selbst das Neuladen der Seite wird abgefangen, um unbeabsichtigte Datenverluste zu vermeiden.

Darüber hinaus unterstützt die Oberfläche eine visuelle Verstärkung von Zuständen, beispielsweise durch einen roten Glow-Effekt bei überlasteten Knoten. Abgerundet wird das Bedienkonzept durch eine konsistente Gestaltung, die auf wiederverwendbaren UI-Elementen basiert. Große, farbige UI-Elemente in der Optionsleiste ersetzen herkömmliche kleinere Felder und erhöhen die Übersichtlichkeit wie auch die Nutzerfreundlichkeit.

5.3 Bewertung anhand der „Goldenen Regeln“

Die Anwendung lässt sich auch im Hinblick auf die von Shneiderman formulierten „Goldenen Regeln der Interaktionsgestaltung“ bewerten. Ein wesentliches Prinzip ist die Konsistenz, die hier durch die Navigation über den Vue Router sowie die einheitliche Struktur von Dialogen, Buttons und Panels gewährleistet wird. Gleiche Aktionen führen stets zu gleichen Ergebnissen, was die Bedienbarkeit erleichtert. Darüber hinaus ist die Rückmeldung klar erkennbar: Jede Aktion – sei es das Speichern eines Grids oder die Anpassung einer Einstellung – wird unmittelbar durch visuelle Hinweise verdeutlicht.

Die Anwendung trägt auch zur Fehlervermeidung bei. Bestätigungsdialoge, permanente Zustandsanzeigen und das Abfangen kritischer Browser-Aktionen wie dem unbeabsichtigten Neuladen stellen sicher, dass Fehlbedienungen rechtzeitig abgefangen werden. Gleichzeitig behalten die Nutzerinnen und Nutzer durch Benutzerkontrolle und Flexibilität jederzeit die Übersicht. Darüber hinaus werden Eingaben gezielt geprüft: In Zahlenfeldern ist es beispielsweise nicht möglich, Buchstaben einzugeben. Wird eine unzulässige Eingabe versucht oder die Eingabe mit der Enter-Taste abgebrochen, bleibt automatisch der zuletzt gültige Wert erhalten. Auf diese Weise werden fehlerhafte Eingaben gar nicht erst zugelassen. Komplexere Eingaben wie die Angabe von Phasen (A, B, C) werden korrekt geparkt und formatiert, sodass auch bei mehrdimensionalen Daten eine zuverlässige Verarbeitung gewährleistet ist.

5.4 Zusammenfassung der Bewertung

Insgesamt zeigt sich, dass die entwickelte Anwendung die Anforderungen an eine benutzerfreundliche und zugängliche Plattform für die Visualisierung und Analyse elektrischer Verteilnetze erfüllt. Im Vergleich zu bestehenden Lösungen besticht sie durch ihre Webbasiertheit, die geringe Einstiegshürde sowie die klare visuelle Rückmeldung. Damit bietet sie insbesondere für Ausbildungsszenarien und den Einsatz im universitären Umfeld Vorteile.

Gleichzeitig lässt sich festhalten, dass der Funktionsumfang bewusst begrenzt ist und nicht mit industriellen Standardlösungen konkurriert. Vielmehr stellt die Anwendung ein sinnvoll reduziertes Werkzeug dar, das zentrale Konzepte der Netzmodellierung und -analyse verständlich und interaktiv vermittelt. Dabei ist die technische Basis bereits solide gesetzt: Mit der Entscheidung für die moderne Webtechnologie Vue und einer klar strukturierten RESTful API wurde eine Lösung gewählt, die technologisch anschlussfähig ist und sich nahtlos in andere Softwareprojekte einfügt. Durch den typisch modularen Aufbau mit Stores, Composables und wiederverwendbaren Komponenten ist die Anwendung zudem flexibel erweiterbar. Neue Dialoge, zusätzliche Messgrößen oder alternative Visualisierungen können ohne tiefgreifende Eingriffe ergänzt werden, sodass die Plattform langfristig anpassbar bleibt.

Kapitel 6 – Fazit und Ausblick

Mit dieser Arbeit wurde eine webbasierte Anwendung entwickelt, die eine anschauliche Modellierung, Visualisierung und Analyse elektrischer Verteilnetze ermöglicht. Die Software setzt auf eine Architektur mit modernen Webtechnologien und bietet eine benutzerfreundliche Oberfläche, die bewusst auf die wesentlichen Funktionen reduziert ist.

Der besondere Mehrwert liegt darin, dass komplexe Netzstrukturen verständlich und interaktiv dargestellt werden können. Durch einfache Workflows, unmittelbare Rückmeldungen und klare visuelle Hervorhebungen eignet sich die Anwendung insbesondere für den Einsatz in der Lehre. Sie unterstützt Lernende dabei, zentrale Konzepte der Netzmodellierung praktisch nachzuvollziehen, ohne durch überflüssige Komplexität überfordert zu werden.

Aufgrund ihres modularen Aufbaus ist die Anwendung zugleich zukunftsfähig. Erweiterungen wie zusätzliche Messgrößen, erweiterte Analysefunktionen oder alternative Visualisierungen lassen sich ohne grundlegende Anpassungen realisieren. Damit stellt die Software eine solide Basis dar, die im Bildungsumfeld eingesetzt und bei Bedarf flexibel weiterentwickelt werden kann

Literaturverzeichnis

50komma2 (2024). *pandapower erreicht Rekorddownloads*. Abgerufen am [27.08.25], von <https://www.50komma2.de/digitalisierung/pandapower-rekord/>

DIGSILENT GmbH (2024). *PowerFactory – Netzberechnung und Simulation*. Abgerufen am [24.08.25], von <https://www.digsilent.de/de/powerfactory.html>

DIGSILENT GmbH (o.J.). *Network Model Diagrams in PowerFactory*. Abgerufen am [21.08.25], von <https://www.digsilent.de/de/fachbeitraege-reader-pf-de/network-model-diagrams-in-powerfactory-2.html>

EPRI – Electric Power Research Institute (2024). *Introduction to OpenDSS*. Abgerufen am [21.08.25], von <https://opendss.epri.com/IntroductiontoOpenDSS.html>

Fraunhofer IEE (2024). *pandapower Open-Source-Tool erreicht 500k Downloads*. Abgerufen am [31.08.25], von <https://www.iee.fraunhofer.de/de/presse-infothek/Presse-Medien/2024/pandapower-open-source-tool-erreicht-500k-downloads.html>

KIT – Institut für Elektroenergiesysteme und Hochspannungstechnik (o.J.). *Institut für Elektroenergiesysteme und Hochspannungstechnik*. Abgerufen am [31.08.25], von <https://www.ieh.kit.edu/2830.php>

LF Energy (o.J.). *Power Grid Model*. Abgerufen am [31.08.25], von <https://lfenergy.org/projects/power-grid-model/>

Shneiderman, B. (1992). *Eight Golden Rules of Interface Design*. Abgerufen am [22.08.25], von <https://www.cs.umd.edu/~ben/goldenrules.html>

Springer (2021). *Artikel: [Elektrische Verteilnetze resilient ausbauen – Herausforderungen und Handlungsoptionen, DOI: 10.1007/s12398-021-00316-x]. Sustainability Management Forum (NachhaltigkeitsManagementForum)*. Abgerufen am [21.08.25], von <https://link.springer.com/article/10.1007/s12398-021-00316-x>

Verteilnetzstudie NRW (2021). *Landesregierung Nordrhein-Westfalen: Verteilnetzstudie NRW 2021*. Abgerufen am [29.08.25], von <https://www.wirtschaft.nrw/pressemitteilung/verteilnetzstudie-nrw>

Wikipedia (o.J.). *Lastflussberechnung*. Abgerufen am [29.09.25], von <https://de.wikipedia.org/wiki/Lastflussberechnung>

Wikipedia (o.J.). *ISO 9241*. Abgerufen am [29.08.25], von https://de.wikipedia.org/wiki/ISO_9241

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift