EGen Hybrid App

Höhere Abteilung für Informatik

01/10/2024 – 01/04/2025

Projektmitglieder: Lukas Parzer

Betreuer:in: Prof. Michael Holzmann

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Bei der Erstellung der Arbeit wurden keine generativen KI-Tools verwendet.

Danksagung

Ich möchte ich bei allen bedanken, die mich während der Erstellung dieser Arbeit tatkräftig unterstützt haben. Insbesondere bei meinem Diplomarbeitsbetreuer, Herrn Professor Michael Holzmann, welcher eine große Hilfe für mich war und mir mit wertvollen Ratschlägen zu einem besseren Ergebnis verholfen hat. Außerdm möchte ich mich bei der Energiegenossenschaft Perg bedanken, vor allem dem Obmann Herrn Johannes Oppitz, welcher mir bei Problemen stets mit einer helfende Hand zur Seite stand.

Abstract

The aim of this work is to display energy data provided by the energy cooperative Perg in an app. This app should be developed using Ionic (see section: 2.2.2) and use graphs to provide insights into historical data. The data is provided by the energy cooperative through a Firebase database (see section: 2.3). For visualization, the ngx-charts package (see section: 2.5.1) was used. The result is an app where data from a specific meter point is visualized and filtered by date.

Zusammenfassung

Aufgabe ist es, die Energiedaten, welche von der Energiegenossenschaft Perg zur Verfügung gestellt werden, in einer App anzuzeigen. Diese App soll mit Ionic (siehe Abschnitt: 2.2.2) entwickelt werden und mithilfe von Graphen einen Blick auf historische Daten ermöglichen. Die Daten werden von der Energiegenossenschaft über eine firebase Datenbank (siehe Abschnitt: 2.3) zur Verfügung gestellt. Zur Visualisierung wurde das ngx-charts Packet (siehe Abschnitt: 2.5.1) verwendet. Das Ergebnis ist eine App bei der die Daten eines Zählpunktes nach Datum gefiltert Visualisiert werden.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage

Um die Anzeige der Energiedaten zu verbessern, soll eine Applikation für die EEG-Perg entwickelt werden, welche durch einen Login die Visualisierung der eigenen historischen Daten in Form von Diagrammen ermöglicht. Zusätzlich sollen Administratoren die Möglichkeit haben, alle Daten anzusehen. Diese Applikation soll sowohl auf Android als auch auf iOS verfügbar sein.

1.2 Auftraggeber

Der Auftraggeber dieser Arbeit ist die Energiegenossenschaft Perg, welche Ende 2022 in Kooperation mit der Raiffeisenbank Perg gegründet wurde. Das Gebiet der EEG-Perg umfasst Perg, Allerheiligen, Windhaag und Teilbereiche von Münzbach und Schwertberg. Mein Ansprechpartner für dieses Projekt war Johannes Oppitz, der Obmann der Energiegenossenschaft.

1.3 Projektinhalt

Inhalt des Projektes ist es, eine Applikation zur Anzeige von Energiedaten mit Ionic (siehe Abschnitt 2.2.2) zu erstellen. Die Funktionen dieser App umfassen die Authentifizierung und die Anzeige der Daten in Form von Diagrammen. Diese grafischen Darstellungen sollen auf einen Zeitraum beschränkbar sein können, um die Übersichtlichkeit der Abbildung zu gewährleisten. Die Administratoren sollen zusätzlich alle Daten sehen können. Die Daten, welche für diese Applikation benötigt werden, werden durch eine weitere Gruppe bereitgestellt und sind von Firebase (siehe Abschnitt 2.3) zu laden.

1.4 Ziele und Nichtziele

Ziel ist es Energiedaten von einer Firebase-Datenbank zu laden und mithilfe von Graphen in einer für den Benutzer bequemen Art, in einer mobilen App anzuzeigen. Ein weiteres Ziel ist es, den Login mithilfe von E-Mail und Passwort zu ermöglichen. Kein Ziel ist es, Energiedaten in

der Datenbank zu speichern oder Daten zu verändern. Auch die Korrektheit der Daten in der Datenbank ist nicht Teil dieser Arbeit.

2 Grundlagen und Methoden

2.1 Wie funktioniert Authentifizierung im Web

Die Authentifizierung eines Benutzers stellt fest, ob ein bestimmter Nutzer existiert und wer er ist. Dazu muss natürlich jeder Nutzer ein Alleinstellungsmerkmal haben, welches oft die E-Mail Adresse ist. Auch andere Authentifizierungsmöglichkeiten werden häufig verwendet. Beispiele hierfür wären, die Identifikation mit Hilfe von Google Authentifikation, Apple Authentifikation oder Microsoft Authentifikation, die Eingabe der Telefonnummer, das Scannen des Fingerabdrucks oder die Verwendung der Gesichtserkennungsfunktion. Damit sich jedoch niemand als jemand anderes ausgeben kann, reicht die E-Mail Adresse nicht aus und sie muss mit einem Passwort bestätigt werden. Wenn dieses Passwort nun unverschlüsselt ist, kann es leicht gestohlen werden, deshalb sollte der Server Passwörter nie unverschlüsselt speichern und die Übertragung sollte mittels dem Sicherheitsprotokolls TLS, welches in HTTPS Seiten enthalten ist, übertragen werden. Die Passwörter sollten gehasht gespeichert werden und am besten sollte hierbei ein Salt verwendet werden, damit Rainbow Table Angriffe verhindert werden können. Ein Salt ist ein zufällig generierter Wert, welcher einem Passwort angefügt wird, um es zusätzlich zu sichern. Zusätzlich zu Salt gibt es auch noch den Pfeffer, welcher wie das Salz funktioniert, allerdings extern gespeichert wird. Bei einem Rainbow Table Angriff werden häufig verwendete Passwörter benutzt, um sich einen Zugang zu den Daten des Opfers zu verschaffen. Bei der Anmeldung wird das Passwort des Nutzers mit dem selben Salzwert gehasht und mit dem gehasht, gespeicherten Passwort verglichen. Nach der erfolgreichen Authentifizierung, wird oft ein Token erstellt, um den Nutzer identifizieren zu können.

2.2 Verwendete Technologien

Zur Entwicklung der Hybrid App wurden viele verschiedene nützliche, moderne, öffentlich zugängliche und kostenlose Technologien verwendet, wie zum Beispiel Ionic und Angular.

2.2.1 Angular

Angular ist ein Open-Source-Framework welches die Verwendung der Scriptsprache JavaScript erleichtert und von Google verwaltet wird. Es dient zur Entwicklung von Singlepage-Webapplikationen und dies wird mit Hilfe von HTML, TypeScript und CSS ermöglicht. Angular ist komponentenbasiert, was es ermöglicht, Codeteile wiederzuverwenden. Auch hilft dies, wegen der klaren Strukturierung. Zusätzlich erleichtert Angular durch die Verwendung von Databindings, die Veränderung von Variablen mithilfe von HTML Elementen und ermöglicht automatische Updates im HTML bei Änderung von Variablen. Zusätzlich verwendet Angular, NodeJS, wodurch, mit Hilfe von einem einzigem Befehl, npm Pakete installiert werden können. Quellen: (Angular[1],Simplilearn[2])

2.2.2 Ionic

Ionic ist ein Framework welches zur Entwicklung von Hybrid-Apps verwendet wird. Es kann auf der Basis von Vue, React oder wie im Fall dieser Arbeit, auf Angular aufbauen. Ionic erlaubt die Entwicklung von sowohl Android, als auch iOS Apps, mit einer einzigen Codebasis. Die App kann außerdem auf einem Webserver gehostet werden, wodurch auch der Zugriff über Linux, Windows oder Mac ermöglicht wird. Zusätzlich hat Ionic eingebaute Komponenten, wie Inputfelder, Knöpfe oder Schieberegler. Ionic ermöglicht auch den Zugriff auf viele native Funktionen, wie die Kamera oder Gyrosensoren, allerdings müssen für manche der zuvor genannten Funktionen Plugins installiert werden. Quellen: (Ionic[3])

2.3 Firebase

Firebase bietet verschiedene Services an. Für diese Arbeit wurde allerdings hauptsächlich die Datenbank und der Authentication Service verwendet. Die Firestore Datenbank ist eine dokumentenbasierte Datenbank, welche die Daten im JSON Format speichert. Der Authentication Service erlaubt es Benutzern, sich über verschiedene Methoden, wie mit dem Google Account oder mit E-Mail und Passwort, zu registrieren und anzumelden. Hierbei wird die Logik von Firebase übernommen.

2.4 Verwendete Entwicklungsumgebungen

2.4.1 Webstorm

Webstorm ist eine Entwicklungsumgebung, welche von Jetbrains entwickelt wurde. Die Entwicklungsumgebung wurde für die Programmierung mit Typescript und Javascript erschaffen. Webstorm hat viele nützliche Funktionen, wie die Hilfe beim Lösen von Git-Merge-Konflikten oder Vorschläge, welche Codeblöcke als nächstes kommen könnten. Außerdem gibt es eine integrierte künstliche Intelligenz, welche bei der Programmierung unterstützt. Quellen: (Ionic[4])

2.4.2 Visual Studio Code

Visual Studio Code ist ein Codeeditor, welcher zur Programmierung in einer beliebigen Programmiersprache entwickelt wurde. Dieser Editor kann entweder als App auf den Computer geladen oder im Browser über https://vscode.dev/ erreicht werden. Außerdem hat der Editor eine große Auswahl von Plugins, wie zum Beispiel Live-Server, mit welchem ein Webserver lokal gehostet werden kann. Quellen: (Ionic[5])

2.5 Verwendete Bibliotheken

2.5.1 ngx-charts

Die ngx-charts Bibliothek bietet Zugriff auf eine breite Auswahl von Graphen. Diese Graphen können auf Wunsch abgeändert werden. So kann entschieden werden, ob und wo eine Legende angezeigt oder welche Farben verwendet werden sollen. Auch kann bei Daten entschieden werden, wie das Datum auf der X-Achse angezeigt werden soll. So wird bei dieser Arbeit auf der Achse nie die Anzahl der Minuten, sondern immer nur volle Stunden angezeigt.

2.5.2 Firebase packages

Für die Entwicklung dieser Diplomarbeit, wurden die folgenden zwei Firebase Pakete verwendet: `"firebase/auth"` und `"firebase/firestore"`. Das auth Paket ermöglicht den Zugriff auf den Authentifikationsservice von Firebase, wodurch Benutzer sich anmelden, registrieren oder ihr Passwort zurücksetzen können. Allerdings muss das Passwort nicht zwingend zurückgesetzt, sondern kann auch geändert werden. Das firestore Paket gewährt währenddessen eine Verbindung zur Firestoredatenbank, somit können Dokumente gelesen, bearbeitet, erstellt oder entfernt

werden. Hierfür gibt es verschiedene Sammlungen, in denen sich die Dokumente befinden können. Auch diese Sammlungen können mithilfe des Pakets erstellt oder gelöscht werden.

3 Implementierung

3.1 Authentifizierung

Zur Authentifizierung wurde Firestore verwendet. Zu Beginn muss sich bei der App registriert werden, das Interface für den Benutzer ist in Abbildung 1 zu sehen. Hierfür wird, wie von vielen anderen Applikationen gewohnt, E-Mail Adresse und Passwort benötigt. Im Hintergrund fragt die App die Datenbank, ob es schon Daten für diese E-Mail gibt und falls dies der Fall sein sollte, werden diese mit dem neu registriertem Account verbunden.

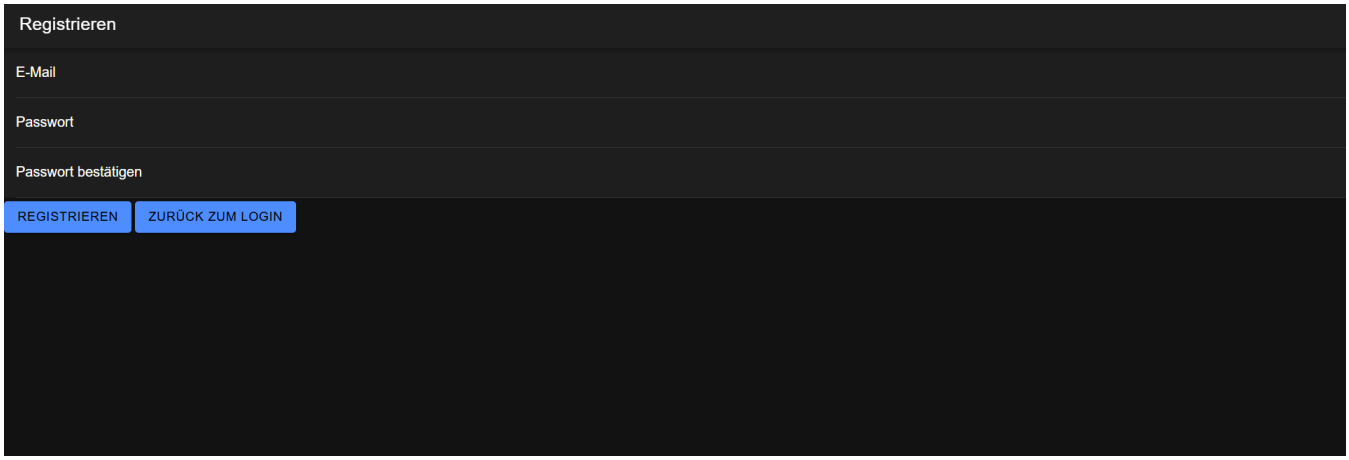

Abbildung 1: Registrierung für die App

Anschließend wird eine Verifizierungsmail von Firebase (siehe Abschnitt: 2.3), ausgeschickt welche bestätigt werden muss, wie in Abbilung 2 zu sehen. Sobald dies geschehen ist, kann das Einloggen erfolgen. Dies kann auch in Abbildung 3 erkannt werden. Falls ein Fehler gemacht wurde, wird eine Meldung am oberen Teil des Bildschirms eingeblendet, auch dies kann in Abbildung 3 erkannt werden. Eine solche Fehlermeldung gibt es auch bei der Registrierung.

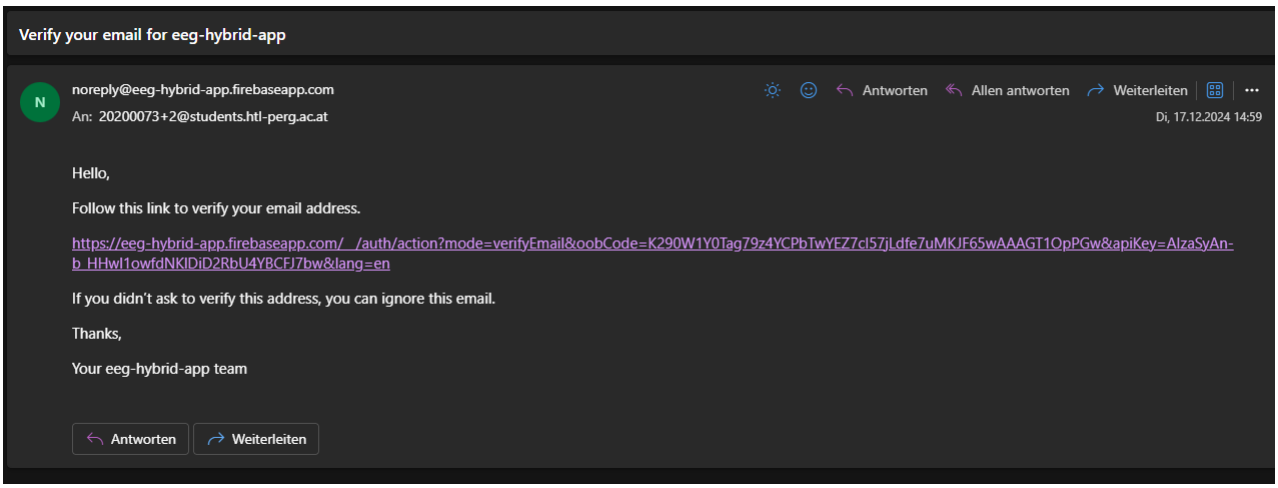

Abbildung 2: Verifikationsmail

In Abbildung 2 ist erkennbar, wie eine Verifikations E-Mail aufgebaut ist. Diese kann auf der Firebase Webseite angepasst werden. Zum Beispiel kann neben Hallo noch ein Benutzername geschrieben werden oder der Text im Allgemeinen überarbeitet werden. Sobald der Link, welcher in der E-Mail zu sehen ist, aufgerufen wird, wird der Account verifiziert und es kann sich eingeloggt werden.

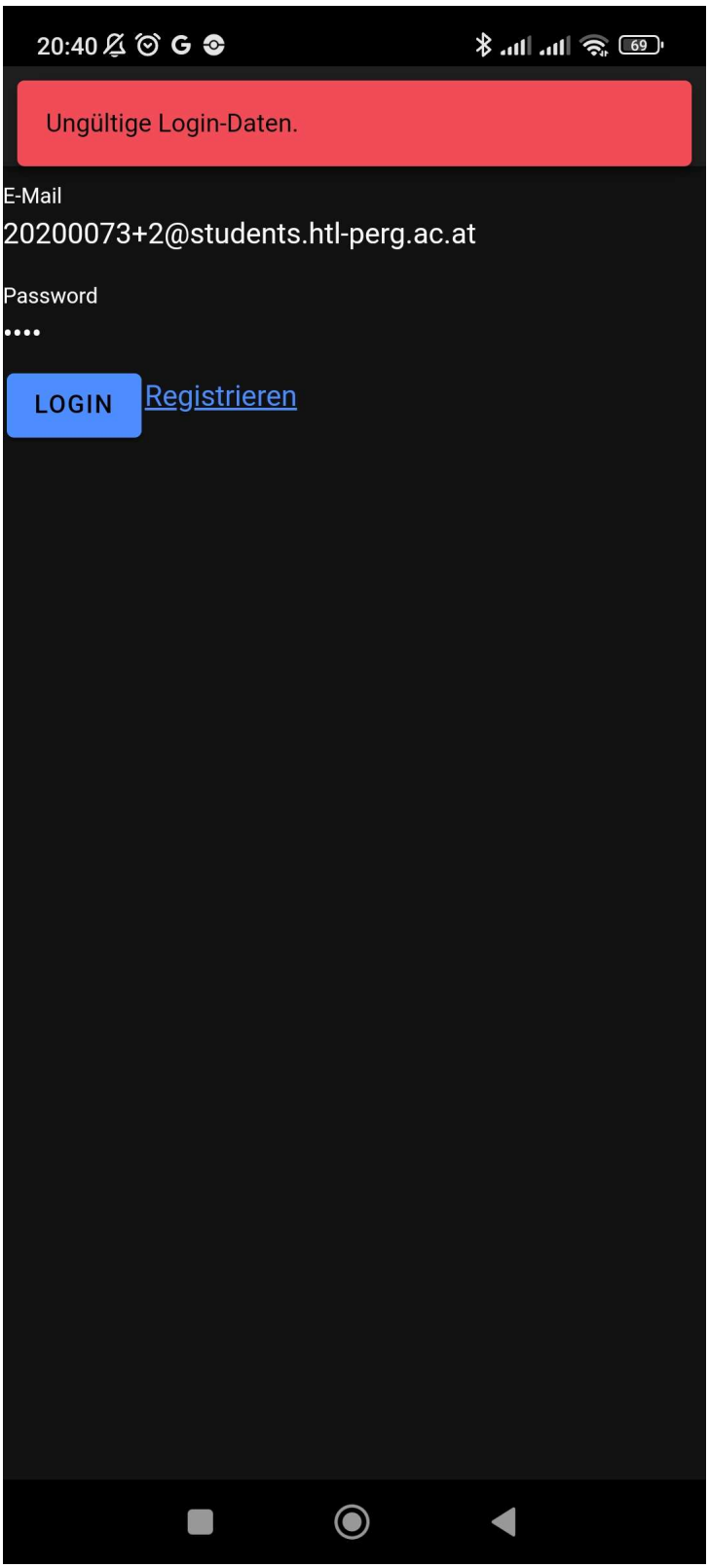

Abbildung 3: Login für die App

3.2 Laden der Daten mit firestore2.5.2

Nachdem Login wird begonnen, die Daten zu laden. Angefangen wird hierbei mit den Zählpunkten des Benutzers welche, wie in Abbildung 4 erkennbar, als spätere Auswahl für den Benutzer dienen. Sobald der Zählpunkt ausgewählt wurde, werden die Zählerzahlen geladen. Diese können durch eine weitere Auswahl eingeschränkt werden, damit keine unerwünschten Daten angezeigt werden. Der Datumsselektor welcher in der Abbildung zu sehen ist, ändert sich je nach gewählter Zeitspanne und ermöglicht das Auswählen des gewünschten Datums.

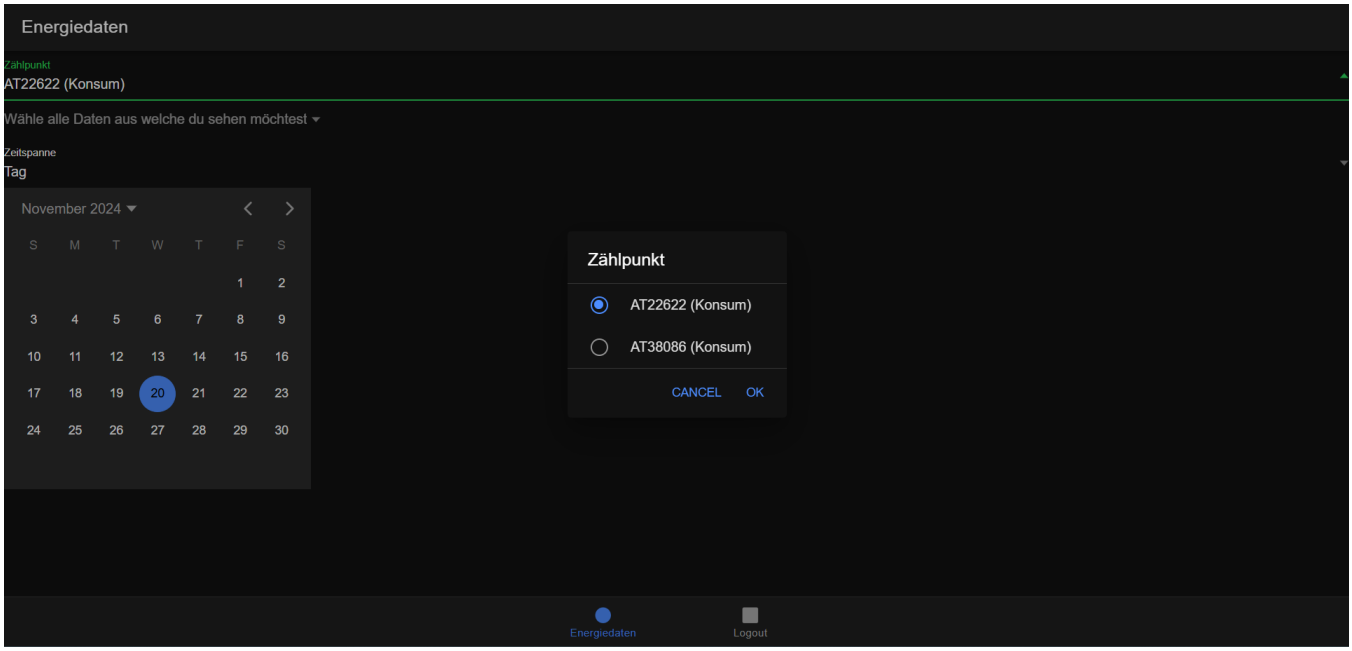

Abbildung 4: Optionen und Laden der Daten

Das in Abbildung 4 gezeigte Menü ermöglicht die Auswahl des Zählpunktes, von dem Daten geladen werden sollen. Nach der Auswahl ist es möglich auszuwählen, welche Daten des Zählpunktes angezeigt werden sollen.

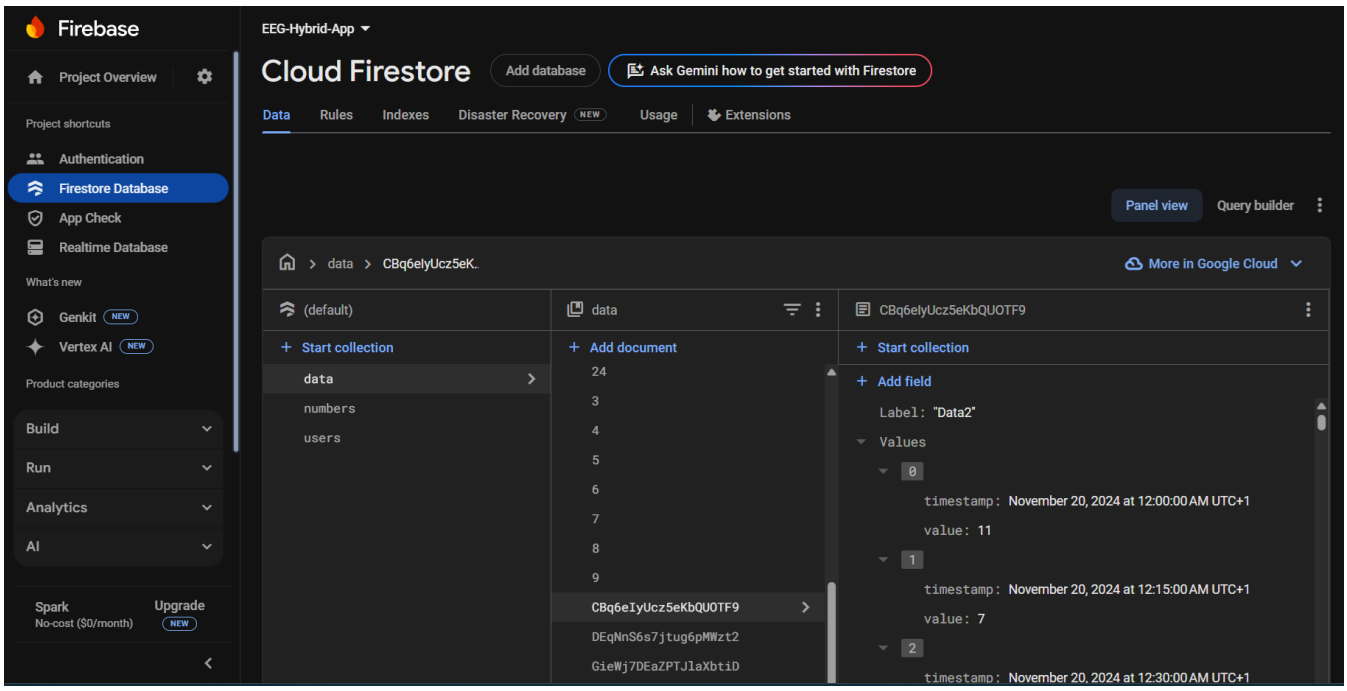

Abbildung 5: Anzeigen der Daten in der Datenbank

In Abbildung 5 ist zu sehen, wie eine Datenbank bei Firebase aufgebaut ist. In der linken Spalte können die Sammlungen gesehen werden. Für diese Arbeit gibt es hier die data, numbers und users Sammlungen. In der mittleren Spalte sind die Dokumente der ausgewählten Sammlung zu sehen und rechts der Inhalt des ausgewählten Dokuments. Im Vergleich zu einer relativen Datenbank kann man sich die Sammlung als Tabelle und die Dokumente als Einträge, also Zeilen, vorstellen. Hierbei ist allerdings zu beachten, dass die Firebase Datenbank, auch als Firestore bekannt, kein Schema hat, wodurch in jeder Zeile unterschiedliche Werte stehen könnten. So könnten zum Beispiel Benutzer gemeinsam mit Waren gespeichert werden.

3.3 Datenvisualisierung mit ngx-charts2.5.1

Um die zuvor geladenen Daten anzuzeigen wurde ngx-charts (siehe Abschnitt: 2.5.1) verwendet. Hierzu mussten die Daten in eine neue Form transformiert werden, welche von ngx-charts verlangt wird. Gesehen werden kann der Code hierfür in Listing 3 und das Format vor der Transformation ist in Listing 1 zu erkennen, wobei das Format nach der Transformation in Listing 2 gesehen werden kann. Diese Daten werden anschließend, je nach Zeitspanne, in Form eines Liniendiagramms, beziehungsweise eines Balkendiagramms angezeigt. Da aus dem resultierendem Graphen auf dem mobilen Endgerät, durch die geringere Bildschirmgröße, nicht mehr viel erkennbar ist, wie in Abbildung 7 zu sehen, musste auch eine Einschränkung nach Datum her. Da zur Speicherung der Zeit der Date Datentyp verwendet wurde, ist bei dem

Liniendiagramm schon automatisch eine, von ngx-charts zur Verfügung gestellte, Zeitspanne dabei, wie in Abbildung 6 ersichtlich, welche eine Möglichkeit bietet die Zeit einzuschränken. Hier ist alle 15 Minuten ein neuer Datenpunkt auffindbar, welcher durch das Anklicken oder Hovern, das Auslesen aller Energiedaten, welche zu diesem Zeitpunkt erfasst wurden, vereinfacht. Das Liniendiagramm (siehe Abbildung 6) ist allerdings nur zu sehen, wenn man als Zeitspanne Tag ausgewählt hat, ansonsten werden die Daten in Form eines Balkendiagramms visualisiert.

Listing 1: Datenformat vor Wandlung

```
[
  {
    "Label": string,
    "Values": [
      {
        "timestamp": Timestamp,
        "value": number,
      }
    ]
  }
]
```

Wie in Listing 1 zu sehen, wurden die Daten in einem Array gespeichert, wobei dieses Objekte sichert. Die Objekte speichern einen String namens Label und mehrere Zeitstempel mit dem Namen timestamp und nummern mit dem Namen value in dem Values Array.

Listing 2: Datenformat nach Wandlung

```
[
  {
    "name": string,
    "series": [
      "name": Date,
      "value": number,
    ]
  }
]
```

In Listing 2 kann erkannt werden, dass sich ein paar der Namen geändert haben. So ändert sich Label zu name, Values zu series und timestamp ebenfalls zu name. Dies ist das Format, welches von ngx-charts verlangt wird. Außerdem wird der Datentyp Timestamp zu Date umgewandelt.

Listing 3: Wandeln der Daten für ngx-charts

```
//Parameter: Daten welche Umgewandelt werden sollen.
convertDataToUse(data: Data) {
  let useData: UseData = {
    name: data.Label, //Label zu name
    series: data.Values.map((item: Value) => ({
      name: new Date(item.timestamp.toMillis()), //timestamp zu name und Datentyp Date
      value: item.value // value bleibt
    }))
  }
  return useData
}
```

In Listing 3 kann die Methode gesehen werden, welche zur Umwandlung des Datenformats implementiert wurde.

Listing 4: Wandeln der Daten für ngx-charts

```
1   private filterMulti() {
2     this.multi = [];
3     let aggDataStr: "not" | "day" | "month" | "year";
4     switch (this.timespan) {
5       case "Tag": aggDataStr = "not"; break;
6       case "Monat": aggDataStr = "day"; break;
7       case "Jahr": aggDataStr = "month"; break;
8       default: aggDataStr = "year"; break;
9     }
10    for (const origElement of this.orig) {
11      let aggData = this.aggregateDataByTimePeriod(origElement, aggData)
12      this.multi.push({
13        name: aggData.name,
14        series: aggData.series.filter((val) => {
15          switch (this.timespan) {
16            case "Tag": {
17              return val.name.getFullYear() === this.selectedDate.getFullYear() &&
                    val.name.getMonth() === this.selectedDate.getMonth() &&
18                val.name.getDate() == this.selectedDate.getDate()
19            }
20            case "Monat": {
21              return val.name.getFullYear() === this.selectedDate.getFullYear() &&
                    val.name.getMonth() === this.selectedDate.getMonth()
22            }
23            case "Jahr": {
24              return val.name.getFullYear() === this.selectedDate.getFullYear()
25            }
26            default: {
27              return true;
28            }
29          }
30        })
31      })
32    }
33    this.reformatData()
34    this.slicedMulti = this.transformedMulti;
35    this.checkAll()
36  }
```

In Listing 4 kann der Code gesehen werden, welcher verwendet wird, um nur die Werte des jeweiligen Tages/Monats/Jahr auszuwählen. Die Methode heißt `"filterMulti"`. In der Methode wird zuerst `"this.multi"` auf ein leeres Array gesetzt. `"this.multi"` ist das Array, wo die auf das Datum gefilterten Daten gespeichert werden. Als nächstes wird eine Hilfsvariable für eine Methode gesetzt, welche später ausgeführt werden soll. Die Variable wird hierbei, wenn `"this.timespan"` auf Tag ist, auf `"not"` gesetzt, bei `"Monat"` auf `"day"`, bei `"Jahr"` auf `"month"` und ansonsten auf `"year"`, `"this.timespan"` ist hierbei die Zeitspanne, welche der Nutzer sehen möchte. Danach wird jedes Element von `"this.orig"` durchlaufen, bei `"this.orig"` handelt es sich um die Energiedaten des Nutzers. Für jedes Element wird die `"aggregateDataByTimePeriod"` Methode aufgerufen, welche die Daten auf die gewünschte Größe summiert. Anschließend werden die Daten auf das ausgewählte Datum eingeschränkt, wobei es hier zu beachten gilt, dass die Einschränkung von der ausgewählten Zeitspanne abhängt. Zum Schluss werden die Daten noch in ein neues Format gebracht, falls ein Balkendiagramm angezeigt werden soll, was die Methode `"reformatData"` übernimmt und es werden noch ein paar Überprüfungen in der `"checkAll"` Methode vorgenommen.

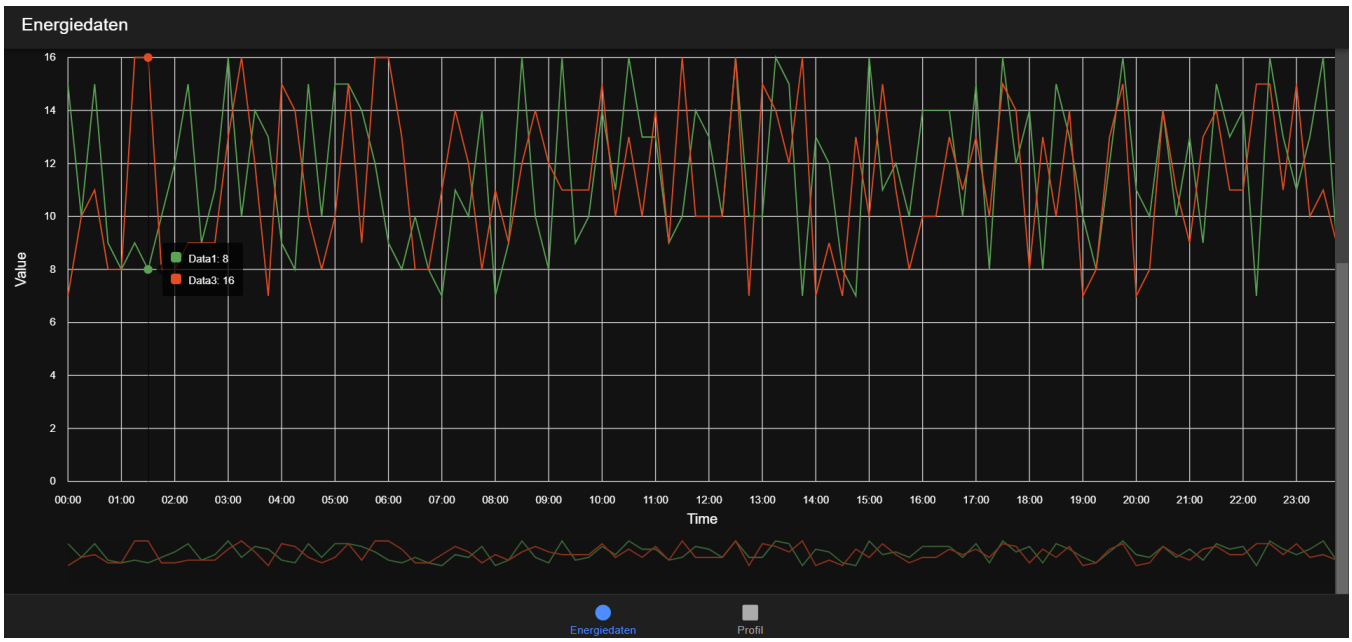

Abbildung 6: Anzeigen der Daten als Liniendiagramm

In Abbildung 6 ist die Visualisierung von Energiedaten mithilfe eines Liniendiagramms zu sehen. Wenn über den Linien gehovert wird, werden die genauen Werte für den nähesten Datenpunkt angezeigt. Auf dem Smartphone kann dies durch das Berühren des Bildschirms erzielt werden. Unter dem Diagramm ist auch die Zeitlinie zu sehen, mit welcher die Daten zeitlich eingeschränkt werden können.

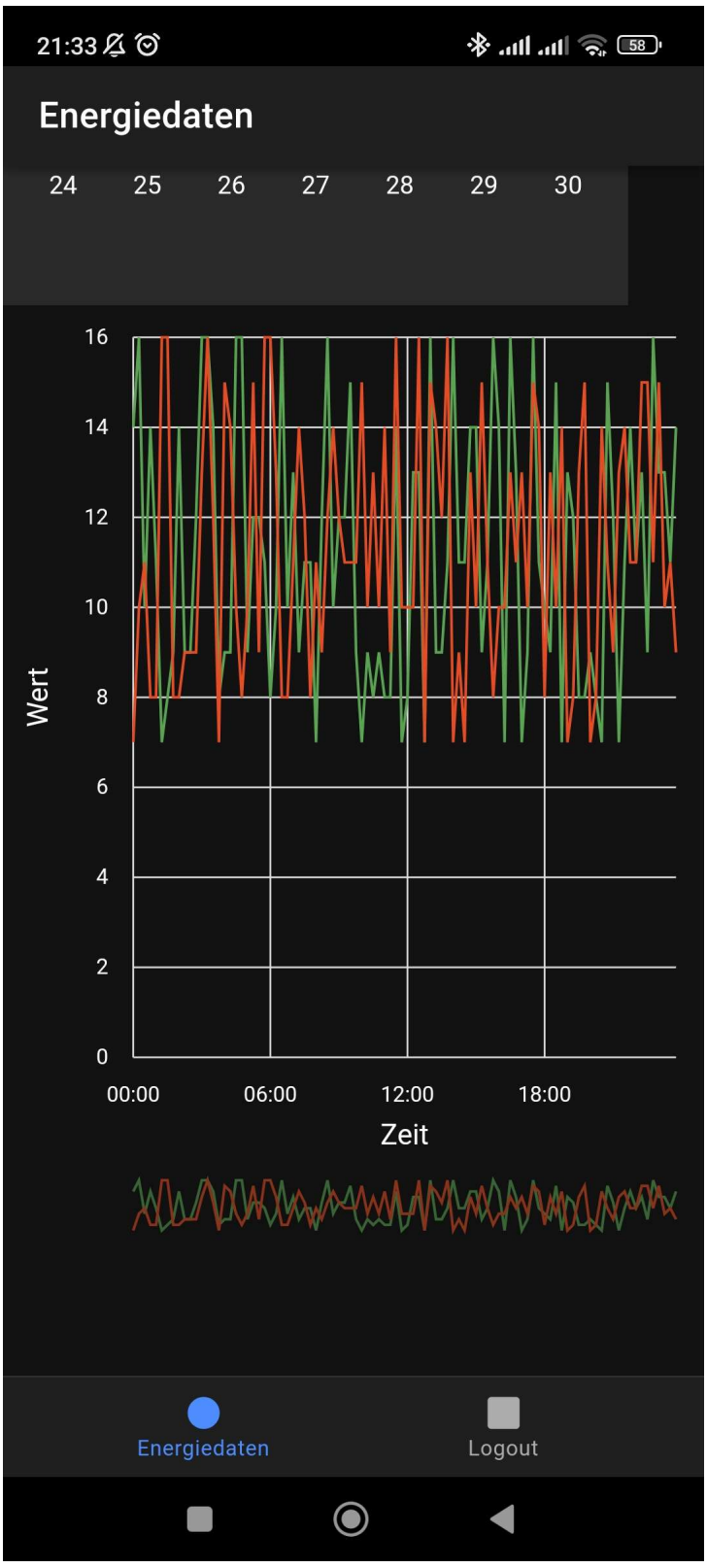

Abbildung 7: Schwer zu erkennen auf dem Smartphone

Allerdings war diese Zeitspanne beim Balkendiagramm nicht vorhanden, wodurch eine eigene Möglichkeit implementiert werden musste. Durch die Verwendung eines Schiebereglers mit zwei Punkten wird die Auswahl nun auch hier einschränkbar, wie dies aussieht kann in Abbildung 8 erkannt werden. Dieses Diagramm wird angezeigt, wenn bei Zeitspanne etwas anderes als Tag

ausgewählt ist. Bei Monat zeigt es die Summe aller Werte des ausgewählten Monats gruppiert nach Tagen, bei Jahr die Werte des Jahres gruppiert nach Monaten und bei Alles werden alle Werte angezeigt und nach Jahren gruppiert. Auch hier musste das Format der Daten verändert werden.

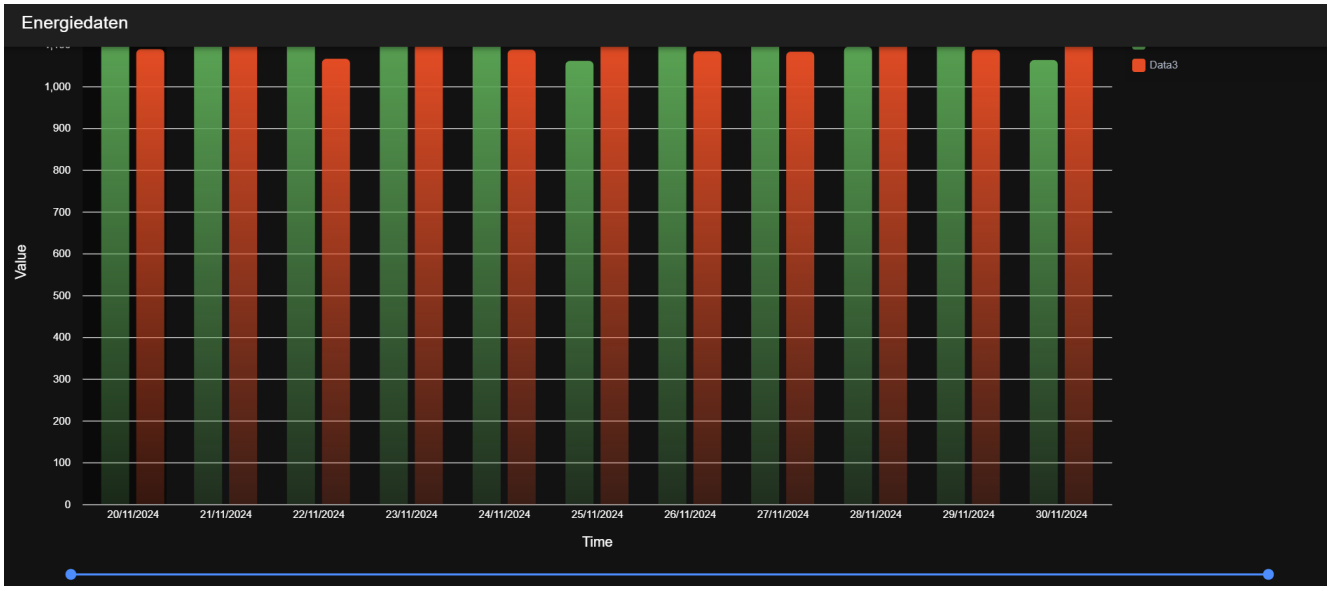

Abbildung 8: Anzeigen der Daten als Balkendiagramm

4 Ergebnis

Das Ergebnis ist eine Webapp und eine Mobile App, wobei beide eine Benutzeransicht und Adminansicht haben.

4.1 Web App

Die Web App kann auf einen Webserver geladen werden und von verschiedenen Personen verwendet werden. Nachdem ausgewählt wurde, welche Daten angezeigt werden sollen, wird ein Liniendiagramm oder Balkendiagramm angezeigt. Das Liniendiagramm erscheint, wenn als Zeitspanne `"Tag"` ausgewählt wird und zeigt die einzelnen Werte eines Tages, wobei in der Datenbank für jede Viertelstunde ein neuer Wert existieren kann. Außerdem befindet sich unter dem Liniendiagramm ein Zeitstrahl, mit dem das Diagramm auf einen Zeitrahmen eingeschränkt werden kann. Das Balkendiagramm wird bei jeder anderen Zeitspanne verwendet und zeigt die Summen der nächst kleineren Zeitspanne. So werden zum Beispiel bei der Zeitspanne "Monat", die Summen der einzelnen Tage angezeigt. Beim Balkendiagramm gibt es keinen Zeitstrahl, stattdessen wurde ein Selektor hinzugefügt, welcher zur Einschränkung der Zeit, als Ersatz für den Zeitstrahl fungiert.

Abbildung 9: Datenansicht im Web

4.2 Mobile App

Für mobile Endgeräte kann die App auch installiert werden. Damit die App den Inhalt auch anzeigen kann, wird von Ionic eine WebView verwendet. WebView ist eine Art eingebetteter Browser, welcher ermöglicht Web-Apps anzuzeigen. Die App funktioniert genau wie im Browser und sieht wie eine normale App auf dem Smartphone aus. Webviews helfen Entwicklern, indem die selbe Codebasis für mobile Apps und Webanwendungen verwendet werden kann, um die App genau wie im Browser darzustellen.

Zusätzlich zur WebView sorgt Ionic dafür, dass die App auf verschiedenen mobilen Betriebssystemen wie Android und iOS funktioniert, ohne dass für jede Plattform ein separater Code geschrieben werden muss. Dies spart Zeit und Ressourcen und ermöglicht es Entwicklern, eine einheitliche Benutzererfahrung auf allen unterstützten Geräten zu bieten. Die App funktioniert also genau wie im Browser, ist jedoch als eigenständige Anwendung auf dem Smartphone installiert und bietet die Benutzerfreundlichkeit einer nativen App.

In Abbildung 10 ist zu erkennen, dass diese App durch die Verwendung des Schiebereglers, die Daten dennoch sinnvoll auf dem Smartphone anzeigen kann. Auch zeigt es gemeinsam mit Abbildung 11, dass die Applikation sowohl Quervormat als auch Hochformat und sowohl den dunklen als auch den hellen Modus unterstützt.

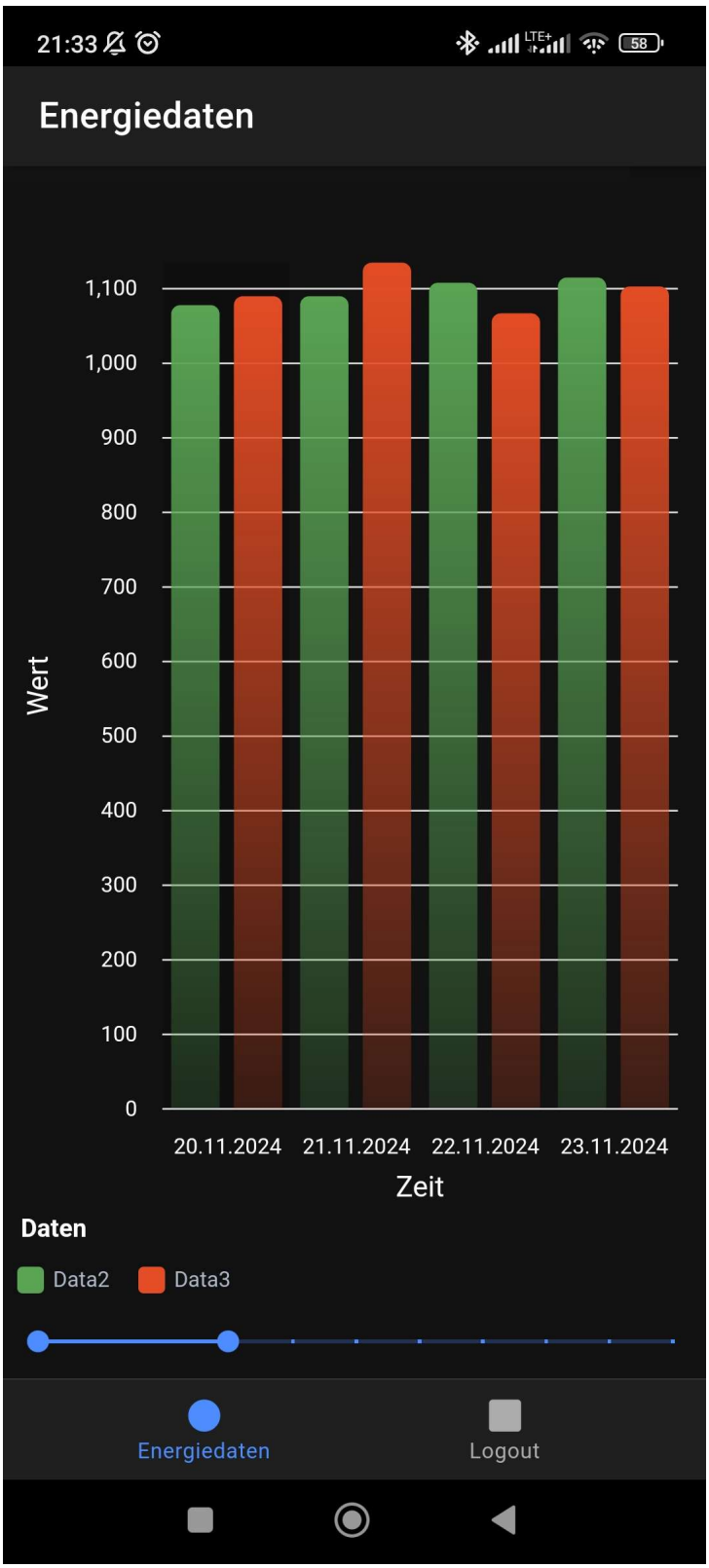

Abbildung 10: Hochformat mit Dunkelmodus

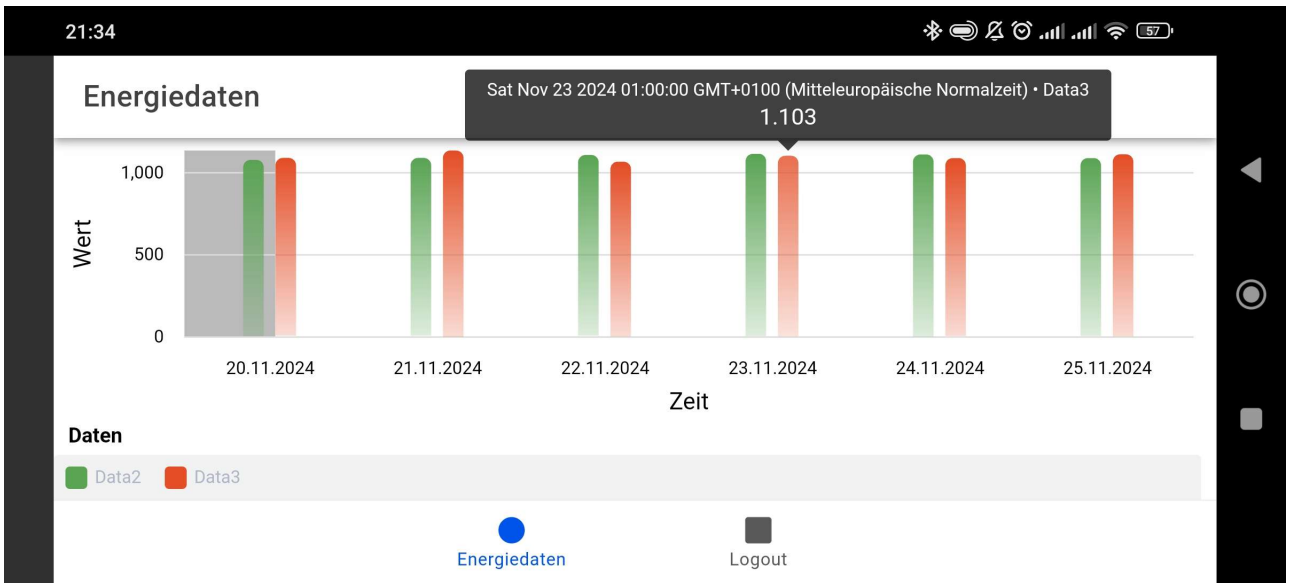

Abbildung 11: Querformat mit hellem Modus

4.3 User View

Die Benutzeransicht ist die Ansicht, welche von den meisten Personen gesehen wird. Hier können eingeloggte Nutzer den gewünschten Zählpunkt auswählen und danach ihre eigenen Daten, auf einen bestimmten Zeitrahmen einschränken und Zählwerte ausblenden. Bei der Auswahl der Zählpunkte kann sofort gesehen werden, ob dieser ein Konsument oder ein Produzent ist. Sobald der Benutzer alle Daten, welche er abrufen wollte gesehen hat, kann er sich mit der in Abbildung 12 zu sehenden Seite ausloggen.

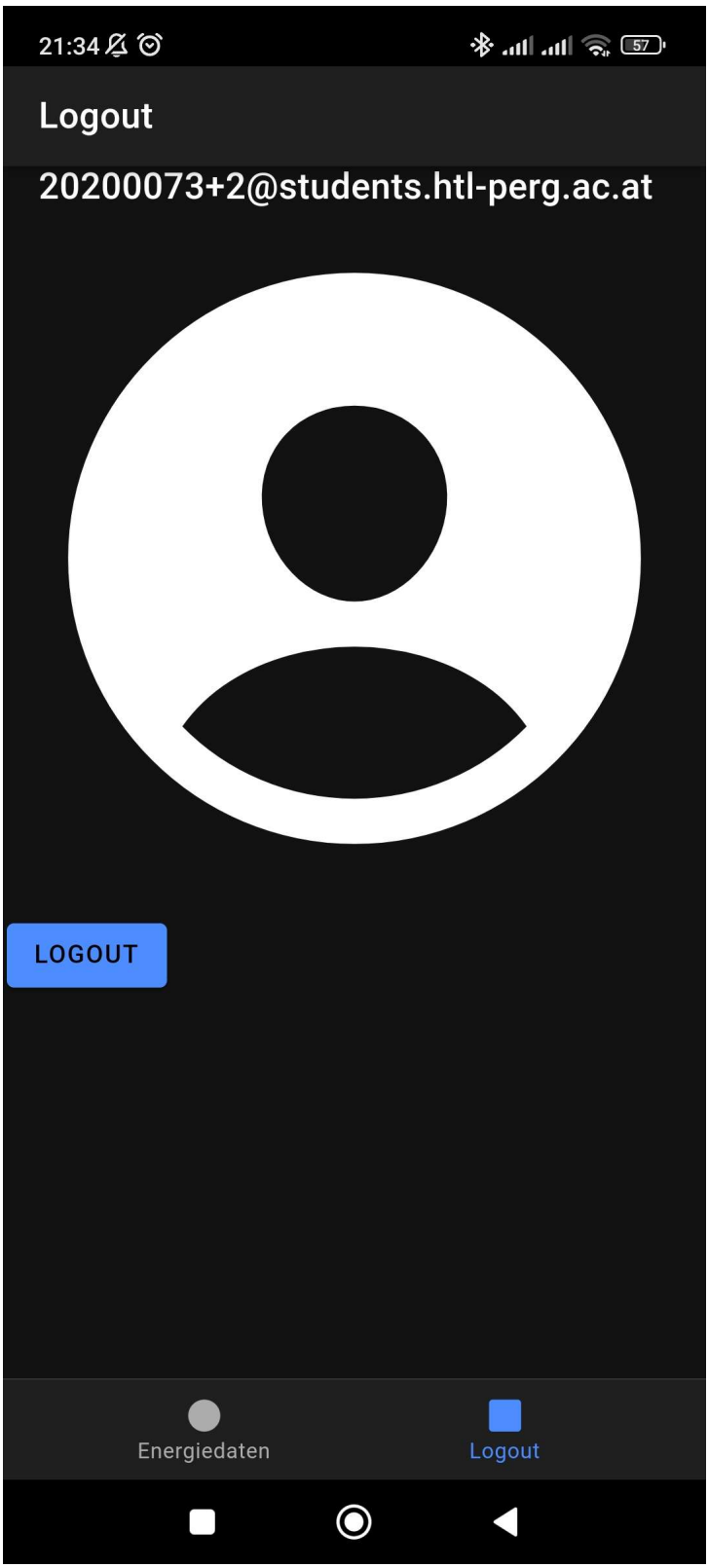

Abbildung 12: Ausloggen

4.4 Admin View

Im Gegensatz zur Benutzeransicht, kann die Adminansicht nur von bestimmten Personen gesehen werden. Sie unterscheidet sich zur Benutzeransicht in dem Punkt, dass auch die Daten von anderen gesehen werden können. Der Benutzer dessen Daten eingesehen werden sollen, kann durch ein Feld ausgewählt werden. Als Standardwert ist hier der eigene eingeloggte Benutzer festgelegt. Die übrigen Felder sind wie bei der Benutzerauswahl.

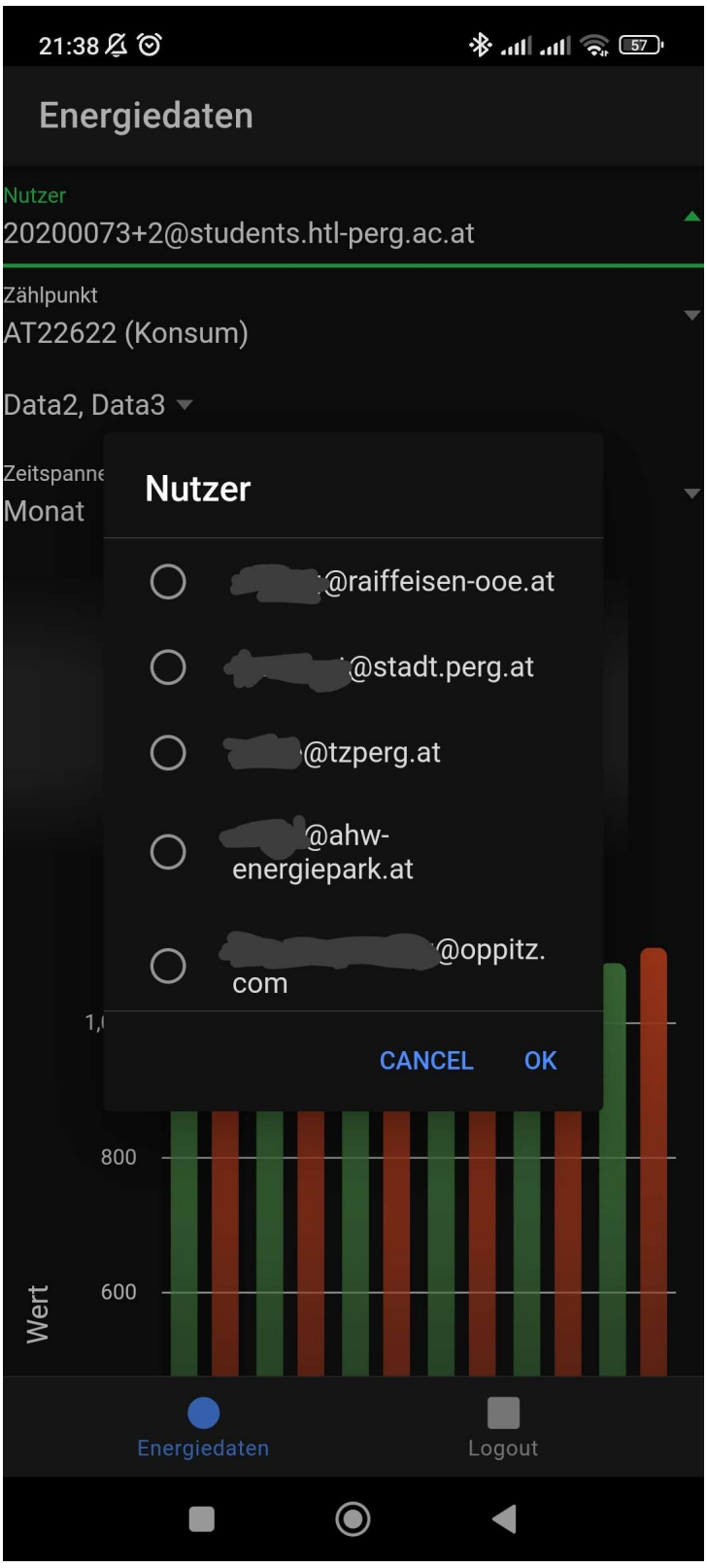

Abbildung 13: Benutzerauswahl für Administratoren

Wie in Abbildung 13 erkennbar, ist hier das Menü für die Benutzerauswahl geöffnet. Nachdem ein Benutzer ausgewählt wurde, werden seine Zählpunkte geladen und diese können angezeigt werden. Der eingeloggte Nutzer wird hierbei wie der ausgewählte Nutzer behandelt und hat auf die

selben Daten Zugriff. Natürlich ist dies nur für Benutzer verfügbar, welche als Administratoren eingetragen sind.

5 Resümee

Diese Arbeit wurde in Zusammenarbeit mit der Energiegenossenschaft Perg entwickelt und soll Mitgliedern, eine bequeme Übersicht über die eigenen Energiedaten und ihre Entwicklung ermöglichen. Im Rahmen dieser Arbeit war es mir möglich neue Themenbereiche zu erlernen und mein Wissen zu erweitern.

Ein paar dieser Themen wären:

- Ionic
- Laden von Daten mit Firestore
- Anzeigen von Daten mit Graphen

Während der Implementierung sind allerdings auch ein paar Herausforderungen aufgetreten. Ein Beispiel hierfür wäre, dass die tatsächlichen Daten erst spät in die Datenbank übertragen wurden, wodurch die tatsächliche Funktionalität erst gegen Ende des Entwicklungsprozesses überprüft werden konnte. Auch erwähnenswert ist die Umwandlung der von der Datenbank abgerufenen Daten in ein von ngx-charts unterstütztes Format.

Literaturverzeichnis

[1] Association for Progressive Communications, „What is Angular?" letzter Zugriff am 16.03.2025. Online verfügbar: https://angular.dev/overview

[2] Chinmayee Deshpande, „What Is Angular? A Guide to the Angular Framework," 2024, letzter Zugriff am 16.03.2025. Online verfügbar: https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular

[3] Ionic, „Ionic Framework - The Cross-Platform App Development Leader," letzter Zugriff am 17.03.2025. Online verfügbar: https://ionicframework.com/

[4] Jetbrains, „Webstorm: Die JavaScript- und TypeScript-IDE von Jetbrains," letzter Zugriff am 17.03.2025. Online verfügbar: https://www.jetbrains.com/de-de/webstorm/

[5] VSCode, „Visual Studio Code - Code Editing. Redefined," letzter Zugriff am 17.03.2025. Online verfügbar: https://code.visualstudio.com/

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang