HAW HAMBURG

BACHELORTHESIS
Finn Isengardt

Entwurf und prototypische Umsetzung eines Data-Service zur Verarbeitung und Regressionsanalyse von Energieverbrauchsdaten

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Finn Isengardt

Entwurf und prototypische Umsetzung eines Data-Service zur Verarbeitung und Regressionsanalyse von Energieverbrauchsdaten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Marina Tropmann-Frick

Eingereicht am: 29. November 2022

Finn Isengardt

Thema der Arbeit

Entwurf und prototypische Umsetzung eines Data-Service zur Verarbeitung und Regressionsanalyse von Energieverbrauchsdaten

Stichworte

Software-as-a-Service, REST, Software-Engineering, Regressionsanalyse, Python, Django, Pandas, Energiewende

Kurzzusammenfassung

In dieser Thesis wird in Zusammenarbeit mit der Energy Data GmbH ein Web-Tool zur Verarbeitung und Analyse von Energieverbrauchsdaten entworfen und prototypisch umgesetzt. Die Verarbeitung der Daten umfasst Formatierung und Speicherung, die Normalisierung von Wettereinflüssen, sowie die Analyse und Vorhersage von Verbräuchen mittels Regressionsanalyse. In dieser Thesis wird der Software-Engineering-Prozess für das Backend der EnergyData-Tools mit Hinblick auf die Anforderungen des Energiemanagements an einen Data-Service beleuchtet.

Finn Isengardt

Title of Thesis

Design and implementation of a Data-Service to process and analyze using stastical regression of energy-consumption data.

Keywords

Software-as-a-Service, REST, Software-Engineering, Regressionsanalysis, Python, Django, Pandas, Climate Change

Abstract

This thesis covers the design and implementation of a web-service platform for processing and analysing energy-consumption data. The data-centric services will be developed in cooperation with the Energy Data GmbH. The processing of energy-consumption data includes the formatting and storing of datasets, as well as using techniques energy management techniques to normalize the data for further analysis and predicitons using linear regression models. The particular focus of this thesis is to demonstrate the software-engineering process regarding the challenges of data centric-software in the energy management field.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

LTA	Long-Term Adjustment (Witterungsbereinigung durch Langzeitdurchschnitt von Gradtagszahlen)
BYA	Base-Year Adjustment (Witterungsbereinigung durch die Gradtagszahlen eines Basisjahres)
ER	Efficiency-Ratio (Energieeffizienzkennzahl)
HDD	Heating-Degreedays (Gradtagszahl für Heizbedarf)
CDD	Cooling-Degreedays (Gradtagszahl für Kühlbedarf)

Glossar

EnergyData-Tools	Die neuen, zusammengeführten EnergieTools der Energy Data GmbH. Der Begriff umfasst das in dieser Arbeit beschrieben Backend-System sowie das parallel entwickelte Frontend der neuen Tools.
Normalisierung	Methoden zur Bereinigung der Witterungseinflüsse auf Energieverbrauchsmessungen.
Degreedays	Degreedays (oder Gradtagszahlen) sind eine Maßeinheit aus dem Energiemanagement zur Ermittlung von Heiz- und Kühlbedarf auf Basis der Außentemperatur und einer Basisgrenztemperatur.
Rawdataset	Rohdatensatz aus Energieverbrauchsmessungen, der im Systemkontext nicht normalisiert oder analysiert wurde.
Normdataset	Ein witterungsbereinigter, normalisierte Datensatz aus Energieverbrauchsmessungen.
Registrierter User	Ein registrierter User auf der Energy Data Website, welcher eingeschränkten Zugriff auf die Data-Services hat.
Subscription User	Ein Bezahlkunde, welcher vollen Zugriff auf alle Data-Services hat.

1 Einleitung

1.1 Motivation

Die Folgen des Klimawandels sind weltweit spürbar, weshalb der Klimaschutz von globalgesellschaftlichem Interesse ist. Im Rahmen des europäischen Klimagesetzes, welches am 29. Juli 2021 in Kraft getreten ist, verpflichten sich Mitgliedstaaten der Europäischen Union zu einer Reihe von Richtlinien und zur nationalen Umsetzung von Maßnahmen zur Erreichung der Klimaneutralität bis zum Jahre 2050 (Publications Office of the European Union, 9.7.2021).

Die Energieeffizienzwende stellt wirtschaftlich, politisch so wie gesellschaftlich erhebliche Herausforderungen im Kontext des Klimaschutzes dar. Die Bewertung und Verbesserung von Energieverbräuchen und die Umsetzung von Energieeffizienzmaßnahmen kann Unternehmen einen wirtschaftlichen Vorteil bieten und Innovation fördern (Doty, 2007, p. 17).

Die Rolle der Informatik ist von großer Bedeutung für die Erreichung der Klimaziele und ihre Anwendungsfälle sind vielseitig. Insbesondere in der Erzeugung von erneuerbarer Energie, der Klimawandelforschung und dem Energiemanagement werden Informationssysteme umfangreich eingesetzt, um den Wandel in der Wirtschaft und Gesellschaft voranzutreiben.

Im Rahmen dieser Arbeit werden die EnergyData-Tools entstehen, welche die Verarbeitung und Analyse von Energieverbrauchsdaten ermöglicht. Unternehmen, Energiemanager und Verbraucher können dieses Angebot nutzen und die EnergyData-Tools in bestehende Arbeitsprozesse und Energiemanagementsysteme einbinden.

1.2 Energy Data GmbH

Die Energy Data GmbH wurde 2021 in Hamburg gegründet und bietet seitdem in dem Bereich Energy-Data-as-a-Service Lösungen an. Das Leistungsportfolio beinhaltet außerdem die Beratung von Unternehmen zur Umsetzung und Einführung von Energiemanagementsystemen und die Zertifizierung nach ISO 50001.

Das Produktangebot der Energy Data GmbH wird beständig erweitert. Seit 2017 wird eine Website im Energie Consulting Bereich betrieben. Seit 2019 wird der Degree Day Calculator und die Degreeday-API angeboten und ist für Kunden über ein Subscription-System nutzbar.

Die Energy Data GmbH unterstützt mit ihrem Software-as-a-Service Angebot ihre Kunden bei der Auswertung der bestehenden Energieverbrauchsdaten durch die Bereitstellung von externen Energiedaten sowie bei der Analyse und Optimierung des Energieverbrauchs (Energy Data GmbH, 2022).

1.3 Zielsetzung

Ziel dieser Arbeit ist es, eine Softwarelösung für das Backend der EnergyData-Tools zu entwerfen und umzusetzen, welche die Verarbeitung und Analyse von Energieverbrauchsdaten ermöglicht. Energiemanager können diese nutzen, um Energieeffizienzmaßnahmen zu bewerten und die Identifikation von Einsparpotentialen zu ermöglichen. Durch die Entwicklung und Dokumentation einer REST-API für das Backend und einer parallel entwickelten Weboberfläche können die EnergyData-Tools in bestehende Energieeffizienzmanagementsysteme eingebunden werden oder von Energiemanagern über eine Weboberfläche verwendet werden.

Im Vordergrund dieser Arbeit stehen der Software-Engineering-Prozess und die prototypische Umsetzung einer zuverlässigen und erweiterbaren Backend-Architektur. Das Ziel dieser Arbeit ist es, die Entscheidungsprozesse, sowie eine adäquate Dokumentation des Systems darzustellen. Abschließend soll der gesamte Software-Engineering-Prozess kritisch hinterfragt werden.

2 Grundlagen

2.1 Energiemanagement

Energiemanagement bezeichnet die aktive Beschäftigung mit dem Thema Energie in Gebäuden, gebäudetechnischen Anlagen sowie nutzerspezifischen Unternehmens- und Produktionssystemen. Dabei ist die zentrale Aufgabe des Energiemanagements, den Verbrauch und die damit verbundenen Kosten zu optimieren (GEFMA, 2020, p. 3).

Für Unternehmen gibt es viele positive Aspekte, Energiemanagement zu betreiben, Energiemanagementsysteme einzuführen und Energieeffizienzwandel als Teil der Unternehmenskultur zu leben. Um diese Vorhaben in Unternehmen zu fördern, wurden im Kontext des Klimawandels in Deutschland unterschiedliche Gesetze, Handlungsempfehlungen und Normen eingeführt, welche den Energieverbrauch bis 2050 um 50 Prozent gegenüber 2008 senken soll (GEFMA, 2020, p. 5).

Unternehmen profitieren auf unterschiedliche Weise von Energiemanagement und insbesondere der Einführung von Energiemanagementsystemen. Die Unternehmen sind in der Lage, ihren Energieverbrauch fortlaufend und zukunftsorientiert zu organisieren und Kosten zu senken. Des Weiteren hat auch das Image eines Unternehmens in Bezug auf Umweltbewusstsein einen hohen Stellenwert.

Laut dem GEFMA-Arbeitskreis Energie profitieren Unternehmen von weiteren Aspekten, wenn effektives Energiemanagement betrieben wird:

- Transparenz über tatsächlichen Verbrauch/Kosten
- Prädiktive Betriebsführung zur Kostenoptimierung in Echtzeit
- Prädiktive Wartung zur Erhöhung der Ausfallsicherheit und Reduzierung der Kosten
- Umfangreiche Kenntnis über Gebäude- und Nutzungsstruktur

2.1.1 Energiedatenanalyse nach ISO 50001

Die ISO 50001 strebt ein ganzheitliches Energiemanagement an, wobei alle Aspekte in dem Umgang mit Energie gleichermaßen und integriert diese in die Strategie und Organisation eines Unternehmens (Geilhausen, 2020, p. 1). In dieser Norm wurde versucht, einen einheitlichen Standard zu schaffen, welcher Unternehmen dazu bewegt, alles für die Einsparung von Energie zu tun. Diese Standardisierung soll die Einsparungen messbar und vergleichbar machen (Simone Brugger-Gebhardt, 2019, p. 3).

Die energetische Bewertung und die Analyse des Verbrauchs sind wesentlicher Bestandteil für die Energiedatenanalyse nach ISO 50001. Die in dieser Arbeit entworfene Software ist nach ISO 50001 in die Analyse des Verbrauchs und Einsatzes und der Energetischen Bewertung einzuordnen.

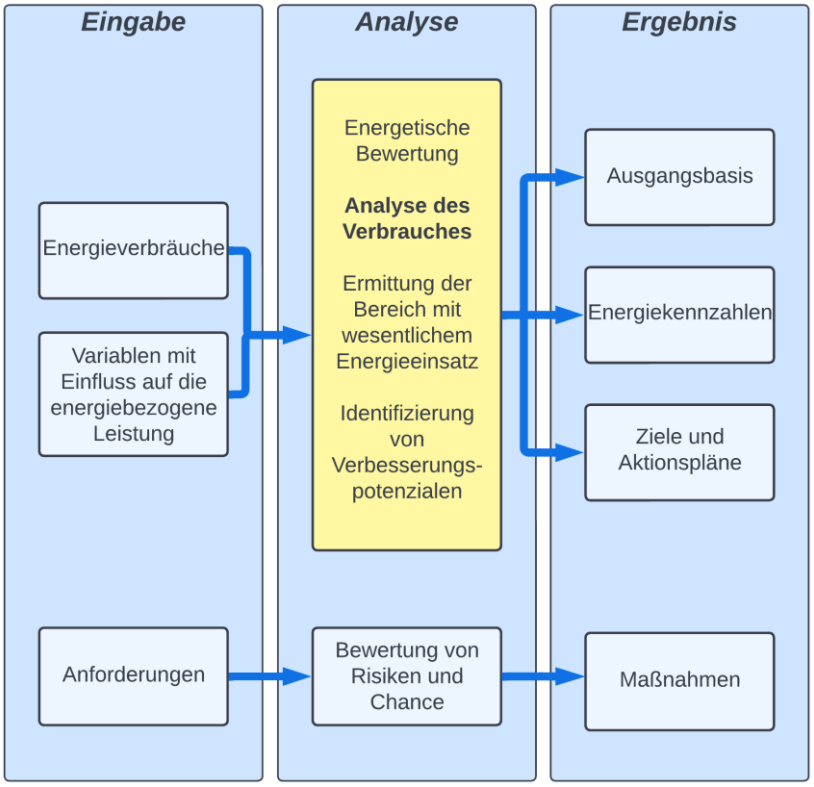

Abbildung 1 Energiedatenanalyse Übersicht (angelehnt an (Petermann, 2018))

2.2 Bereinigung der Energieverbrauchsdaten (Normalisierung)

Die energetische Bewertung nach ISO 50001 erfordert den Vergleich von energiebezogenen Leistungen. Um einen Vergleich unter gleichwertigen Bedingungen zu ermöglichen, müssen Energieverbrauchsdaten häufig von Einflüssen bereinigt werden.

Ein Anwendungsfall in der energetischen Bewertung ist es, die Energieverbrauchsdaten von Wettereinflüssen und Klimaschwankungen zwischen den Jahren zu bereinigen (Geilhausen, 2020, p. 31). Das Wetter hat direkten Einfluss auf die Energieverbrauchsdaten, denn es kann beispielsweise besonders harte oder milde Winter geben. Wenn die Temperaturen in einem Winter besonders niedrig waren, kommt es zur Abweichung des Verbrauchs durch diesen extremen Wettereinfluss. Es ist bei einem Vergleich des Messungszeitraums mit anderen Messungszeiträumen nicht immer nachvollziehbar, ob ein erhöhter Verbrauch durch besonders kalte Tage entstanden ist, oder ob sich der Energieverbrauch durch andere Einflüsse verändert hat.

Durch die Normalisierung entstehen Energieverbrauchswerte, die durch historische Wetterdaten korrigiert wurden und ermöglicht eine Betrachtung der Energieverbrauchsdaten unabhängig von Wettereinflüssen.

2.2.1 Gradtagszahlen

Die Wetterdaten können als Gradtagszahlen, bzw. Degreedays abgebildet werden. (Geilhausen, 2020, p. 31) Bei der Normalisierung der vorgestellten Methoden werden die Gradtagszahlen als Korrekturfaktor genutzt.

Die Gradtagzahl ist die Summe aus den Differenzen einer angenommenen Basistemperatur (Rauminnentemperatur) von beispielsweise 20 °C und dem jeweiligen Tagesmittelwert der Außentemperatur über alle Tage eines Zeitraums, an denen dieser unter der Heizgrenztemperatur des Gebäudes liegt. (Simone Brugger-Gebhardt, 2019, p. 74)

Feiner unterschieden wird auch zwischen Heating Degreedays und Cooling Degreedays, welche den Kühl- und Heizbedarf abbilden.

2.2.2 Effizienz-Kennzahl

Die Effizienz-Kennzahl (ER) von Verbrauch und Gradtagen ist ein Berechnungsverfahren zur Bereinigung der Wettereinflüsse (Simone Brugger-Gebhardt, 2019, p. 86). Der Verbrauch wird durch die Summe der Gradtagzahlen in einem bestimmten Zeitraum dividiert. Das Ergebnis ist die Energieeffizienzkennzahl. Der Verbrauch pro Gradtagszahl kann anschließend mit anderen Zeiträumen verglichen werden. Wenn das ermittelte Verhältnis sinkt, ist die Energieverbrauchseffizienz gestiegen.

Für alle Messzeiträume in dem Energieverbrauchsdatensatz wird diese Formel angewendet:

$$\frac{Energieverbrauch}{Gradtagszahl}$$

2.2.3 Langzeitdurchschnitt – Witterungsbereinigung

Die Langzeitdurchschnitts-Bereinigung (LTA – Long-Term Average) ist ein Berechnungsverfahren, welches den Langzeitdurchschnitt von Gradtagszahlen nutzt, um die Energieverbrauchsdaten von Wettereinflüssen zu bereinigen (Simone Brugger-Gebhardt, 2019, p. 74).

Bei dem LTA werden für eine Messperiode, beispielsweise den März, die gemessenen Temperaturen der letzten zehn Jahre in genau diesem Zeitraum zu einem Durchschnittswert aggregiert und anschließend wird der eigentliche Verbrauch normalisiert. Hierfür ist es nötig, den Standort der Messdaten zu kennen und die Wetterdaten der am nächsten gelegenen Wetterstation zu nutzen. Es entstehen Messwerte, welche durch den Langzeitdurchschnitt normalisiert wurden und von Witterungseinflüssen bereinigt wurden.

Für alle Messzeiträume in dem Energieverbrauchsdatensatz wird diese Formel angewendet:

$$\frac{Gradtagszahl}{Langzeitdurchschnitt \times Energieverbrauch}$$

2.2.4 Basisjahr – Witterungsbereinigung

Bei der Normalisierung durch eine energetische Ausgangsbasis, einem Basisjahr (BYA), werden die Verbrauchswerte anhand der Gradtsgszahlen eines feststehenden Jahres und der Gradtagszahl der zu normalisierenden Messzeitraums normalisiert. Hierfür müssen die Messdaten einen gewissen Zeitraum abdecken, also mindestens ein vollständiges Basisjahr enthalten. Alle Messdaten, die nach diesem Basisjahr kommen, werden mit dessen Messwerten verglichen und korrigiert (Petermann, 2018).

Für alle Messzeiträume in dem Energieverbrauchsdatensatz wird diese Formel angewendet:

$$\frac{Gradtagszahl}{Basisjahr\ Gradstagszahl} \times Energieverbrauch$$

2.3 Lineare Regressionsanalyse

Im Rahmen einer linearen Regressionsanalyse soll der beobachtete Zustand einer abhängigen Variablen (Ergebnisvariablen) in Zusammenhang mit einer oder mehreren unabhängigen Variablen gebracht werden. Praxisbezogen kann so z. B. dem Ziel nachgegangen werden, einen messbar vorliegenden Energieverbrauch in Abhängigkeit von der Außentemperatur darzustellen. Grundlage für solch eine mathematische Analyse zur Abbildung der Realität ist, dass mehrere Messungen oder Daten der einbezogenen Größen vorliegen.

Ein erstelltes Modell kann herangezogen werden, um allgemein einen Überblick über die Abhängigkeiten der Zielgröße von den Einflussfaktoren zu erhalten oder auch damit Entwicklungen in der Zukunft einzuschätzen. Dabei kann entweder von einem oder mehreren Einflussfaktoren ausgegangen werde.

Am Beispiel des Energieverbrauchs liegt nahe, dass sich die Abhängigkeit meistens nicht nur auf einen Einflussfaktor, sondern auf mehrere bezieht, denn neben der Außentemperatur mögen auch die vorliegende Dämmung, die Lage oder die Anzahl der technischen Geräte im Haus entscheidend den Verbrauch beeinflussen. In diesem ausgeweiteten Fall mit mehreren Dependenzen spricht man von einer multiplen, anstatt einer einfachen Regressionsanalyse.

In dieser Arbeit wird auf den Fall der linearen Regressionsanalyse eingegangen, d. h. das Modell ist linear in allen Faktoren.

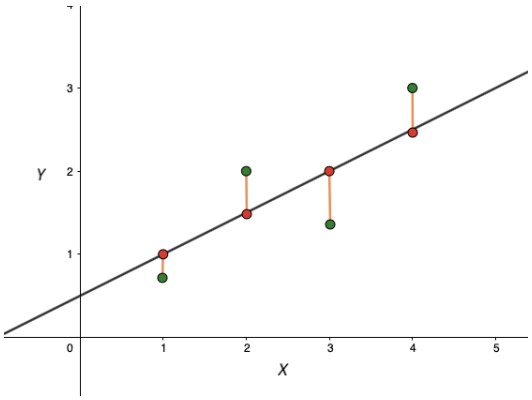

Abbildung 2 Das lineare Modell

2.3.1 Modellbildung mittels Least Squares

Im Falle einer linearen Regression sind für die Modellbildung n Messungen der Ergebnisvariable *Y* und der zugehörigen Abhängigkeit *X* bekannt. Hinzu kommen zwei noch unbekannte Parameter β_0 und β_1, die die Modellgerade definieren, sowie ein Ausgleichsparameter ε.

Das Modell setzt sich dann zusammen aus:

$$Y = \beta_0 + \beta_1 * X + \varepsilon$$

bzw. im Falle der Energieverbrauchsanalyse mit Gradtagszahlen:

$$\text{Energieverbrauch} = \beta_0 + \beta_1 * \text{Gradtagszahl} + \varepsilon$$

Die beiden Parameter β_0 und β_1 definieren eine Gerade, die möglichst genau den gemessenen Zusammenhang zwischen *X* und *Y* darstellen soll. Optimal wäre das Modell, wenn die Werte der Ergebnisvariablen *Y* direkt auf der definierten Geraden liegen und damit dem prognostizierten Wert $\hat{Y} = \beta_0 + \beta_1 * X$ entsprechen würden.

Da dies generell nicht der Fall ist, wird ein Fehlerterm ε_i eingeführt. Dieser gibt die auftretende Abweichung zwischen den Daten und dem Modell an. Das ε_i fungiert als „catch-all" (James, 2013, p. 63) für die angenommene Vereinfachungen. So kann der Zusammenhang beispielsweise nicht vollständig linear sein, andere Einflussfaktoren von Bedeutung sein oder Messfehler unterlaufen sein. ε_i beschreibt in diesem Modell nicht berücksichtigte Faktoren (Grus, 2020, p. 181).

Folglich kann diese Abweichung, welche auch als Residuum bezeichnet wird, zur Charakterisierung des Modells herangezogen werden. Dazu wird die Summe über die quadrierten Residuen ε_i gebildet. Je geringer die Abweichungen, desto passender das Modell. Diesen Vorgang der Minimierung der Residuensumme bezeichnet man als Least Squares (kleinsten Quadrate). Die Minimierung erfolgt über die geeignete Wahl der Parameter β_0 und β_1 beispielsweise mit dem Gradient Descent Verfahren (Grus, 2020, p. 101).

Die Minimierungsaufgabe wie folgt formulieren:

$$\min_{\beta_0, \beta_1 \in \mathbb{R}} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 * X_i))^2 = \min_{\beta_0, \beta_1 \in \mathbb{R}} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 = \min_{\beta_0, \beta_1 \in \mathbb{R}} \sum_{i=1}^{n} {\varepsilon_i}^2$$

2.3.2 Das Bestimmtheitsmaß R^2

Als Varianz bezeichnet man die Abweichung einer Variablen von ihrem Mittelwert. Dieses Maß der Streuung kann dazu genutzt werden, die Güte des Modells zu bestimmen.

$$\text{Mittelwert } \bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Eine hohe Varianz in den Daten ist nicht notwendigerweise störend, es gilt nur zu ergründen, inwiefern die Varianz durch das vorliegende Modell beschrieben werden kann.

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = \frac{\text{erklärte Varianz}}{\text{gesamte Varianz}}$$

Dazu seien in den nachfolgenden Abbildungen Daten mit einer hohen und geringeren Streuung dargestellt. Das R^2 ist ein Maß, das angibt, inwiefern die vorliegende Varianz durch die gewählte Abhängigkeit beschrieben werden kann. Der R^2-Wert links ist deutlich geringer und kann so interpretiert werden, dass nur ein geringer Prozentsatz der vorhandenen Varianz in *Y* durch die Beziehung zwischen *Y* und *X* erklärt werden kann. Hingegen ist der Anteil auf der rechten Seite deutlich höher.

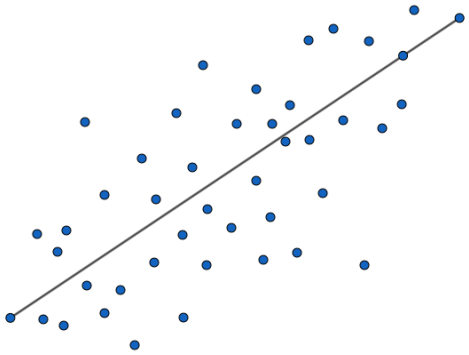

Abbildung 3 Niedriger R²-Wert

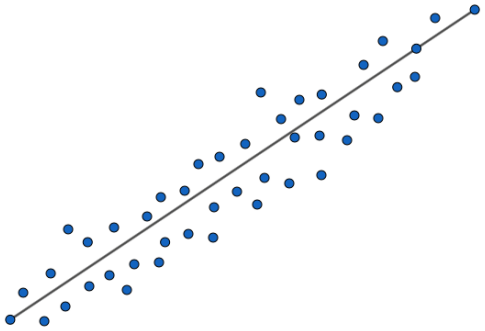

Abbildung 4 Hoher R²-Wert

Inwiefern ein R^2-Wert als gut oder unzureichend eingestuft werden kann, variiert je nach betrachteter Thematik ab. Wenn bekannt ist, dass der untersuchte Zusammenhang linear ist, so ist ein hoher R^2-Wert für das Modell zu erwarten. Sofern er dies nicht sein sollte, sollte der Ursprung der Daten überprüft werden und das zugrundeliegende Experiment auf Korrekt-

heit überprüft werden. Auf der anderen Seite ist in typischen Anwendungen in der Biologie, Psychologie, Marketing und anderen Feldern ein lineares Model höchstens eine grobe Approximation der Daten, da die Inhalte von weiteren, nicht berücksichtigten Faktoren beeinflusst werden (James, 2013, p. 86).

Abschließend ist festzuhalten, dass lineare Regression dazu geeignet ist, anhand von Daten ein Modell zu entwerfen, das die Realität ansatzweise widerspiegelt. Anhand dessen können Zusammenhänge aufgedeckt und veranschaulicht werden.

In der Praxis kann durch die Regressionsanalyse der relative Einfluss verschiedener unabhängigen Variablen auf abhängige Variablen aufgezeigt werden. Dies ermöglicht zum Beispiel in der Energieeffizienzanalyse nach ISO 50001 den Vergleich von Variablen (z. B. Temperatur, Energiesparmaßnahmen) und ihren Effekt auf die abhängige Variable (Energieverbrauch).

Limitationen sind dadurch gesetzt, dass für das Modell die Annahmen getroffen werden, dass die Zusammenhänge zwischen den Einflussfaktoren und dem Ergebnis additiv sind und das Modell linear ist. Somit wird angenommen, dass mehrere Einflussfaktoren sich untereinander nicht beeinflussen (James, 2013, p. 86). Auf diese Punkte kann durch Modifikation des Modells gezielt eingegangen werden, sofern der Sachverhalt es erfordert.

3 Anforderungsanalyse

3.1 Definition of Done

Die Definition of Done für das Gesamtprojekt ist es, ein Softwareprodukt bereitzustellen. Diese soll Anwendungsfälle aus dem Energiemanagement abdecken und für die Endnutzer einen Mehrwert schaffen. Die in dieser Arbeit spezifizierte, entworfene und prototypisch umgesetzte Backend-Architektur bietet beim Projektabschluss eine REST-API, mit der die spezifizierten Anwendungsfälle durchgeführt werden können. Das Frontend für die Energy-Data-Tools wird parallel entwickelt.

Ist das Software-Projekt ausgeliefert, können Endnutzer:

- Energieverbrauchsdatensätze hochladen und verwalten, formatieren und aggregieren
- Witterungseinflüsse mit verschiedenen Methoden bereinigen (Normalisierung)
- Regressionsmodelle erstellen und visualisieren

Das Software-Projekt ist erfolgreich abgeschlossen, wenn die geforderten Software-Features umgesetzt wurden und in der Produktionsumgebung laufen. Des Weiteren soll parallel zu dem hier entworfenen Backend ein neues Frontend umgesetzt werden, welche alte und neue Tools vereint. Der Projekterfolg ist außerdem abhängig von einer gelungenen und zukunfts-orientierten Übergabe für die Weiterentwicklung des Systems.

3.2 Vorgehen

Der Projektzeitraum endet voraussichtlich mit der Abgabe dieser Arbeit. Mit Abschluss des Projektes soll den Kunden eine lauffähige Software bereitgestellt werden und die Integration mit dem neuen Frontend soll erfolgreich durchgeführt worden sein.

Für die Analyse der Anforderungen, dem Softwareentwurf und der Realisierung werden agile Entwicklungsmethoden angewendet. Das gesamte Vorgehen ist Use Case orientiert, um überschaubare Arbeitspakete zu erhalten und die mehrwertschaffenden Funktionalitäten des Systems zu priorisieren.

In dem agilen Projektmanagement spielt das Product Backlog eine entscheidende Rolle für die Planung und Verteilung von Aufgaben. Laut Sommerville können Elemente in dem Backlog User Storys, Funktionsdefinitionen oder zusätzliche Tätigkeiten sein. (Sommerville, 2018) Das Backlog soll teamübergreifend gepflegt werden und einzelne Tasks sollen priorisiert und nach Use Cases sortiert werden.

Für dieses Projekt gehen Projekt-Manager, Frontend- und Backend-Entwickler Feature für Feature durch folgenden Prozess:

- Ermittlung der Anforderungen
- Entwurf des Frontends und Backends
- Umsetzung und Testen des Features
- Fehlerbehebung und Refactoring

3.3 Stakeholder

Das Projekt hat verschiedene Personen oder Entitäten mit Interesse an dem Erfolg, diese werden als Stakeholder bezeichnet. Es kann sich beispielsweise um Investoren, Projekt-Manager oder auch Endnutzer handeln. Die Interessen an einem Projekt können sehr unterschiedlich sein und auch die Definition von Erfolg unterscheidet sich.

Im Nachfolgenden werden die Stakeholder mit ihren unterschiedlichen Erfolgsdefinitionen dargestellt. Die Analyse der verschiedenen Interessensgruppen dient dazu, Anforderungen an das Projekt passgenau auf die verschiedenen Erfolgsdefinitionen auszurichten. Diese Analyse ermöglicht es Risiken oder mögliche Konfliktpotentiale zwischen Interessensparteien zu erkennen.

Stakeholder	*Energy Data GmbH*
Erfolgsdefinition	1. Wirtschaftlichkeit des Projektes 2. Vergrößerung des Kundenstammes 3. Erweiterung des Produktsortiments
Stakeholder	*Projektmanager*
Erfolgsdefinition	1. Auslieferung des Software-Produktes 2. Einhaltung der Projektziele 3. Wissenszuwachs für zukünftige Projekte 4. Erkennen und überwinden von Projektherausforderungen 5. Verbesserung der Arbeitsweise im Team
Stakeholder	*Entwickler*
Erfolgsdefinition	1. Auslieferung des Software-Produktes 2. Entwicklung von funktionsfähiger Software, welche alle Anforderungen erfüllt 3. Bereitstellung, Betrieb und Wartung des Systems 4. Wissenszuwachs für zukünftige Projekte
Stakeholder	*Endnutzer*
Erfolgsdefinition	1. Produkt kann in bestehenden Arbeitsprozesse eingebunden werden 2. Benutzung ist intuitiv und flexibel 3. Das Produkts hat einen Mehrwert für ihre eigenen Arbeitprozesse 4. Das Produkt ist zuverlässig einsetzbar 5. Das System kann in bestehende Energiemanagementsysteme eingebunden werden

Tabelle 1 Stakeholder

3.4 Konventionen

3.4.1 Coding-Style

Für eine hohe Programmcode-Qualität soll der PEP 8 – Styleguide for Python Code (Rossum, 2022) systemweit eingehalten werden. Die konsistente Einhaltung des Styleguides soll zu einer guten Leserlichkeit und Wartbarkeit des Programmcodes führen.

3.4.2 Versions-Management

Für das Versions-Management soll für das Backend ein Git-Repository genutzt werden. Mit automatisierten GitHub-Actions wird eine neue Softwareversion Tests durchlaufen, welche die Korrektheit des Programms vor der Bereitstellung überprüft.

3.5 Risiken

Ein Software-Projekt ist immer von Risiken begleitet, welche einen erfolgreichen Projektabschluss gefährden können. Es ist daher im Interesse der Stakeholder, diese zu mitigieren. Hierfür ist es nötig in der Planungsphase mögliche Risiken zu identifizieren, diese zu definieren und Maßnahmen festzulegen, mit denen den Risiken im Vorfeld entgegengewirkt werden kann.

Ein erkennbares Risiko entsteht durch die Übergabe des Projektes an nachfolgende Entwickler. Es handelt sich hierbei um ein personenbezogenes Risiko (Sommerville, 2018, p. 721).

Da dieses Projekt zeitlich begrenzt ist, sollten im Vorfeld Maßnahmen für eine zukunftsorientierte Übergabe erfolgen. Für zukünftige Entwickler soll die Einarbeitung erleichtert und das Verständnis für die Softwaresysteme gefördert werden. Die Stakeholder erhalten dadurch ein Produkt, welches nicht an den Entwickler gebunden ist, sondern auch ohne große Komplikationen oder hohe Kosten erweitert werden kann.

Zu den festgelegten Maßnahmen zur Mitigation dieses Risikos zählen:

- Dokumentation aller geplanten und entwickelten Software-Komponenten, sowie dem Systemkontext. Auch unfertige Komponenten müssen dokumentiert werden.

- Dokumentation der lokalen Entwicklungsumgebung und Anleitungen zum lokalen Deployment sowie dem Testen der Software
- Dokumentation und Anleitungen zum Deployment in der Produktionsumgebung
- Kommentierung des Programmcodes
- Dokumentation der REST-API und Einhaltung von REST-Standards

3.6 Zielgruppen

In 3.3 wurden die Stakeholder von diesem Projekt definiert, wobei die Endnutzer als eine Interessegruppe aufgelistet sind. Die Zielgruppen dieser Software sind jedoch vielseitig und werden daher kleinteiliger in Persona unterteilt, um eine genauere Analyse ihrer Anforderung an die Software zu ermöglichen. Persona stellen fiktive Nutzer der Software dar und ihre individuellen Anforderungen. Das von Allan Cooper vorgeschlagene Konzept ermöglicht, in der Anforderungsanalyse verschiedene Nutzungsschwerpunkte der Software zu erkennen und diese zielorientiert in der Entwurfsphase zu konkretisieren (Allan Cooper, 2007).

P.1 Persona – Energiemanager

Energiemanager nutzen die EnergyData-Tools hauptsächlich zur Verwaltung, Verarbeitung und Analyse von Energieverbrauchsdatensätze. Ein möglicher Anwendungsfall ist beispielsweise die datenfundierte Bewertung von Energieeffizienmaßnahmen.

- Datenoperation
- Datensatzverwaltung
- Datenanalyse

P.2 Persona - Energieberater

Ein Energieberater berät Unternehmen zu Themen wie Energieeffizienz. Hierfür möchte ein Energieberater die Datenbanalyse nutzen. Die Visualisierung der Daten zur Präsentation der Ergebnisse ist für diese Persona besonders wichtige Anforderung.

- Datenanalyse
- Datenvisualisierung

P.3 Persona – API Nutzer

Die Persona „API Nutzer“ ist ein Experte der Softwareentwicklung, welche die EnergyData-Tools mittels API in ihre eigene Software einbinden wollen. Sie erwarten eine technische Dokumentation zur Nutzung der API sowie die Einhaltung von REST - Standards. Die Front-end-Entwickler der Energy Data GmbH gehören ebenfalls zu dieser Zielgruppe.

- Einbindung der API in Software-Systeme
- Benötigt konsistente, technische Dokumentation
- Einhaltung von REST-Standards

3.7 Systemspezifikation

3.7.1 Ist-Zustand

Das in dieser Arbeit spezifizierte und entworfene System soll als Backend-System eingesetzt werden. Das bereits bestehende Frontend wird parallel neu entwickelt und in dieser Arbeit nicht vorgestellt. In der folgenden Abbildung ist der Ist-Zustand der bestehenden Infrastruktur grob zur Übersicht dargestellt. Die Systeme sind als Blackbox dargestellt und zeigen keine weiteren Details über ihre Komplexität oder wie die Systeme bereitgestellt werden.

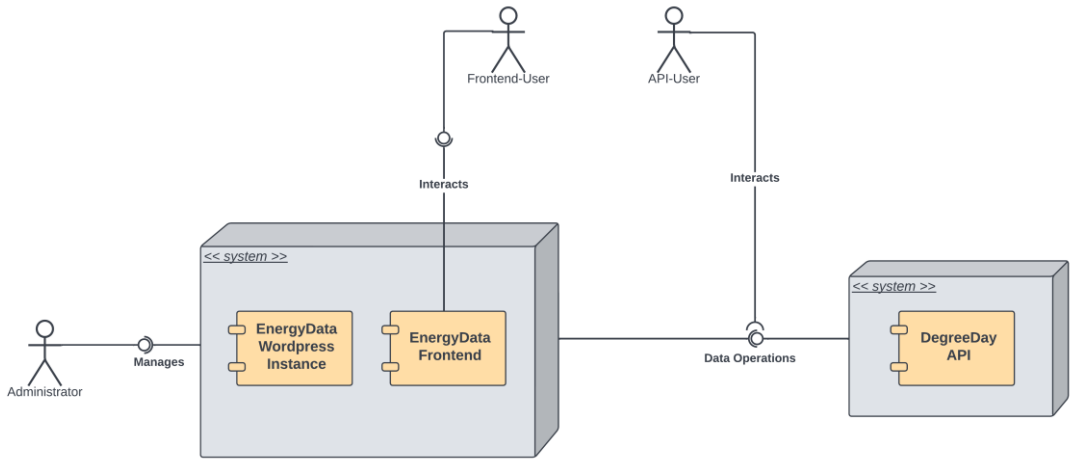

Abbildung 5 Ist-Zustand Infrastruktur

3.7.2 REST-API

Das Backend soll eine REST-API anbieten. Diese Schnittstellen werden vom Frontend für die Abbildung der Use Cases genutzt. Die REST-API ermöglicht auch technischen Nutzern, das Backend-System in bereits bestehende Energiemanagementsysteme einzubinden (Siehe P.3 Persona – API Nutzer). Jede Use Case Komponente hat eine View Subkomponenten, welche die REST-API implementiert. Bei dem Entwurf der Schnittstellen sind Design-Guidelines befolgt worden, welche in NFA.2 REST API Standards spezifiziert sind.

3.7.3 Authentifizierung

User

Die User sind registrierte Benutzer, welche in der WordPress-Instanz gespeichert werden. Das hier beschriebene Backend hat keinen Einfluss auf die Rollenvergabe. Die WordPress-Instanz sorgt für einen autorisierten Zugriff und gibt bei Aufrufen die benötigten Informationen zur Autorisation mit. Dies ermöglicht im Backend eine Speicherung und Abrufbarkeit von userspezifischen Daten. Das Backend authentifiziert die User mittels einer User ID, welche von der Wordpress-Instanz geliefert wird.

Benutzerrollen

Das Backend-System verarbeitet die Benutzerrollen, welche bei den Aufrufen mitgeben werden. Es gibt zwei Rollen, zwischen denen unterschieden wird:

- Registrierter User
- Subscription User

API Key

Der API Key wird von der Degreeday-API für die Autorisierung genutzt. Ein registrierter User kann nur einen beschränkten Zeitraum an historischen Wetterdaten abfragen und ist daher in seiner Nutzung eingeschränkt. Bei einem Backend-Aufruf kann ein API Key für die Autorisierung mitgegeben werden. Die Rollen der Nutzer haben ebenfalls dedizierte API

Keys mit unterschiedlichen Berechtigungen. Der API Key ist eine sensible Information, welche nicht vom Backend nach außen gelangen darf.

3.7.4 Beschreibung der Messdaten

Messungen von Verbräuchen sind nicht standardisiert und die Anforderungen von den Zielgruppen an die Energieverbrauchsdaten können sich stark unterscheiden.

Das Format, die Speicherung, die Messintervalle und auch die Qualität und Vollständigkeit der Energieverbrauchsdatensätze können sich unterscheiden. Alle Varianten haben gemeinsam, dass Verbrauchsmessungen auf Zeiträume abgebildet werden.

Um die Anforderungen an die Energy Data-Tools zu spezifizieren gilt es vorerst, die Ausprägungen von den zu verarbeitenden Daten zu verstehen.

Die Messdaten haben zusammengefasst folgende Besonderheiten und Herausforderungen:

- CSV-Separatoren sind nicht standardisiert und stehen in Abhängigkeit zum System des Endnutzers und der Messanlagen, welche die Messdaten speichern
- Der Datensatz kann fehlerhafte Felder aufweisen
- Der Datensatz kann lückenhafte Messungen aufweisen
- Datumsformate können sich unterscheiden
- Die Messintervalle können verschiedenen zeitliche Abstände und Unregelmäßigkeiten aufweisen

Externes Datenformat – CSV Verarbeitung

Ein häufig für die Speicherung von Energieverbrauchsdatensätzen verwendetes Format ist CSV (Comma Separated Values). Die externen Systeme für die Speicherung trennen die einzelnen Messungen mit unterschiedlichen Separatoren. Daher haben verschiedene Nutzer andere Anforderungen an die Verarbeitung von Datensätzen.

Die EnergyData-Tools sollen daher dem Endnutzer eine große Flexibilität bieten und verschiedenste CSV-Ausprägungen unterstützen.

Messintervalle

Die zeitlichen Abstände, in denen die Messungen gespeichert werden, können sehr unterschiedlich, unregelmäßig oder lückenhaft sein. Energieverbrauchsmessungen können beispielsweise in minütlichen, stündlichen, täglichen, wöchentlichen oder monatliche Messungen gespeichert werden. Es können auch unregelmäßige, manuelle Messungen vorgenommen werden. Die Zeitintervalle haben großen Einfluss auf die Möglichkeiten zur Weiterverarbeitung der Daten, sowie die Qualität und Genauigkeit von Verbrauchsanalysen.

Lückenhafte Messungen

Durch manuelle Messungen, Datenverlust und weiteren Ereignisse können Messdaten Lücken aufweisen. Für diese Zeiträume ist kein genauer Verbrauch bekannt. Es muss davon ausgegangen werden, dass bei der Verarbeitung der Energiedaten durch bestimmte Methoden, Probleme bei lückenhaften Datensätzen auftreten werden. Daher gilt für den Entwurf von den Verarbeitungsmethoden diese Sonderfälle zu beachten.

Internes Messdatenformat

Das interne Format von Energieverbrauchsmessungen soll im gesamten System verwendet werden, um Messzeiträume systemweit standardisiert abzubilden.

Messzeitraum Beginn	***Messzeitraum Ende***	***Energieverbrauch***
Datum	Datum	Verbrauch (z.B in kWh)

Tabelle 2 Internes Messdatenformat

3.7.5 Use Cases

Use Case 1 – Hochladen von Energieverbrauchsdaten

Ziel	Hochladen von Datensätzen, bzw. Erstellung von Rohdatensätzen
Akteur	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer möchte einen Energieverbrauchsdatensatz im CSV-Format hochladen .

Erfolgsszenarien

1. Der Rohdatensatz wurde validiert, formatiert und persistiert und damit für den Endnutzer für weitere Datenoperationen abrufbar gemacht.

Alternativszenarien

1. Der Upload wurde mit Fehlermeldung zurückgewiesen.

Use Case 2 – Formatierung von Rohdatensätzen

Ziel	Der Endnutzer soll beim Hochladen (Use Case 1 – Hochladen von Energieverbrauchsdaten) eine hohe Flexibilität an verschiedenen Formaten benutzen können. Verschiedene Formate werden in das interne Format überführen (Siehe Internes Messdatenformat, Abschnitt Internes Format).
Akteur	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Energieverbrauchsdaten sind nicht standardisiert gemessen und gespeichert werden und können in verschiedenen Formaten vorliegen.

Erfolgsszenarien

1. Die CSV-Datei ist valide und konnte eingelesen werden.
2. Der Rohdatensatz wurde auf seine korrekte Formatierung überprüft.
3. Der Rohdatensatz wurde in das interne Format überführt.
4. Es wurden Metainformationen über die Beschaffenheit der Daten extrahiert.

Alternativszenarien

1. Die Formatierung wurde mit Fehlermeldung zurückgewiesen.
    a. enthält keine valide CSV – Datei, z. B. nicht unterstützt CSV-Separatoren.
    b. Ist über dem Dateigrößenlimit.
    c. Rohdaten in falschem Format, z. B. unerlaubte Reihenfolge der Spalten, falsche Datumsformatierung.
1. Endnutzer hat Metainformationen über den Datensatz eingesehen.

Use Case 3 – Herunterladen von Datensätzen in CSV-Format

Ziel	Herunterladen von Datensätzen in CSV-Format
Akteur	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer lädt einen Datensatz im CSV-Format herunter.

Erfolgsszenarien

1. Der Datensatz kann aus dem internen Format in eine Datei im CSV-Format umgewandelt und vom Endnutzer heruntergeladen werden.
2. Informationen über den Datensatz werden, für eine bessere Identifikation und Präsentation der Daten, in den Kopf der CSV-Datei geschrieben.

Use Case 4 – Aggregation zeitlicher Intervalle

Ziel	Die Messintervalle von Datensätzen sollen zusammengefasst und die zeitlichen Intervalle der Messdaten sollen vereinheitlicht werden, um die Daten für weitere Operationen vorzubereiten.
Akteure	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer hat unregelmäßige oder hochfrequentierte Messungen (z. B. 15-minütlich) und möchte den Datensatz in einem anderen Zeitintervall (Täglich, Monatlich, Jährlich) aggregieren.

Erfolgsszenarien

1. Minütliche Messungen werden zu einem Tag zusammengefasst.
2. Tägliche Messungen werden in monatliche Messintervalle zusammengefasst.
    a. Schaltjahre wurden beachtet.
3. Tägliche Messungen werden in jährliche Messintervalle zusammengefasst
4. Für alle Fälle gilt, dass der Energieverbrauch der aggregierten Zeiträume aufsummiert wurde.

Use Case 5 – Normalisierung (Alle Methoden)

Ziel	Die Bereinigung der Verbrauchswerte von Witterungseinflüssen mittels Normalisierungsmethoden (LTA, BYA, ER)
Akteur	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer möchte einen Datensatz mit Energieverbrauchsmessungen normalisieren, d. h. von Wettereinflüssen bereinigen (Siehe Bereinigung der Energieverbrauchsdaten (Normalisierung)).

Vorbedingungen

- Die Messungen im Datensatz sind in der Datenbank gespeichert.
- Die Degreeday-API kann die benötigten Daten bereitstellen.
- Für LTA: Der User muss einen API Key mit einer Berechtigung für lange Zeiträume haben (Subscription User).
- Für BYA: Der Datensatz muss mindestens ein gesamtes Jahr enthalten, welches als Basisjahr genutzt werden kann.

Erfolgsszenarien

1. Für jede ausgewählte Normalisierungsmethode werden die benötigten Daten von der Degreeday-API abgefragt und auf den Rohdatensatz angewendet. Es wird ein neuer, normalisierter Datensatz angelegt und abrufbar gemacht. Dieser soll dem Rohdatensatz zugehörig sein.

2. Die Operation LTA wird auf jedes Messintervall ausgeführt. Es konnten die benötigten historischen Daten von der Degreeday-API aufgerufen werden.
3. Das Basisjahr, also das frühste ganze Jahr im Datensatz, wird erkannt, die benötigen Degreeday-Werte für das Basisjahr werden von der Degreeday-API angefragt und genutzt, und für jedes Messintervall wird der Verbrauchswert in das Verhältnis zum Verbrauch vom Basisjahr gesetzt.
4. Die Methode ER (Efficiency Ratio) wird erfolgreich auf den Datensatz angewendet.

Alternativszenarien

1. Hat der Endnutzer keine Berechtigung für Anfragen länger als 12 Monate, kann die Operation LTA nicht ausgeführt werden.
2. Enthält der Datensatz kein ganzes Jahr kann die BYA Methode nicht ausgeführt werden.

Use Case 6 – Lineare Regressionsanalyse

Ziel	Eine oder mehrere lineare Regressionsanalyse über einen Energieverbrauchsdatensatz durchführen und Qualitätsindikatoren ermitteln.
Akteur	P.1 Persona – Energiemanager, P.2 Persona - Energieberater, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer möchte einen bestehenden Datensatz mit Energieverbrauchsmessungen mittels linearer Regression analysieren, um Verbräuche vergleichbar zu machen und Vorhersagen zu treffen. Die Qualität des Regressionsmodells soll mittels Qualitätsindikatoren bestimmt werden. Der User möchte mit Parametern Einfluss auf den Trainiervorgang der Modelle nehmen.

Vorbedingungen

- Die Messungen im Datensatz sind in der Datenbank gespeichert.
- Die Degreeday-API stellt die benötigten Daten bereit.

Erfolgsszenarien

1. Es kann ein gültiges Regressionmodell für den Datensatz erzeugt werden, persistiert und abrufbar gemacht werden. Es können Qualitätsindikatoren ermittelt und abrufbar gemacht werden.

Use Case 7 – Iterative Lineare Regressionsanalyse

Ziel	Es sollen iterativ eine Vielzahl von Regressionsmodellen erzeugt werden mit auf- und absteigenden Basistemperaturen. Das Regressionsmodell mit den idealen Qualitätsindikatoren soll herausgehoben werden.
Akteur	P.1 Persona – Energiemanager, P.2 Persona - Energieberater, P.3 Persona – API Nutzer
Auslöser	Analog zu Use Case 6 – Lineare Regressionsanalyse. Es sollen ideal Werte für die Basistemperaturen gefunden werden, in dem mit auf- und absteigenden Basistemperaturen mehrere Regressionsmodelle erzeugt werden.

Vorbedingungen

- Es wird auf die Implementierung des Use Case 6 – Lineare Regressionsanalyse zurückgegriffen. Die Vorbedingungen sind Analog zu Use Case 7.

Erfolgsszenarien

1. Es können mehrere Regressionsmodelle erzeugt werden mit auf- und absteigenden Basistemperaturen.

Risiken	Die Schrittgröße für die Basistemperaturen ist ausschlaggebend für die Laufzeit, die Effizienz der Operationen soll daher bei der Implementierung besonders berücksichtigt werden. Ein erkennbarer „Bottleneck“ ist hier die mehrfache Verwendung der Degreeday-API, da mit jedem Schritt eine andere Basistemperatur für die Gradtagszahlen benötigt wird.

Use Case 8 – Abruf von Regressionsmodellen

Ziel	Die Projektinformationen, das lineare Modell, die Metriken, die Parameter und die einzelnen Dateneinträge von angelegten Regressionsmodellen sind abrufbar.
Akteur	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer möchte auf die Ergebnisse der Regressionsanalyse zugreifen und zugehörige Entitäten abrufen.

Erfolgsszenarien

1. Der Endnutzer hat die angefragten Daten erhalten.

Use Case 9 – Abruf von Datensätzen

Ziel	Die Projektinformationen, Metainformationen und die einzelnen Dateneinträge von einem Datensatz sind abrufbar.
Akteur	P.1 Persona – Energiemanager, P.3 Persona – API Nutzer
Auslöser	Ein Endnutzer möchte auf die Ergebnisse zugreifen und ihm zugehörige Entitäten abrufen.

Erfolgsszenarien

1. Der Endnutzer hat Projektinformationen zu einem Datensatz erhalten.
2. Der Endnutzer hat Metainformationen zu einem Datensatz erhalten.
3. Der Endnutzer hat den Inhalt von einem Datensatz erhalten.
4. Der API Key darf nicht ausgegeben werden.

Use Case 10 – Verwalten von Datensätzen

Ziel	Verwaltung von Datensätzen: Umbenennen, Löschen, Favorisieren und verschiedene Filterungsoperationen
Akteur	P.1 Persona – Energiemanager

Auslöser Ein Endnutzer möchte die Ergebnisse seiner Operationen verwalten.

Erfolgsszenarien

1. Der Endnutzer hat seinen Datensätze umbenannt.
2. Der Endnutzer hat seinen Datensätze favorisiert.
3. Der Endnutzer hat seinen Datensätze gelöscht.
4. Der Endnutzer hat seine Datensätze gefiltert erhalten.

3.7.6 Fachliches Datenmodell

In der nachfolgenden Abbildung des fachlichen Datenmodells werden die Entitäten und Datentypen des Softwareentwurfs dargestellt. Es zeigt die Relationen der Entitäten und deren Kardinalitäten. Das fachliche Datenmodell ermöglicht die Entitäten und ihren Kontext zu verstehen. Zur eindeutigen Identifizierung mit geringer Wahrscheinlichkeit für Kollision sollen UUID4 für die wichtigsten Entitäten genutzt werden. Es sind weitere Vorgaben für die Datentypen und Größen der Daten enthalten. Wenn für eine Entität keine Zeile mit ID abgebildet ist, dann werden einfache inkrementierte IDs genutzt.

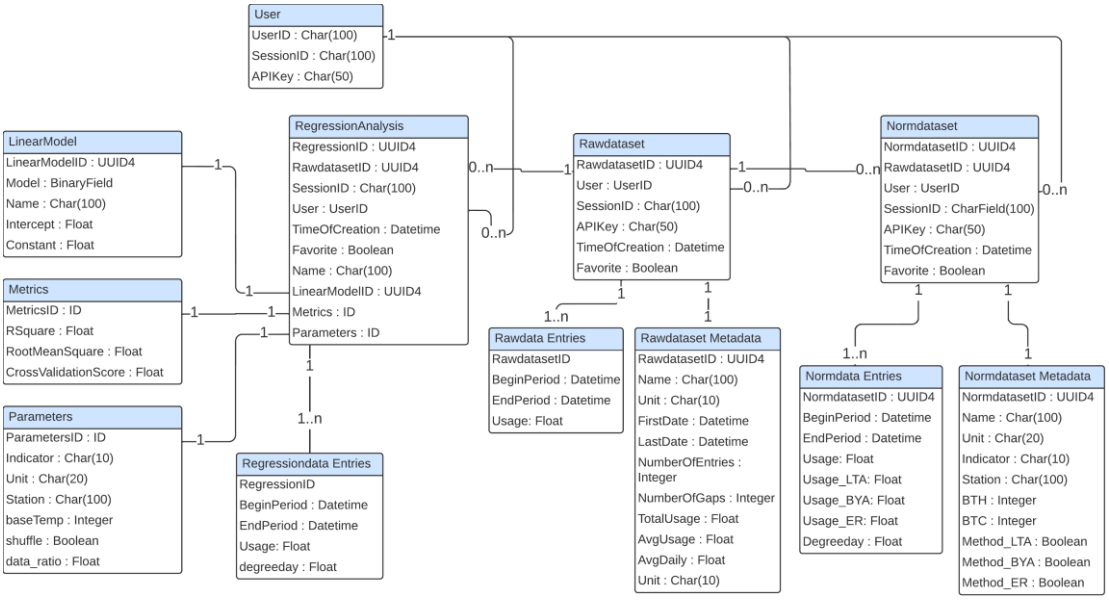

Abbildung 6 Fachliches Datenmodell

Fachliches Datenmodell - User

Die User-Entität steht mit den Entitäten „Rawdataset“, „Normdataset“ sowie der „RegressionAnalysis“ in Beziehung, da alle Entitäten dem User zugehörig sind. Dies ermöglicht die Datensätze und Modelle anhand der User ID zu verwalten. Zusätzlich werden die letzte Session ID, sowie der zugehörige API Key des Users gehalten.

Fachliches Datenmodell - Rawdataset

Die „Rawdataset“ Entität ist ein Rohdatensatz, welcher in das interne Format überführt wurde. Die Entität repräsentiert die projektartigen Informationen über den Energieverbrauchsdatensatz. Dieser Datensatz bildet die Grundlage für Datenoperationen wie die Normalisierung oder Aggregation in andere zeitliche Intervalle oder die Erstellung von Regressionsmodellen.

Die Rawdataset-Entität hält Projektinformationen über:

- TimeOfCreation - Erstellungszeitpunkt
- Favorite - Favorisierter Datensatz
- APIKey – Genutzter API Key

Hierfür haben das Rawdataset sowie das Normdataset jeweils eine Subentität für ihre Metadaten (Fachliches Datenmodell – Rawdataset-Metadata) und mindestens einen, bis n-viele Datensatzeinträge. (Fachliches Datenmodell – Rawdataset-Entry)

Fachliches Datenmodell – Rawdataset-Entry

Eine „Rawdataset-Entry“-Entität ist genau ein Messzeitraum in einem Energieverbrauchsdatensatz. Durch die Entkapselung der einzelnen Messungen in eigene Entitäten können die einzelnen Einträge in einer eigenen Tabelle persistiert werden.

- BeginPeriod – Anfang des Messzeitraums
- EndPeriod – Ende des Messzeitraums
- Usage – Energieverbrauchswert für Messzeitraum

Fachliches Datenmodell – Rawdataset-Metadata

Die Rawdataset Metadata Entität enthält Informationen und Analysen über den Messdatensatz:

- Total Usage – Summe aller Messungen, Gesamtverbrauch
- Average Usage – Durchschnittsverbrauch
- Average Daily – Durchschnittlicher Verbrauch von Tageswerten
- NumberOfEntries – Anzahl der Messungen
- NumberOfGaps – Anzahl Lücken in den Messzeiträumen

Fachliches Datenmodell – Normdataset

Das Normdataset ist der Projekt-Eintrag für einen normalisierten Datensatz. Er hält Daten analog zum Fachliches Datenmodell - Rawdataset, also Informationen zur Verwaltung des normalisierten Datensatzes. Da die Normalisierung den Datensatz grundlegend verändert, werden normalisierte Datensätze als eigene Entität spezifiziert.

Fachliches Datenmodell – Normdataset-Entry

Jeder Eintrag eines normalisierten Datensatzes hält, zusätzlich zum unbereinigten Verbrauchswert, die Ergebnisse der Normalisierungsmethoden. Diese neu ermittelten Verbräuche sind in den Spalten nach ihren Methoden benannt.

- Usage – Verbrauch
- LTA_Usage – normalisierter Verbrauch
- BYA_Usage - normalisierter Verbrauch
- ER_Usage - normalisierter Verbrauch

Des Weiteren wird in jedem Messzeitraum auch die zugehörige Gradtagszahl gehalten, welcher Grundlage für die Berechnungen in den Normalisierungsmethoden ist und von der Degreeday-API abgefragt wird.

- Degreeday - Gradtagzahl

Fachliches Datenmodell – Normdataset-Metadata

Die Speicherung der Metadaten eines normalisierten Datensatzes unterscheidet sich insofern von einem Rohdatensatz, als dass die angewendeten Methoden und die dazugehörigen Parameter gehalten werden. Daher persistiert Normdataset-Metadata Entität die Parameter, welche für die Normalisierung genutzt wurden.

- Unit – Einheit des Energieverbrauchs
- Station - Wetterstation
- Indicator – Heating- oder Coolingdegreedays, oder beide
- BTC (Base Temperature Cooling) – Basis Temperatur Kühlung
- BTH (Base Temperature Heating) – Basis Temperatur Heizung

Zusätzlich wird gekennzeichnet, welche Methoden für die Normalisierung genutzt wurden.

- Method_LTA - Langzeitdurchschnitt – Witterungsbereinigung
- Method_BYA - Basisjahr
- Method_ER - Effizienz-Kennzahl

Fachliches Datenmodell – Regression Analysis

Die Projektinformationen von Regressionsanalysen werden in der Entität „RegressionsAnalysis“ persistiert und können mit den gehaltenen Informationen verwaltet werden. Die RegressionAnalysis-Entität hat Subentitäten, welche genauere Informationen zu dem Linearen Modell persistieren. Die Fremdschlüssel sind hochgezählte IDs mit 1–1 Beziehungen zu folgenden Subentitäten:

- LinearModel – Das serialisierte Regressionsmodell
- Metrics - Metriken bzw. Qualitätsindikatoren des linearen Modells
- Parameters - Parameter für das Trainieren des Modells

Die Regressions Analysis Entität ist einem User zugeordnet. Das Projekt kann favorisiert werden. Die Session, der Erstellungszeitpunkt sowie ein Name werden bei dem Anlegen des Projektes gespeichert.

Diese Informationen sind für die Verwaltung des Regressions-Projektes nötig:

- Name – Projektname, Bezeichnung des Modells
- TimeOfCreation – Erstellungszeitpunkt
- SessionID – ID der Session bei der Erstellung
- UserID – Identifiziert den User, dem das Projekt zugehörig ist

Fachliches Datenmodell – LinearModel

Das „LinearModel" persistiert das serialisierte Objekt des Regressionsmodells als Binary-Field. Durch die Serialisierung des Laufzeitobjektes als BinaryField kann der Zustand des linearen Modells exakt persistiert werden und das Laufzeitobjekt bei Bedarf wiederhergestellt werden.

- Model – Das serialisierte Laufzeitobjekt des Modells

Die Entität enthält zusätzlich die Informationen, welche das lineare Modell beschreiben (Siehe Modellbildung mittels Least Squares).

- Constant –Steigung des linearen Modells
- Intercept – Schnittpunkt mit Y-Achse

Fachliches Datenmodell – Metrics

Jedes Regressions Projekt persistiert in einer „Metrics-Entität" die Indikatoren für Güte des erzeugten Regressionmodells.

- RSquare - Das Bestimmtheitsmaß $\boldsymbol{R^2}$
- RootMeanSquare – Das quadratisches Mittel
- CrossValidationScore – R^2 ermittelt durch Cross Validation

Fachliches Datenmodell – Parameter

Die „Parameters-Entität" persistiert die verwendeten Parameter, welche für die Erzeugung des Regressionsmodells genutzt werden:

- Unit – Einheit des Energieverbrauchs

- Station - Wetterstation
- Indicator – Heating- oder Cooling Degreedays
- Shuffle – Shuffle der Trainings- und Testdaten
- Data_Ratio – Verhältnis von Trainings- und Testdaten

3.7.7 Nicht-funktionale Anforderungen

NFA.1 Datenzentriertheit

Die Verarbeitung und Analyse von Energieverbrauchsdaten stellt die zentrale Funktionalität der EnergyData-Tools dar und ist mehrwertschaffend für das Endprodukt. Diese Datenzentriertheit ist ausschlaggebend für die Auswahl von Technologien und Programmierumgebungen. Die Berücksichtigung dieser Randbedingung bedeutet, dass die Technologien auf performante und umfangreiche Datenverarbeitung ausgelegt sein sollen.

NFA.2 REST API Standards

Bei dem Entwurf der Schnittstellen sollen Design-Guidelines befolgt werden, welche eine gute REST API ausmachen. Die REST API stellt die Schnittstelle der Software nach außen dar und muss daher verschiedene Qualitätsanforderungen erfüllen, um den Nutzen für die User dieser Software zu maximieren (Bloch, 2006).

- Abbildung der Use Cases
    - Die Funktionalität einer REST-Schnittstelle soll durch die Benennung der Schnittstellen verständlich gemacht werden
    - Es dürfen keine Implementierungsdetails nach außen sickern. Dies bedeutet in der Praxis, dass der Benutzer der API nicht wissen muss, wie das Backend genau arbeitet
- Verwendung von JSON
    - Jede POST-Operation soll die Parameter als JSON erwarten
    - Jede Operation soll die Antworten als JSON zurückgeben
- Umgang mit Fehlern

- Auftretende Fehler im Backend sollen schnellstmöglich und mit menschenleserlichen Detailnachrichten zurückgegeben werden
- Die http Antwortcodes sollen genutzt werden (Positiv, wie Negativ)

Weitergehend soll bei Entwurf und Implementierung der REST-API die Best-Practices aus (Fredrich, 2013) befolgt werden.

NFA.3 Performance

Die Performance ist für datenzentrierte Software eine wichtige Anforderung. Es werden unter gewissen Umständen große Datenmengen verarbeitet. Die angenehme Nutzbarkeit der Software soll für den Endnutzer gewährleistet sein. Daher gilt es für den nachfolgenden Entwurf mögliche Bottlenecks zu identifizieren sowie während der Realisierung eine performante Implementierung zu berücksichtigen.

NFA.4 Dateigrößenlimit

Als nicht-funktionale Anforderung wird eine Maximalgröße für zu verarbeitende Datensätze festgelegt, um zu verhindern, dass das System zu hohe Antwortzeiten hat. Diese Größe gilt als Richtwert und kann in der Entwicklungsphase durch manuelles Testen angepasst werden.

Als Dateigrößenlimit soll vorerst eine Grenze von 10 Megabyte gelten. Die Größe entstand aus dem Erzeugen von möglichen Testdatensätzen mit minütlichen Messungen über mehrere Jahrzehnte.

NFA.5 Erweiterbarkeit & Wiederverwendbarkeit

Bereits implementierte Komponenten haben eine höhere Zuverlässigkeit als neu entworfene Komponenten. Dies resultiert daraus, dass diese bereits in der Praxis getestet wurden. Schwachstellen, die im Entwurf entstanden sind, wurden in der Implementierungsphase beseitigt. Des Weiteren besteht bei der Wiederverwendung von Komponenten ein geringes Risiko in Bezug auf die Kosten (Sommerville, 2018, pp. 492-494).

Insbesondere die Implementierungsaspekte für die Wiederverwendbarkeit von Software-Komponenten sollen folglich im Entwurf beachtet werden.

NFA.6 Resilienz

Die Resilienz des Systems kann aus verschiedenen Perspektiven betrachtet werden. Für diesen Entwurf ist es ausreichend das System in Bezug auf seine Fehlertoleranz und die Ausfallsicherheit zu betrachten. Diese Eigenschaften sollen bereits beim Systementwurf berücksichtigt werden.

- Das System soll tolerant mit fehlerhaften Benutzereingaben umgehen
- Das System soll dem Nutzer transparente Fehlermeldungen zurückgeben, um die potentielle Folgefehler von Endnutzern zu vermeiden
- Die Wartung des Systemsbetriebs soll umfangreich dokumentiert sein

NFA.7 Sicherheit

Die Sicherheit des Systems vor Angreifern soll ebenfalls im Systementwurf berücksichtigt werden. Hierbei soll ein realistisches Kosten-Nutzen Verhältnis beachtet werden.

- Das System sollte ausreichend gegen Cyberangriffe geschützt sein
- Die Kommunikation mit dem System soll verschlüsselt sein

4 Softwareentwurf

4.1 Architekturentscheidungen

4.1.1 Entwurfsentscheidungen

Laut Ian Sommerville ist der Architekturentwurf ein kreativer Prozess, der von der Art des Systems, der Herkunft und Erfahrung des Systemarchitekten und den spezifischen Systemanforderungen abhängt. Daher betrachtet er den Entwurfsprozess als eine Reihe von zu treffenden Entscheidungen. (Sommerville, 2018)

Um die Entscheidungsprozesse nachvollziehbar zu machen, werden in dem folgenden Abschnitt Entscheidungen zum Systementwurf dokumentiert. Von der Programmiersprache, über die Auswahl der Technologien bis hin zu dem Entwurf der Systemarchitektur und Vorgaben für die Implementierung. Entscheidungen sollen die Anforderungen aus der Anforderungsanalyse erfüllen.

E.1 Entwurfsentscheidung Programmiersprache

Die EnergyData-Tools sind vorwiegend eine Sammlung von datenzentrierte Tools (NFA.1 Datenzentriertheit) Die Verarbeitung und Analyse von Daten steht im Vordergrund und bietet den direkten Mehrwert für die Endnutzer der Tools. Daraus lässt sich für den Entwurf schließen, dass die Auswahl der Architektur und Technologien vorrangig diese Anforderung erfüllen soll. Für diesen Zweck ist es nützlich, wenn eine Programmiersprache ein Ökosystem von Bibliotheken und Frameworks hat, welche für diese Anforderung genutzt werden können.

Python erfüllt diese Anforderungen, insbesondere durch eine Vielzahl von Bibliotheken aus dem Data Science und dem Machine Learning. Daher fällt die Wahl der Programmiersprache auf Python.

E.2 Entwurfsentscheidung Bibliotheken

Zur Datenverarbeitung soll die umfangreiche Data-Science Python-Bibliothek „Pandas“ zum Einsatz kommen. Sie bietet verschiedene Datenoperationen und umfangreiche Hilfsmittel zur Verarbeitung und Analyse von großen Datenmengen.

Zur statistischen Analyse und Machine Learning Funktionalitäten der EnergyData-Tools soll die Bibliothek „scikit-learn“ zum Einsatz kommen. Sie bietet umfangreiche Mittel zum Trainieren von linearen Modellen und erfüllt alle Anforderungen, welche durch die Regressionsanalyse an die Software gestellt werden.

E.3 Entwurfsentscheidung Framework

Das High-Level Framework „Django“ bietet ein umfangreiches Framework zur Entwicklung von Webservices in der Programmiersprache Python. (Django, 2022)

Django erfüllt insbesondere die Randbedingung NFA.5 Erweiterbarkeit, da es sich um eine komponententenbasierte Architektur handelt. Jede neue Komponente kann als neue Django-Applikation in die bestehende Architektur eingebunden werden. Die Erweiterung um neue Funktionalität und die geeignete Entkopplung als Software-Komponenten wird dadurch gefördert. Daher stellt Django auch für langfristige Softwareprojekte eine geeignete Wahl dar. Ein weiterer Vorteil ist der Freiraum, welches das Django-Framework bietet. Die Architektur ist erweiterbar und auf die individuellen Anforderungen dieses Projektes anpassbar.

E.4 Entwurfsentscheidung Persistenzlösung

Ein großer Vorteil von Django ist der Umgang mit Persistenzlösungen. Insbesondere die Erstellung von Tabellen und Relationen wird durch automatisiert erstellte Migrationsbefehlsketten erleichtert. Diese werden auf Grundlage der vorhandenen Datenmodellen und ihren Relationen erzeugt. Die Austauschbarkeit der Persistenzlösung ist durch die automatisierte Migration nicht teuer. Dennoch sollte eine geeignete Lösung gewählt werden und auch der gesamte

Systemkontext und die zukünftige Wartung sollten bei der Entscheidungsfindung mit in Betracht gezogen werden.

Das bereits eingesetzte Degreeday-API Backend nutzt eine PostgreSQL-Datenbank zum Speichern von Wetterdaten. Um in der Systemlandschaft eine einheitliche Persistenztechnologie zu erhalten und keine neue Technologie einzuführen, wäre es von Vorteil, für die EnergyData-Tools ebenfalls PostgreSQL zu verwenden. PostgreSQL ist eine Objekt-Relationale Datenbank mit hoher Performance und Robustheit. Sie erfüllt alle Anforderungen, welche die EnergyData-Tools an eine Persistenzlösung haben.

Aus diesen Gründen wird für die EnergyData-Tools ebenfalls eine PostgreSQL zum Einsatz kommen, um die Wartungskosten des Gesamtsystems gering zu halten. Bei der Wartung ist nur Expertenwissen über eine Art von Persistenzlösung nötig.

E.5 Entwurfsentscheidung Komponenten

Der Systementwurf soll komponentenorientiert sein. Die Funktionalitäten bilden die Use Cases aus der Anforderungsanalyse ab und werden in Use Case Komponenten gekapselt. Diese Hauptkomponenten, oder Use Case Komponenten, bieten für die Umsetzungsphase sinnvolle Arbeitspakete. Bei der Reihenfolge der Implementierung müssen die Abhängigkeiten zwischen den Komponenten beachtet werden.

E.6 Entwurfsentscheidung Architekturstil

Der Mehrwert, welche durch die Software entstehen soll, steht in direkter Abhängigkeit zu den Use Cases. Jeder Use Case beschreibt genau einen Anwendungsfall, von dem ein Nutzer des Systems einen Mehrwert hat. Daher sollen die Komponenten des Systems stark an den Use Cases orientiert sein. Die NFA.5 Erweiterbarkeit & Wiederverwendbarkeit und die damit einhergehenden Vorteile soll durch die Kapselung in mehrwertschaffende Use Case Komponenten erfüllt werden.

Durch die Wahl auf das Framework Django sind die groben Züge der Architektur festgelegt. Das Django Framework verwendet den MVT-Architekturstil, eine Abwandlung des Model-

View-Controller Stils. (Gang of Four, 1994, pp. 4-6) Hierbei ersetzt das Django-Template den View. Der klassische Django-View wird als Controller eingesetzt.

Das System in diesem Entwurf nutzt eine REST-API als einzige externe Schnittstelle. Daraus folgt, dass es keinen Bedarf für das Django-Template gibt, welches standardmäßige als Frontend eingesetzt wird. Das Frontend ist nicht Teil dieses Architekturentwurfs, sondern wird als separates Projekt in einer React-Application umgesetzt (Siehe Systemspezifikation).

E.7 Entwurfsentscheidung Schichtenarchitektur

Django ist für spezielle Anwendungsfälle flexibel gestaltet und ist in der Umsetzung wenig restriktiv. Daher sind weitere Entwurfsentscheidungen nötig, welche den Architekturstil eingrenzen und auf die Systemanforderungen zuschneiden. Zum Einsatz kommt eine Schichtenarchitektur, welche näher an der klassischen 3-Schichtenarchitektur ist. Die wichtigste Regel für diese Architektur ist, dass die Vertikalität eingehalten wird. Daraus folgt, dass jede Subkomponente in den einzelnen Schichten allein ihrer Verantwortlichkeit nachgeht. In der Praxis darf ein View keinen Zugriff auf ein Repository machen. Es dürfen bei Aufrufen zwischen Subkomponenten keine Schichten übersprungen werden. Selbiges gilt für Abhängigkeiten zwischen Hauptkomponenten. Eine Datenverarbeitungskomponente darf auf die Repository-Schnittstellen oder die Datenverarbeitungskomponenten einer anderen Hauptkomponente zugreifen, jedoch nicht auf dessen Präsentations-Schicht. Weitergehend sollen die Schichten Top-Down verwendet werden. Daraus folgt, dass jede Schicht ihre Schnittstellen der darüber liegenden Schicht anbietet und die der darunter liegenden Schicht nutzt.

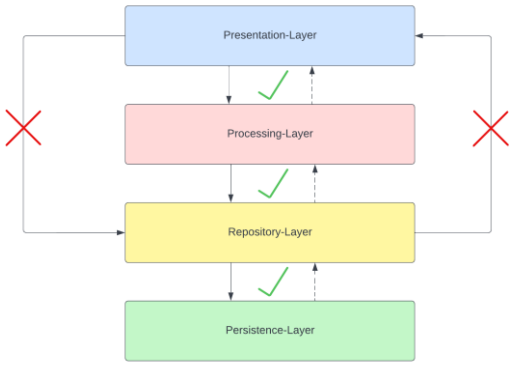

Abbildung 7 Schichtenarchitektur

E.8 Entwurfsentscheidung Datenzugriffsschicht

Die Datenzugriffsschicht („Repository-Layer“) folgt dem Repository-Pattern und bündelt die Datenzugriffe einer Komponente in eine eigene Klasse. Mit steigender Komplexität kann die Nutzung von Design-Patterns vorteilhaft sein und insbesondere der Datenzugriff benötigt daher eine erhöhte Abstraktion und Kapselung.

Daher wird die bestehende Architektur um das Repository-Pattern erweitert. Die festgelegte Regel ist, dass der Programmiercode in einer Komponente, welche direkten Zugriff auf die Datenbankmodelle macht, in die zugehörige Respository-Klasse gehört. Es sollte sich nach dieser Architekturregel kein Datenbankszugriff in einer anderen Schicht befinden. Durch diese Entwurfsentscheidung soll die Wiederverwendbarkeit von Programmiercode verbessert werden, da viele Zugriffsoperationen und komplexere Datenoperationen voraussichtlich häufiger verwendet werden.

E.9 Entwurfsentscheidung Datenverarbeitungsschicht

In der Datenverarbeitungsschicht („Processing-Layer“) werden die Datensätze verarbeitet. Sie kapselt wiederverwendbare, komplexe Datenperationen jeder Komponente in eine eigene Klasse. Diese Architekturschicht unterscheidet sich von der Datenzugriffsschicht insofern, als dass sie keine direkten Datenbankzugriffe macht, sondern hierfür die wiederverwendbaren Operationen Repository-Klassen benutzt. Die Datenverarbeitungsschicht führt mit den gegebenen Mitteln aus Datenverarbeitungsbibliotheken aus E.2 Entwurfsentscheidung Bibliotheken die gewünschten Datenoperationen durch. Die Wiederverwendbarkeit und das Single Responsibility Prinzip sind, analog zu E.8 Entwurfsentscheidung Datenzugriffsschicht, ausschlaggebende Gründe für die Einführung dieser zusätzlichen Architekturschicht.

E.10 Entwurfsentscheidung Präsentationsschicht

Der standardmäßige Einsatz von Django würde eine Frontend-Schicht beinhalten. In diesem Entwurf wird das Django-Framework jedoch nicht für Front- und Backend eingesetzt, sondern ausschließlich als Backend und zur Bereitstellung einer REST-API. Die Frontend-Schicht kann wird redundant und kann vollkommen weggelassen werden. Die Views, welche die Präsentationsschicht abbilden, stellen die REST-API bereit.

4.2 Sichten

4.2.1 Kontextabgrenzung

Die Kontextabgrenzung soll eine nicht-technische Sicht auf das Gesamtsystem darstellen. Die Darstellung als UML-Komponentendiagramm wurde für die Kontextsicht gewählt, wobei auch die Schnittstellen-Richtung dargestellt ist. Der Hauptzweck der Darstellung ist insbesondere, nicht-technischen Stakeholdern den Systemkontext verständlich machen zu können. Die Schnittstellen-Richtung kann sich dabei als hilfreich erweisen, um beispielsweise die Interaktion zwischen dem Frontend und dem Backend zu erläutern. Alle Systemkomponenten sind als Black-Box dargestellt.

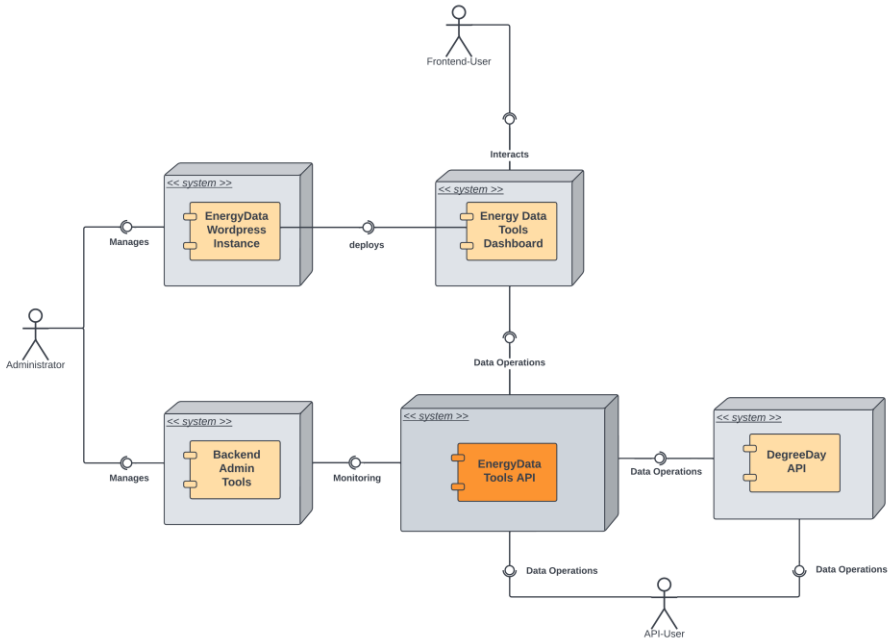

Abbildung 8 Kontextabgrenzung

In der Abbildung 5 Ist-Zustand Infrastruktur ist dargestellt, wie die Website und die Degreeday-API zu Projektbeginn betrieben werden. Im Mittelpunkt steht die Wordpress-Instanz. Auf dieser wird die Benutzeroberfläche für die Degreeday-API gehostet.

Mit den neuen Anforderungen soll eine neue Systemlandschaft entstehen. Hierbei werden drei neue Systemeblöcke eingeführt. Das EnergyData-Dashboard, welches bestehende Front-

end-Komponenten übernimmt und die Benutzeroberfläche für die neuen EnergyData-Tools bereitstellt. Die BackendAdmin Tools zur Überwachung und Monitoring und die EnergyData Tools API, das in dieser Arbeit beschriebene Backend-System. Die Degreeday-API wird vom Frontend entkoppelt und an das neue Backend angebunden. In der Abbildung 8 sind die groben Richtungen der Abhängigkeiten dargestellt, sowie die Sichten der verschiedenen Nutzer auf das Gesamtystem.

EnergyData Tools API

Das in dieser Arbeit entworfene und prototypisch umgesetzte Backend-System. Die Kommunikation mit externen System erfolgt ausschließlich über eine REST-API. Es ist für die Abbildung der Use Cases verantwortlich und ermöglicht die Verarbeitung von Daten über eine REST-API.

EnergyData Wordpress Instanz

Die Wordpress Instanz betreibt die Webseite der Energy Data GmbH. Sie übernimmt die Präsentation des Unternehmens, die Akquise von neuen Kunden, dem Verarbeiten von Bezahlvorgänge und Subscriptions, Benutzerregistrierung und -verwaltung ist. Die Benutzerinformationen werden zur Authentifizierung und Berechtigungsprüfung in dem Backend-System genutzt. Auf dem Wordpress Instanz läuft das neue Frontend als React Applikation, welches in Abbildung 8 übesichtshalber als eigener Systemblock dargestellt ist.

EnergyData-Tools Dashboard

Das EnergyData-Dashboard ist eine React-Application, welche nur registrierten Nutzern und Bezahlkunden zugänglich ist. Hier können Endnutzer die EnergyData-Tools nutzen und ihre Energiedaten verarbeiten, visualisieren und analysieren. Es stellt das Frontend zu dem hier beschrieben Softwareentwurf dar und wird parallel dazu entwickelt. Das Frontend nutzt die REST-API der EnergyData-Tools und nutzt die Benutzerinformationen aus dem Wordpress-Backend zur Authentitifizierung.

Degreeday-API - Beschreibung

Die Degreeday-API ist eine eigenständige Microservice-Architektur entwickelt in Golang, welche ein Data-Service zur Bereitstellung von Degreedays darstellt. Es ist bereits funktionsfähig und wird vor dem Beginn dieses Projektes vom Frontend für das Tool „Degreeday Calculator“ genutzt.

Es besteht eine starke Abhängigkeit zwischen den neuen Tools und der Degreeday-API, da für die viele Datenoperationen, welche die EnergyData-Tools anbieten, Degreedays und historische Daten genutzt werden müssen.

Degreeday-API - Schnittstelle

Mit Query-Parametern werden der Degreeday-API die operationsspezifischen Parameter mitgegeben. Die Schnittstellen antworten mit CSV-Dateien im Request-Body. Das von der der Degreeday-API verwendete Kommunikationsprotokoll ist HTTP.

Die Degreeday-API ist ausführlich dokumentiert. Im Nachfolgenden sind daher nur die verfügbaren Parameter aufgelistet, um einen Überblick zu geben.

Degreeday-API – Parameter:

- Indicator – Indikator für Art von Gradtagszahl
- Tb - Basistemperatur
- Start - Startzeitpunkt
- End - Endzeitpunkt
- Breakdown – Intervall der Gradtage
- Key – API Key
- Week_Start – Optional: Start der Woche (bei wöchentlichen „breakdown“)

Backend Admin Tools

Die Backend Admin Tools soll den Administratoren ermöglichen das Backend zu überwachen und Benutzeraktivitäten nachzuvollziehen. Für das Minimum Viable Product ist es ausreichend, wenn Administratoren Webzugriff auf die Postgres-Datenbank haben.

4.2.2 Verteilungssicht

Die Verteilungssicht bildet die Systemtopologie ab. Des Weiteren sind die Kommunikationsprotokolle zwischen den Infrastruktur-Einheiten dokumentiert.

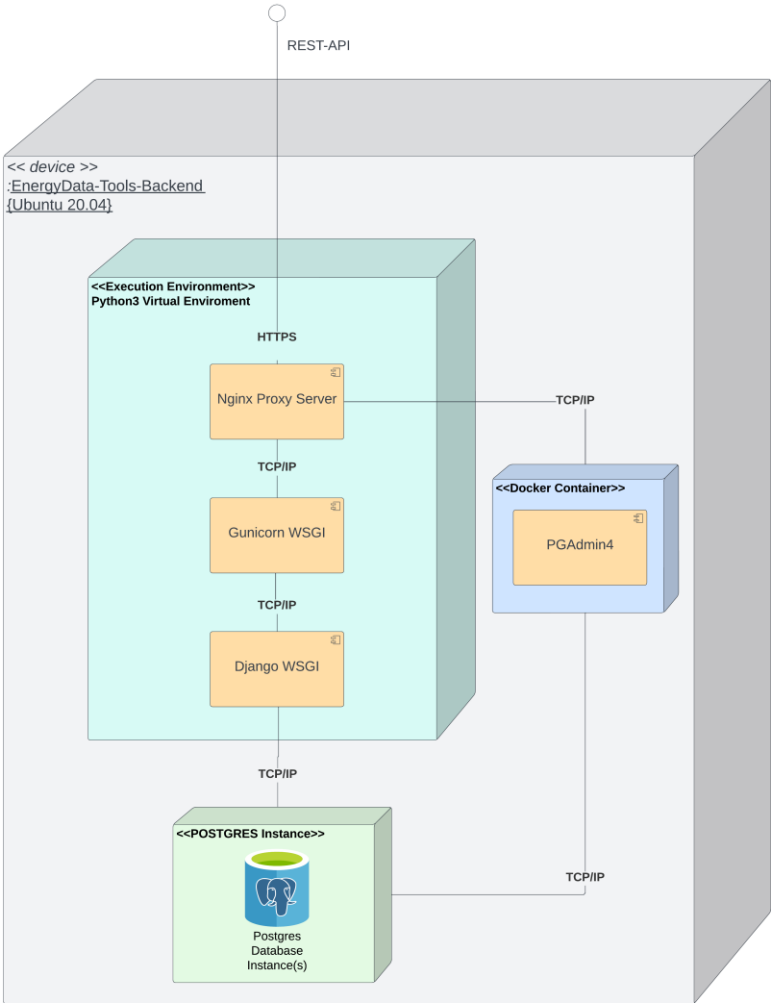

Abbildung 9 Verteilungssicht

Ubuntu Server

Für die Bereitstellung des Backends kommt ein Ubuntu Server zum Einsatz. Auf dem Ubuntu Server wird die Backend-Infrastruktur umgesetzt und gewartet.

Python Virtual Environment

Die benötigte Webserver-Infrastruktur soll in einer virtuellen Python Umgebung bereitgestellt werden. In dieser Laufzeitumgebung sind die benötigten Abhängigkeiten installiert. Django ermittelt die benötigten Abhängigkeiten. Dies soll die Bereitstellung und die Wartung der Infrastruktur erleichtern.

Nginx Reverse Proxy Server

Nginx ist ein WSGI Server und soll in dem Deployment als Middleware bzw. Reverse Proxy Server eingesetzt. Der Proxy Server ist für die Verarbeitung aller Client Anfragen zuständig und leitet diese zu ihrem Ziel weiter.

Die Anfragen für die PGAdmin Instanz sollen an die entsprechende Docker Instanz weitergeleitet werden. Alle anderen Anfragen sollen an die Gunicorn Instanz weitergeleitet werden.

Für die Einhaltung der nicht-funktionalen Anforderungen an das System in Bezug auf die Sicherheit soll Nginx HTTPS einsetzen.

Datenbank Administration

Die Datenbank Administration soll durch eine PGAdmin4-Applikation realisiert werden, welche alle gängigen Datenbankmanagement Werkzeuge zur Verfügung stellt sowie Einblicke in das Nutzungsverhalten der Endnutzer gibt. Die PGAdmin4-Instanz soll in einem Docker Container auf dem Ubuntu Server bereitgestellt werden und läuft daher unabhängig von der virtuellen Python Umgebung der Backend Tools.

4.2.3 Bausteinsicht - Visualisierung

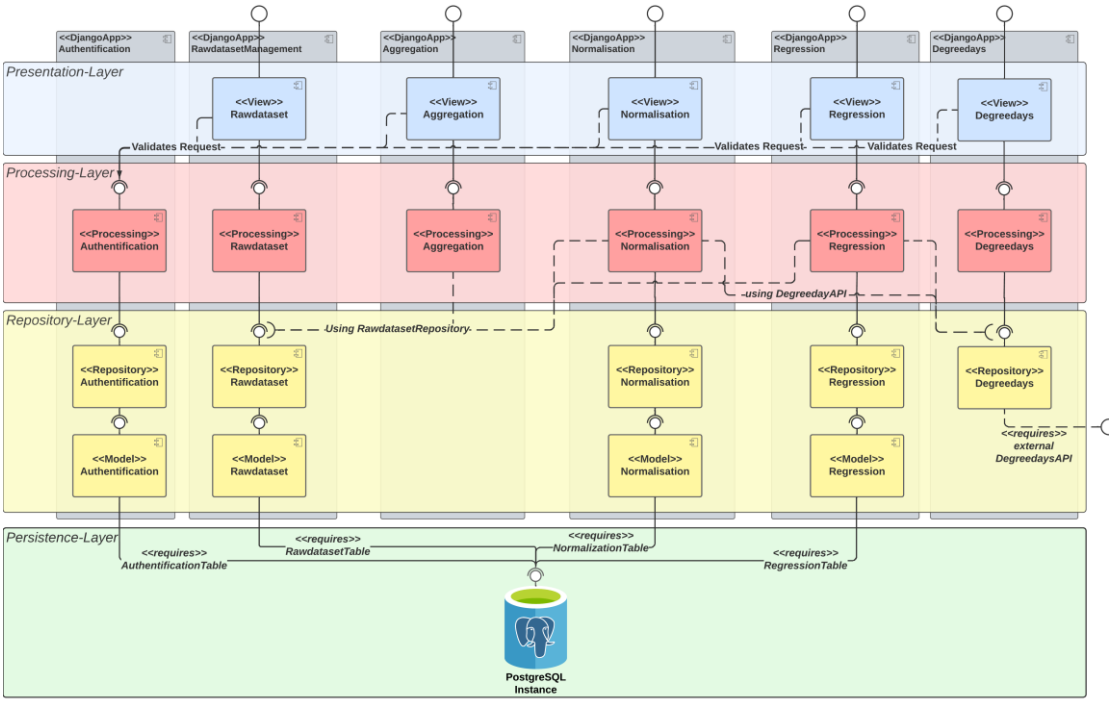

Abbildung 10 Bausteinsicht

UML, gestrichelte Linien kennzeichnen Abhängigkeiten zwischen Hauptkomponenten

4.2.4 Bausteinsicht – Komponenten

Jede Hauptkomponente wird im folgenden Abschnitt erläutern und mit K.n nummeriert. Jede Definition beschreibt die allgemeine Zuständigkeit, die REST-Operationen, Besonderheiten und Abhängigkeiten der jeweiligen Hauptkomponente.

Grundsätzlich gibt die Bausteinsicht in diesem Entwurf möglichst wenig bis keine Implementierungsdetails vor. Die Vorgaben für die Realisierung entstehen durch die Abhängigkeiten zwischen Komponenten, die Richtungen der Schnittstellen sowie die Einhaltung der Vorgaben aus den Entwurfsentscheidungen.

4.2.5 Bausteinsicht - Subkomponenten

Die Hauptkomponenten, in der Abbildung 10 Bausteinsicht als grau dargestellt, haben jeweils Subkomponenten entsprechend der Schichtenarchitektur. Jede Hauptkomponente kapselt ihre Funktionalitäten entsprechend der Vorgaben in den dazugehörigen Subkomponenten. In diesem Abschnitt werden Eigenschaften beleuchtet, die für alle Subkomponenten gelten (Siehe E.6 Entwurfsentscheidung Architekturstil).

SK.1 View (Präsentationsschicht)

Die Views bilden die Präsentationsschicht einer Use Case Komponenten ab. Die Views verarbeiten die Anfragen an die REST-API und nutzen die Schnittstellen der Processing-Subkomponenten für die Verarbeitung der Daten.

SK.2 Processing (Datenverarbeitungsschicht)

Die Processing-Subkomponente kapseln die Datenverarbeitung einer Use Case Komponenten. Sie verarbeiten die Anfragen an den Views und nutzen die Schnittstellen der Repostiory-Subkomponenten für das Abfragen und Speichern von Daten.

Durch die unterschiedlichen Anforderungen der Use Cases kann sich die Umsetzung der Processing-Komponenten mitunter stark unterscheiden. Die Auswahl von geeigneten Bibliotheken und Implementierungsaspekten sind individuell an die Anforderungen des Use Case anzupassen.

SK.3 Repository (Datenzugrifsschicht)

Die Repository-Subkomponente kapseln die Datenzugriffe ihrer Use Case Komponenten. Sie verarbeiten die Anfragen an der Processing-Subkomponente und bieten wiederverwendbare Operationen an. Jede Repository-Subkomponente nutzt die Model-Subkomponente.

SK.4 Model (Datenzugrifsschicht)

Die Model-Subkomponente wird von der Repository-Komponente verwendet, um die Datenentitäten abzubilden. In dieser Komponente werden die Datentypen und die Relationen des fachlichen Datenmodells abgebildet (Siehe Fachliches Datenmodell).

4.2.6 Bausteinsicht – Use Case Komponenten

K.1 Authentifcation

Die Komponente „Authentification“ ist für die Authentifizierung von allen Requests zuständig, welche die REST-API bzw. die jeweiligen View-Subkomponenten verarbeiten soll.

Hierfür werden Logiken abgebildet, um verschiedene Zugriffssituationen von Nutzern zu unterstützen und dabei einen Missbrauch der API zu verhindern. Die Authentification Komponente beruft sich auf die in der Wordpress-Instance gespeicherten und verwalteten API Keys und User IDs.

Besonderheit

Die Authentification Komponente ist eine interne Komponente und bildet keinen Use Case ab. Es ist keine REST-API und damit kein View vorgesehen.

K.2 Rawdataset

Die Rawdataset Komponente ist vorrangig für das Hochladen und die Verwaltung von Datensätzen zuständig. Sie bietet Schnittstellen zum Hochladen, ermöglicht die Abrufbarkeit von Metainformationen sowie die effiziente Abfrage aller Daten eines Datensatzes (Siehe Use Case).

Die Datensätze können als präsentierbare CSV-Datei exportiert werden (Siehe Use Case 3 – Herunterladen von Datensätzen in CSV-Format). Verwaltungsoperationen und Filteroperationen werden ebenfalls über die REST-Schnittstellen angeboten.

REST-Schnittstellen

Method	***URL***	***Beschreibung / Referenz***
POST	/rawdataset/upload/	Hochladen und formatieren von einem Energieverbrauchsdatensatz
PUT	/rawdataset/{UUID}/	Informationen vom Datensatz ändern
DELETE	/rawdataset/{UUID}/	Datensatz löschen
GET	/rawdataset/{UUID}/	Datensatzinformationen abrufen
GET	/rawdataset/{UUID}/data/	Inhalt vom Datensatz abrufen
GET	/rawdataset/{UUID}/csv/	Datensatz als CSV-Datei abrufen
GET	/rawdataset/getByUser/	Datensätze nach User filtern
GET	/rawdataset/getByKey/	Datensätze nach APIKey filtern
GET	/rawdataset/getBySession/	Datensätze nach Session filtern

Tabelle 3 REST-Schnittstellen Rawdataset

K.3 Aggregation

Die Komponente „Aggregation“ ist für die Formatierung und Aggregation von Datensätzen zuständig. Sie bietet die Möglichkeit die zeitliche Form von Datensätzen anzupassen. Prüfungsmethoden validieren die Korrektheit der Aggregation. Anschließend wird der Rohdatensatz überschrieben. Vor der Aggregation in monatliche oder jährliche Zeitintervalle soll der Datensatz auf tagesschärfe aggregiert werden.

Abhängigkeiten

Die Aggregation Komponente steht in direkt Abhängigkeit zur K.2 Rawdataset Komponenten und nutzt deren Repository-Subkomponente, um die Veränderungen an den Datensätzen zu persistieren. Besonderheit: Die Aggregation Komponente hat keine Repository-Subkomponenten, da die Operationen auf den Rohdatensätzen der Komponente durchgeführt werden.

REST-Schnittstellen

Method	*URL*	*Beschreibung / Referenz*
POST	/aggregation/aggregate/{UUID}/	Hochladen und formatieren von Energieverbrauchsdatensatz

Tabelle 4 REST-Schnittstellen Aggregation

K.4 Normalisation

Die Komponente „Normalisation“ ist für die Normalisierung von Datensätzen zuständig. Die in Use Case 5 – Normalisierung spezifierten Normalisierungsmethoden werden als Operationen auf einem Datensatz angeboten. Alle drei Methoden können gleichzeitig auf einem Rawdataset angewendet werden. Eine erfolgreiche Normalisierung erzeugt einen neuen, normalisierten Datensatz, welcher jeweils die normalisierten Verbrauchswerte sowie den unbereinigten Verbrauchswert hält (Siehe Fachliches Datenmodell – Normdataset-Entry).

Abhängigkeiten

Die Normalisation-Komponente steht in Abhängigkeit zu der Komponente K.2 Rawdataset, da für die Normalisierung bereits formatierte und persistierte Rohdatensätze genutzt werden. Des Weiteren besteht eine Abhängigkeit zu der Degreedays-Komponente, da die Normalisierungsmethoden die Gradtagszahlen bzw. Wetterdaten für die Wetterbereinigung benötigen.

REST-Schnittstellen

Method	*URL*	*Beschreibung / Referenz*
POST	/normdataset/normalise/	Erstellen eines normalisierten Datensatzes
PUT	/normdataset/{UUID}/	Informationen vom Datensatz ändern
DELETE	/normdataset/{UUID}/	Datensatz löschen
GET	/normdataset/{UUID}/	Datensatzinformationen abrufen
GET	/normdataset/{UUID}/data/	Inhalt vom Datensatz abrufen

GET	/normdataset/{UUID}/csv/	Datensatz als CSV-Datei abrufen
GET	/normdataset/getByUser/	Datensätze nach User filtern
GET	/normdataset/getByKey/	Datensätze nach APIKey filtern
GET	/normdataset/getBySession/	Datensätze nach Session filtern
GET	/normdataset/getByRawdataset/	Datensätze nach Rohdatensatz filtern

Tabelle 5 REST-Schnittstellen Normdataset

K.5 Regression

Die Regressions-Komponente ist für das Erstellen von Regressionsanalysen zuständig. Bei dem Aufruf dieser Operation wird auf Basis eines Rohdatensatz ein lineares Regressionsmodell erstellt und in persistiert. Bei der iterativen Regressionsanalyse werden insgesamt fünf lineare Modelle erstellt mit auf- und absteigenden Basistemperaturen (Siehe Use Case 6 – Lineare Regressionsanalyse, Use Case 7 – Iterative Lineare Regressionsanalyse).

Abhängigkeiten

Die Regressions-Komponente steht in Abhängigkeit zu der K.2 Rawdataset Komponente. Die Komponente hat zusätzlich eine Abhängigkeit zu der Degreedays-Komponente, da für die lineare Regressionsanalyse die Energieverbrauchswerte und die dazugehörigen Gradtagzahlen benötigt werden.

REST-Schnittstellen

Method	***URL***	***Beschreibung / Referenz***
POST	/regression/linearModelCV/	Erstellen eines linearen Regressionsmodell
DELETE	/regression/{UUID}/	Regressionsmodell löschen
GET	/regression/{UUID}/	Regressionmodell abrufen
GET	/regression/{UUID}/data/	Daten vom Regressionmodell abrufen

GET	/regression/getByRawdataset/	Regressionmodell nach Rohdatensatz filtern

Tabelle 6 REST-Schnittstellen Regression

K.6 Degreedays

Die Komponente Degreedays ist für die Weiterleitung von allen Requests zuständig, welche vor der Umsetzung dieses Entwurfs direkt an die bestehende Degreeday Services gingen. Diese werden nun ausschließlich über das neue Backend-System verarbeitet.

Zusätzlich werden hier die Antworten der Degreeday-API, welche ausschließlich im CSV-Format sind, zu JSON umgewandelt.

Abhängigkeiten

Die Degreedays Komponente ist von der Degreeday-API abhängig. Sollte diese nicht funktionsfähig sein, kann auch die Degreedays-Komponente nicht korrekt arbeiten.

Durch die häufige Verwendung von Degreedays in den Use Cases zu Verarbeitung der Datensätze bestehen Abhängigkeiten von der Degreedays Komponente (Siehe Abhängigkeiten K.4 Normalisation und K.5 Regression).

REST-Schnittstellen

Method	***URL***	***Beschreibung / Referenz***
GET	/degreedays/	Abfragen von Degreedays mit Parametern

Tabelle 7 REST-Schnittstellen Degreedays

4.2.7 Laufzeitsicht

Im folgenden Abschnitt werden die Laufzeitsichten der Hauptkomponenten dargestellt und erläutert. Die Komponenten aus der Bausteinsicht werden hier als Laufzeitinstanzen betrachtet. Es wird die UML-Sequenzdiagramm Notation verwendet.

Authentifizierung

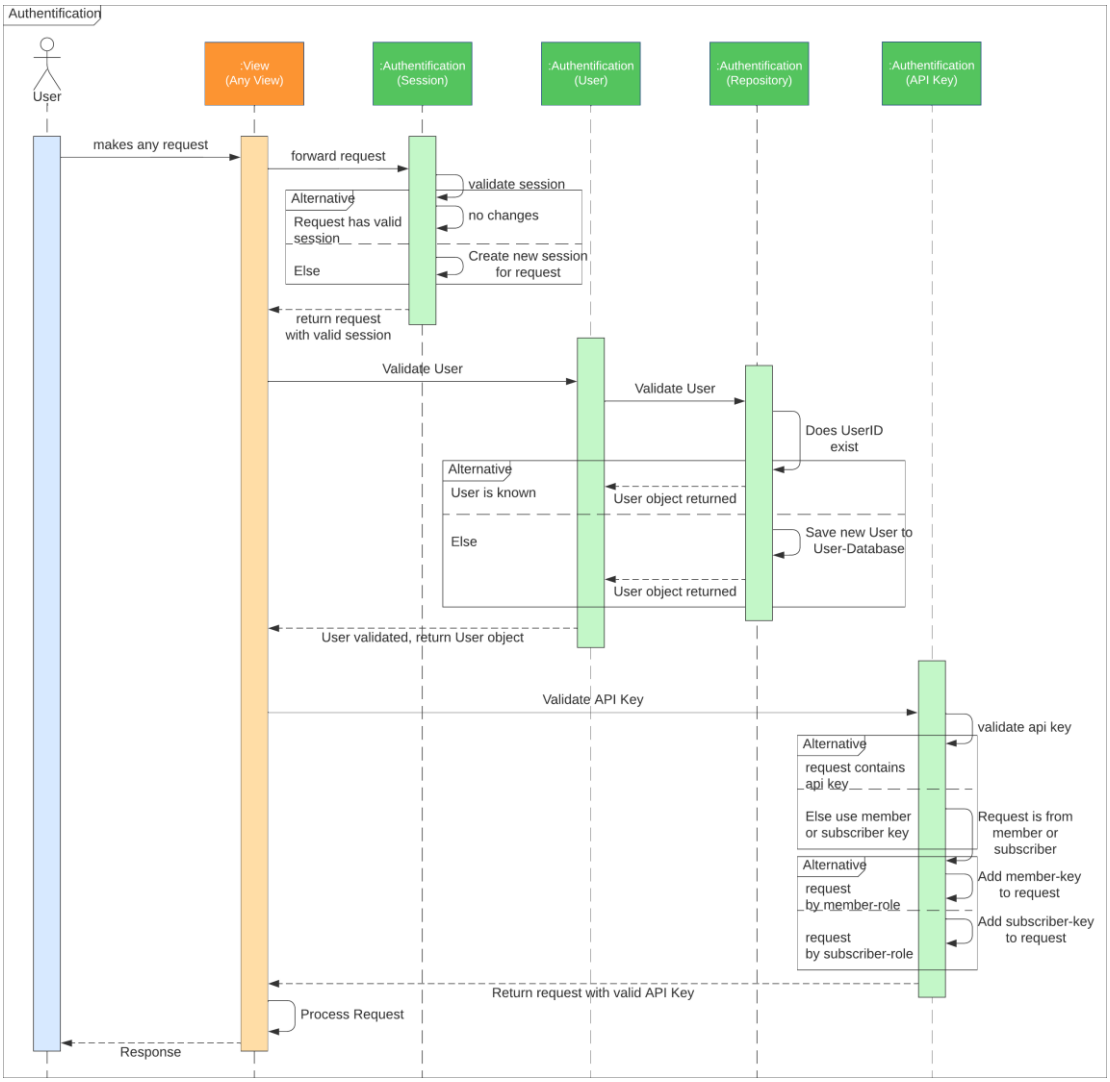

Abbildung 11 Laufzeitsicht Authentifizierung

Die Abbildung 11 Laufzeitsicht Authentifizierung visualisiert den Prozess der Authentifizierung eines beliebigen Requests. Jeder Requests, der von der REST-API empfangen wird, durchläuft diesen Prozess.

- Session validieren oder neue Session erstellen
- User ID validieren oder neuen User anlegen
- API Key validieren, oder Rolle prüfen und API Key hinzufügen

Für die Validierung der Session wird geprüft, ob der Request eine Session ID hat. Wenn nicht, wird eine neue Session für diesen Request erstellt.

Anschließend wird der User validiert. Die im Request enthaltene User ID wird durch die Wordpress-Instanz als Cookie mitgeliefert. Die Authentifizierungskomponente prüft, ob dieser User bereits in der Datenbank steht. Wenn nicht, wird eine neue User Instanz gespeichert.

Bei der Validierung des API Keys wird geprüft, ob dem Request einen validen API key mitgegeben wurde. Ist dies der Fall, wird dieser API Key verwendet. Sollte kein API Key mitgegeben wurden, handelt es sich entweder um einen eingeloggten Free User oder einen Subscriber.

Free User und Subscriber haben unterschiedliche Rechte bei dem Aufruf der Degreedays-API und werden daher unterschiedliche API Keys zugewiesen bekommen. Wenn der Request von der Rolle Subscriber kommt, dann kriegt der Request den entsprechenden API Key angehangen. Wenn der Request von keiner bestimmten Rolle kommt, wird der eingeschränkte Member Key verwendet.

Verwaltungsoperationen

Die Verwaltungsoperationen gelten für Entitäten:

- Fachliches Datenmodell - Rawdataset
- Fachliches Datenmodell – Normdataset
- Fachliches Datenmodell – Regression Analysis

Auf die Erstellung von Laufzeitdiagrammen für alle Verwaltung- und Filteroperationen der Komponenten wird verzichtet, da diese Vorgänge vergleichsweise trivial sind.

- View verarbeitet Request (Unterscheidung POST, GET, UPDATE etc.)
- View nutzt Processing-Layer für Bereitstellung und Formatierung der Rückgabe
- Processing-Layer nutzt Repository-Layer für den Filteroptionen
- RepositoryLayer macht die Datenbankzugriffe und gibt die Daten zurück

Das Repository nutzt die zugehörigen Models, um Zugriffe auf der Datenbank durchzuführen. Die Antworten werden von der Processing-Komponente formatiert im JSON-Format an den View zurückgegeben. Sollte bei den Zugriffen ein Fehler auftreten, werden menschenleserliche Fehlermeldungen zurückgegeben.

Das Löschen soll kaskadisch erfolgen, d. h. beispielsweise die Löschung eines Datensatzes mit einer bestimmten ID löscht auch den übergreifenden „Rawdatset“ Eintrag, den „Rawdataset-Metadata“ Eintrag sowie alle dazugehörigen „Rawdaset-Entries“.

Upload und Formatierung

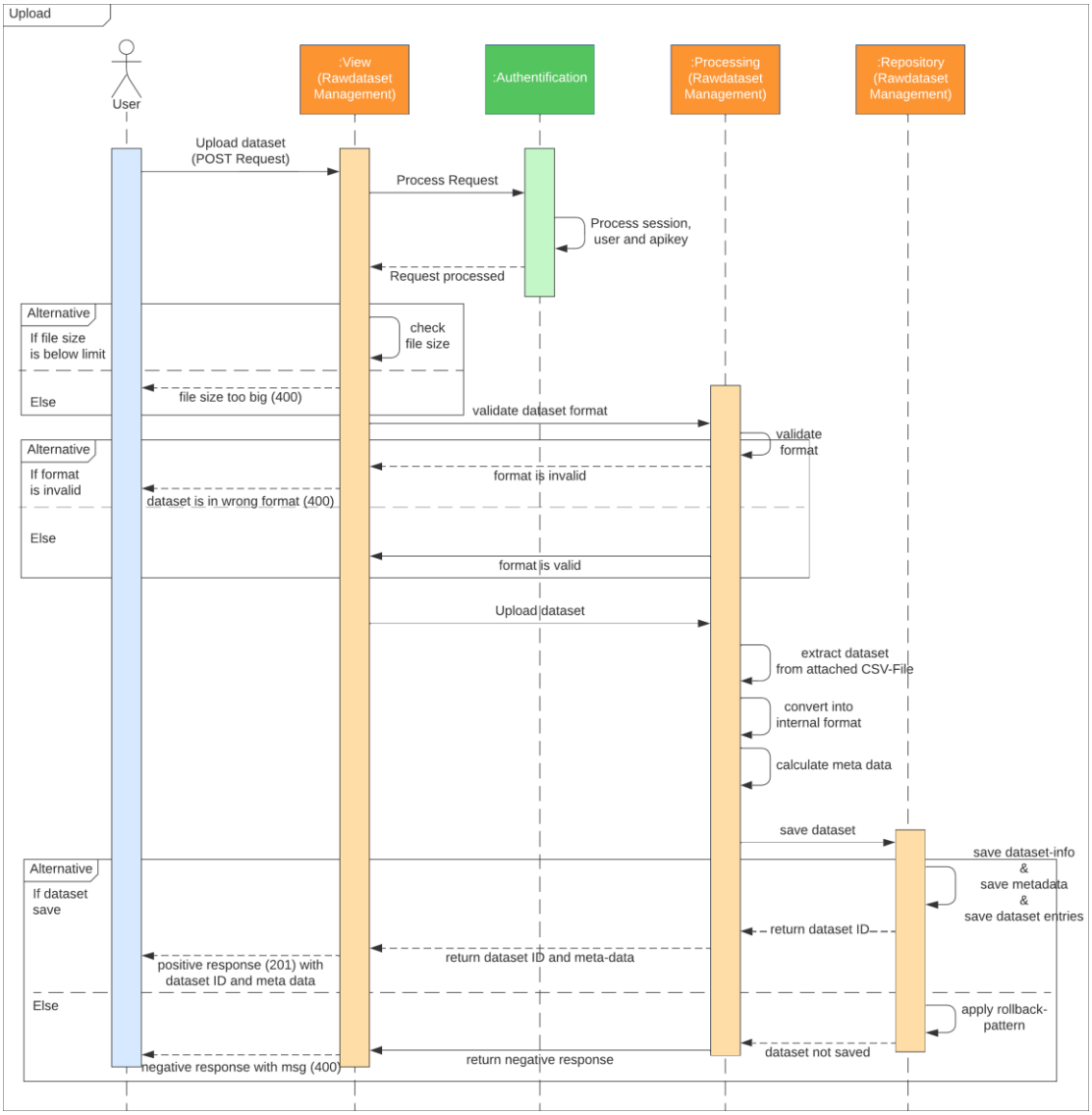

Abbildung 12 Laufzeitsicht Upload

Das Hochladen von Datensätzen erfolgt im CSV Format. Die Datensätze werden als Datei an die Requests erwartet. Die Upload Schnittstelle prüft die Datensätze auf Korrektheit und wandelt verschiedene Ausprägungen von Datensatzformate in das interne Format um (Siehe Internes Messdatenformat).

Bei erfolgreichem Hochladen wird ein Datensatz mit ermittelten Metainformationen angelegt und die einzelnen Einträge gespeichert. Die Metainformationen, die beim Hochladen angelegt werden, sind ein Fachliches Datenmodell – Rawdataset-Metadata spezifiziert.

Der Requests für Upload und die Formatierung der Datensätze wird von dem K.2 Rawdataset verarbeitet:

- Der angehängte Datensatz wird auf maximale Dateigröße überprüft
- Die Processing-Komponente überprüft das Format der CSV-Datei
- Sind die Bedingungen einer validen CSV-Struktur erfüllt, werden die Daten aus der CSV-Datei extrahiert. Dabei wird auf verschiedene CSV-Separatoren geprüft und verschiedene Ausprägungen der Messdaten verarbeitet
- Die Daten werden in das interne Format überführt
- Die Metainformationen werden aus dem Datensatz extrahiert
    - Ermittlung von Durchschnittswerten und Summierungen über die Energieverbrauchswerte
    - Zählen von Lücken
    - Messwerten ohne Verbrauch
    - Start und Enddatum der Messungen
    - Durchschnittslänge der Messzeiträume
- Die Schnittstellen der Repository-Komponenten werden für die Speicherung genutzt
    - Speicherung der Entität: Fachliches Datenmodell - Rawdataset
    - Speicherung der Entität: Fachliches Datenmodell – Rawdataset-Metadata
    - Speicherung der Entitäten: Fachliches Datenmodell – Rawdataset-Entry

Aggregation

Die verfügbaren Operationen zur Aggregation der Daten in verschiedene Zeitintervalle werden alle über die REST-Schnittstelle der Aggregationskomponente verarbeitet. Je nach Operationsparametern werden verschiedene Aggregationen ausgeführt (Siehe Use Case 4 – Aggregation zeitlicher Intervalle).

Vor jeder Operation wird der Datensatz auf tägliche Intervalle aggregiert. Der Datensatz kann minutenweise Messungen enthalten. Für die Aggregation in monatliche oder jährliche Intervalle müssen die Messungen auf tagesschärfe aggregiert worden sein. Bevor der Datensatz überschrieben wird, validieren Prüfungsmethoden die Korrektheit der Operationen.

- Auf tagesschärfe aggregieren
- Tagesschärfe validieren
    - Optional: Monate zusammenfassen und validieren
    - Optional: Jahre zusammenfassen und validieren

Normalisierung

Die verfügbaren Operationen zur Normalisierung der Daten werden alle über die REST-Schnittstelle der Normalisierungskomponente verarbeitet. Je nach Operationsparametern werden verschiedene Normalisierungen auf dem Datensatz angewendet und persistiert.

LTA - Ablauf

- Degreedays für den Messzeitraum von der Degreedays-API abfragen
- Degreedays und Langzeitdurchschnitt auf die Intervalle der Messdaten aggregieren
- Für jede Messung im Energieverbrauchsdatensatz:
    - Den normalisierten Verbrauchswert mit der LTA Formel ermitteln

BYA - Ablauf

- Ermitteln, ob der Datensatz mindestens ein ganzes Jahr enthält
    - Wenn nicht, dann kann die Methode nicht angewendet werden, weil das Basisjahr nicht zur Verfügung steht.
- Jeder Zeitraum in den Verbrauchswerten wird mit dem korrespondierenden Zeitraum im Basisjahr normalisiert.

ER - Ablauf

- Das Berechnungsverfahren für die Ermittlung der Kennzahlen wird auf alle Verbrauchswerte angewendet.

Regression

Im Nachfolgenden ist der Ablauf für das Erstellen von einem linearen Regressionsmodell basierend auf einem Datensatz abgebildet (Siehe Use Case 6 – Lineare Regressionsanalyse). Erkennbar sind die Abhängigkeiten zu der Repository Subkomponente, der Rawdataset Management. K.2 Rawdataset und dem Repository der K.6 Degreedays.

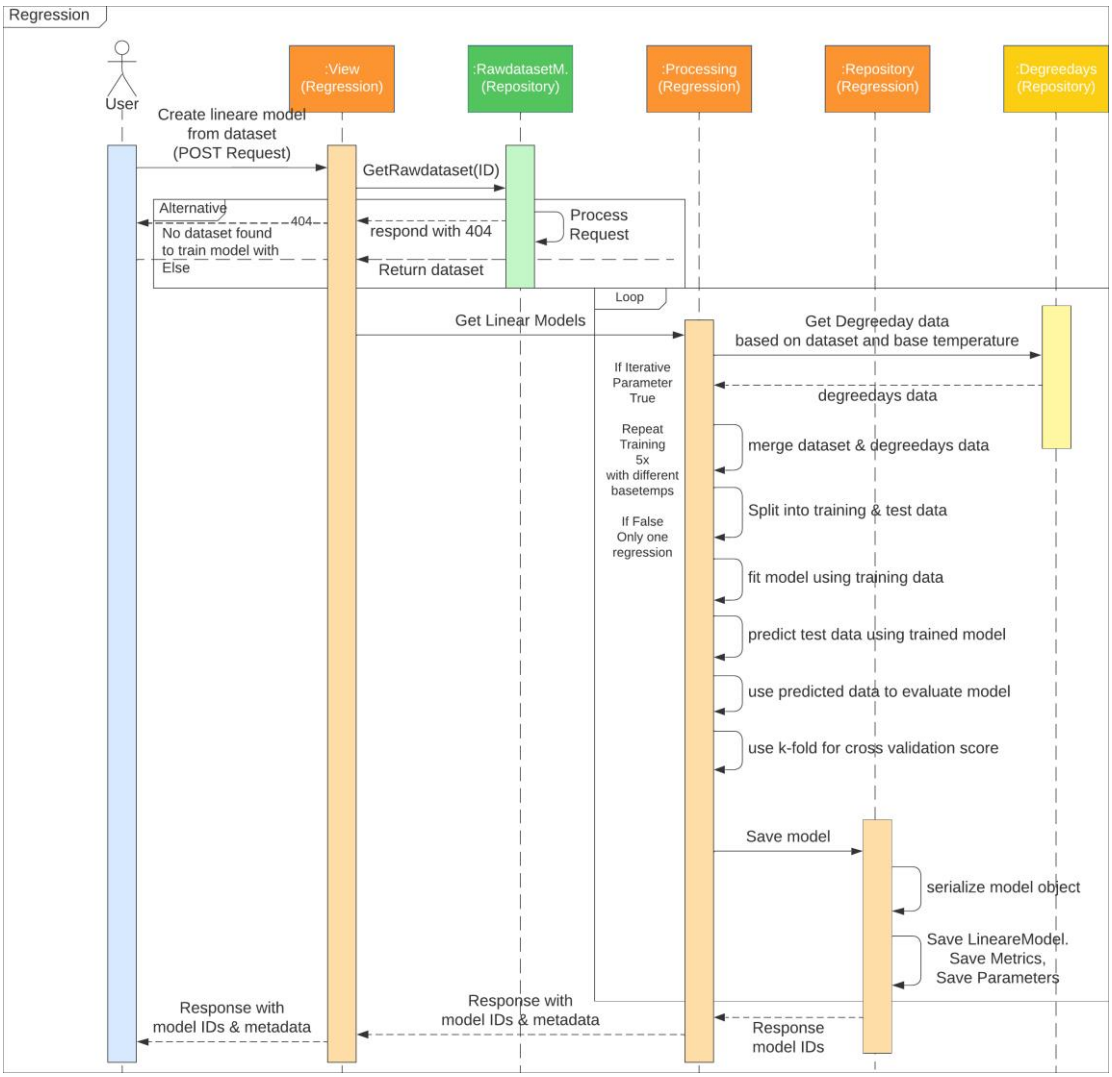

Abbildung 13 Laufzeitsicht Regression

Sollte bei dem Aufruf kein gültiger Rohdatensatz angegeben sein, dann kann auch kein lineares Modell erstellt werden. Ist ein Datensatz vorhanden, dann wird basierend auf dessen zeitlichen Messabständen die Degreedays angefragt.

Der Processing-Layer der Regressionskomponente bereitet die Daten für die Regressionsanalyse vor. Hierfür müssen die Messdaten mit den Degreedays in einer Tabelle zusammengeführt werden (Siehe E.2 Entwurfsentscheidung Bibliotheken).

- Energieverbrauchsdaten und Gradtagszahlen in eine Tabelle überführen
- In Trainings- und Testdaten aufteilen, die Aufteilung kann je nach Parametern und Einstellungen variieren
- Das Modell mittels Ordinary Least Squares auf dem Trainingsdatensatz trainieren
- Das Modell auf dem Validierungsdatensatz testen, es entsteht ein Datensatz mit vorhergesagten Werten
- Die Genauigkeit der vorhergesagten Werte überprüfen und Das Bestimmtheitsmaß $\boldsymbol{R^2}$ ermitteln
- Eine Cross-Validation Score mittels k-Fold ermitteln

Ein lineares Modell ist trainiert und die geforderten Metriken sind ermittelt, wenn dieser Vorgang erfolgreich abgeschlossen ist. Anschließend werden die Daten in den entsprechenden Entitäten persistiert (Siehe Fachliches Datenmodell – Regression Analysis). Hierfür verwenden die Processing-Subkomponenten die Schnittstellen der Repository-Komponente.

In der Abbildung 13 Laufzeitsicht Regression ist ein „Loop“ abgebildet. Dieser bildet den Use Case 7 – Iterative Lineare Regressionsanalyse ab, bei dem die Basistemperatur verändert wird und iterativ fünf Regressionsmodelle trainiert werden. Jeder Durchlauf wird individuell persistiert.

Degreedays

Die Degreeday Komponente leitet lediglich alle Anfragen an die Degreedays-API weiter, die vorher direkt an die Degreeday-Services gingen. (Vergleiche Ist-Zustand und Verteilungssicht)

Die REST-Schnittstelle liest jene Parameter ein, die für das Abfragen der Degreedays nötig sind (Siehe Degreeday-API - Schnittstelle).

Die Anfragen werden weitergeleitet und die Degreedays-API wird aufgerufen und die Antwort wird zurückgeben.

4.3 Schnittstellenverträge

Die Schnittstellenspezifikation ein wichtiger Bestandteil des Softwareentwurfs, um die Abhängigkeiten zwischen den Komponenten vertraglich festzulegen und transparent zu dokumentieren. (Sommerville, 2018, p. 237).

Übersichtshalber sind in den folgenden Grafiken die Schnittstellen-Verträge zwischen Use Case Komponenten abgebildet und erläutert. Die Subkomponenten einer Komponente haben zusätzlich die vertikalen Abhängigkeiten, die zu der Abbildung der Use Cases nötig sind und aus der verwendeten Schichtenarchitektur entstehen.

Aus der Bausteinsicht - Visualisierung ist ersichtlich, dass nur jene Abhängigkeiten erläutert werden, bei denen Subkomponenten einer Komponente Schnittstellen von anderen Komponenten erwarten.

Authentifizierung

Bezeichnung	**Schnittstellen-Vetrrag**
ValidateUser	Alle Views (SK.1 View)
ValidateSession	Alle Views (SK.1 View)
ValidateAPIKey	Alle Views (SK.1 View)

Tabelle 8 Schnittstellen Authentifizierung

Die View-Subkomponenten haben allgemeingültige Schnittstellenverträge. Jeder View nutzt die Schnittstellen der Authentifizierungs Komponente zur Validierung jedes empfangenen Requests. Jeder View verlässt sich vollständig auf die Korrektheit und nutzt die Session ID, API Key und die User ID, welche durch die verwendeten Schnittstellen validiert worden sind.

Rawdataset Repository

Bezeichnung	**Schnittstellen-Vetrrag**
SaveRawdata	Aggregation(Processing), Regression(Processing)
UpdateMetadata	Aggregation(Processing), Regression(Processing)
GetRawdataset	Aggregation(Processing), Regression(Processing), Normalisation(Processing)
GetFirstDate	Normalisation(Processing),Regression(Processing)
GetLastDate	Normalisation(Processing),Regression(Processing)

Tabelle 9 Schnittstellen Rawdataset Repository

Degreedays Repository

Bezeichnung	**Schnittstellen-Vetrrag**
GetDegreedays	Normalisation(Processing),Regression(Processing)

Tabelle 10 Schnittstellen Degreedays Repository

5 Realisierung

5.1 Stand der Umsetzung

Die Umsetzung der Tools ist weit fortgeschritten. Das neue Frontend und das in dieser Arbeit beschriebene Backend sind bereits im Live-Betrieb eingesetzt und von Kunden getestet worden. Die Umsetzung der Regressionsanalyse ist bisher prototypisch realisiert. Die REST-Schnittstellen für das Anlegen eines Regressionsmodells und der iterativen Regressionsanalyse ist vollständig umgesetzt. Es werden die spezifizierten Entitäten zurückgegeben. Das bedeutet aus Sicht der Frontendentwickler ist dieser Use Case - Vorgang bereits im finalen Stand. Im Backend ist jedoch noch Refactoring nötig sowie die Umsetzung der Persistierung und Verwaltung der angelegten Regressionsmodelle.

Alle anderen Use Cases konnten vollständig umgesetzt werden und sind bereits mit dem parallel entwickelten Frontend im Live Betrieb.

5.1.1 Frontend EnergyData Toolbox

Das Frontend ist im Entwicklungsstand prototypisch und wird agil weiterentwickelt. Die Grundfunktionalitäten sind implementiert und die Tools sind für die Kunden zugänglich. Die User Experience wird getestet und gemeinsam mit Rückmeldung der Nutzer fortlaufend verbessert. Das Regressions-Tool befindet sich noch in der Entwicklungsphase und ist derzeit nicht für Nutzer verfügbar.

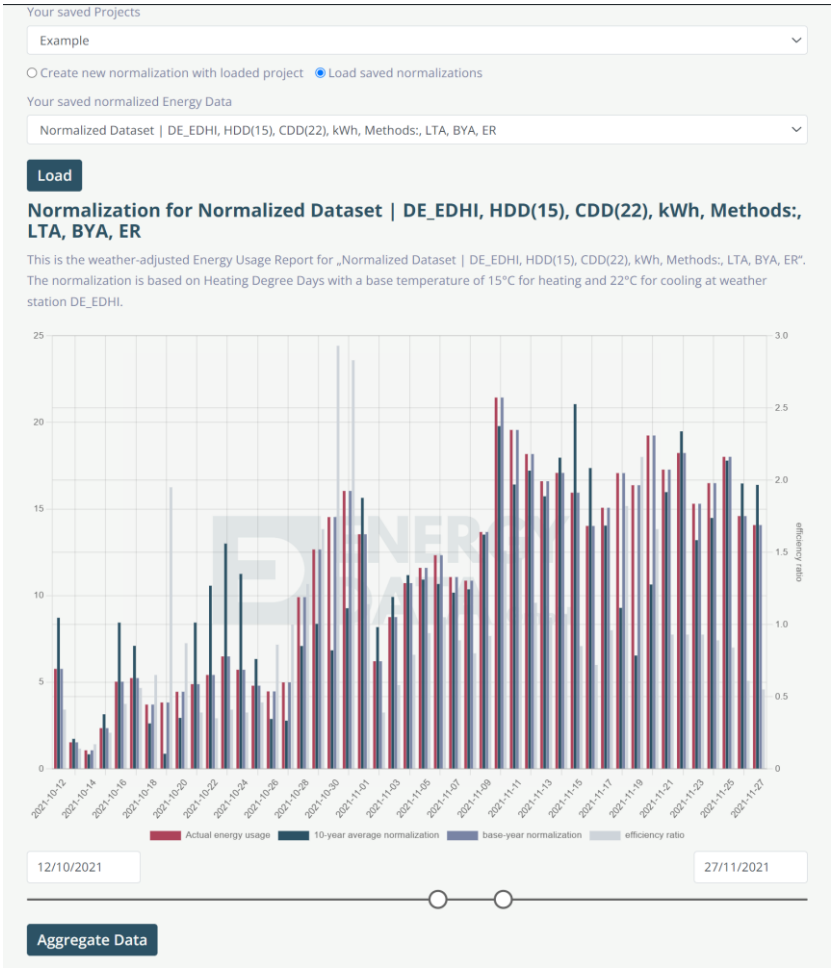

Abbildung 14 Frontend Normalisierungstool (EnergyData GmbH)

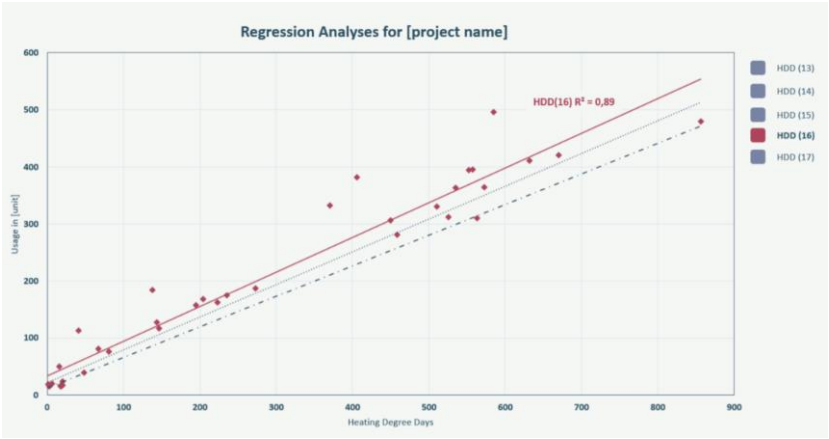

Abbildung 15 Regressionstool Prototyp (EnergyData GmbH)

5.1.2 Use Cases Übersicht

Use Case	***Entwicklungsstand***
Use Case 1 – Hochladen von Energieverbrauchsdaten	Vollständig
Use Case 2 – Formatierung von Rohdatensätzen	Vollständig
Use Case 3 – Herunterladen von Datensätzen in CSV-Format	Vollständig
Use Case 4 – Aggregation zeitlicher Intervalle	Vollständig
Use Case 5 – Normalisierung (Alle Methoden)	Vollständig
Use Case 6 – Lineare Regressionsanalyse	Prototypisch
Use Case 7 – Iterative Lineare Regressionsanalyse	Prototypisch
Use Case 8 – Abruf von Regressionsmodellen	Prototypisch
Use Case 9 – Abruf von DatensätzenUse Case 10 – Verwalten von	Vollständig
Use Case 10 – Verwalten von Datensätzen	Vollständig

Tabelle 11 Entwicklungsstand Use Cases

5.1.3 Nicht-Funktionale Anforderungen Übersicht

Nicht-Funktionale Anforderung	***Erfüllung***
NFA.1 Datenzentriertheit	Erfüllt
NFA.2 REST API Standards	Erfüllt
NFA.3 Performance	Teilweise erfüllt
NFA.4 Dateigrößenlimit	Erfüllt
NFA.5 Erweiterbarkeit & Wiederverwendbarkeit	Teilweise erfüllt

NFA.6 Resilienz	Teilweise erfüllt
NFA.7 Sicherheit	Teilweise erfüllt

Tabelle 12 Nicht-Funktionale Anforderungen Übersicht

5.2 Aufbau der Applikation und Datenbank

Die Use Case Komponenten sind als Django Apps realisiert und bilden die entsprechenden Schichten mit ihren Subkomponenten ab. Die Struktur der Datenbank wird automatisiert durch die Django Models erstellt, welche die Entitäten aus dem fachlichen Datenmodell abbilden. In der Standard Django URL Komponente werden die Routinginformationen einer Komponente eingetragen, die Ressourcenpfade der REST-API definiert und mit den Views verknüpft.

Datenstruktur: Pandas Dataframe

In den Processing-Komponenten kommt insbesondere die Datenverarbeitungs-Bibliothek Pandas zum Einsatz. Die Datensätze werden daher im gesamten System als „Dataframes“ - Datenstruktur verwendet und verarbeitet. Die umfangreichen Methoden der Dataframes-API werden genutzt, um Datensätze zu bereinigen oder gängige Operationen wie Summierungen oder Durchschnittsberechnungen auf den Dataframes durchzuführen.

5.2.1 Subkomponenten

Im Nachfolgenden werden die Implementierungsaspekte der Schichten-spezifischen Subkomponenten erläutert.

Die REST-API wurden unter Benutzung der Bibliothek Django-REST-Framework realisiert. Das Framework ermöglicht es den Django-View mittels Decorator-Pattern zu erweitern und die Einhaltung von REST-Standards zu gewährleisten (Siehe NFA.2 REST API Standards). In den Views werden Requests empfangen und verarbeitet und HttpResponses des Django-Restframeworks zurückgegeben. Ein wichtiger Teil der Implementierung ist hier die Einhal-

tung der Response-Codes des http-Standards und die Einhaltung der NFA.2 REST API Standards.

Die Repository-Subkomponenten sind als einfache Python-Klasse realisiert, welche Methoden zur Verarbeitung der Daten anbietet. Jede Repository-Subkomponente nutzt die Django-Models ihrer Hauptkomponente. Die Django QuerySet API wird eingesetzt, um die Datenbankzugriffe zu realisieren. Die QuerySets werden bei Bedarf in die interne Dataframe-Datenstruktur umgewandelt, um die Weiterverarbeitung zu ermöglichen.

Das Model ist eine Django Klasse und wird von allen Repository-Komponenten benutzt, die ihnen Datenentitäten verarbeiten. Das Django-Model wird verwendet. In dieser Komponente werden die Datentypen und die Relationen des fachlichen Datenmodells abgebildet (Siehe Fachliches Datenmodell).

5.2.2 Regressionsanalyse

Die Realisierung des Regressionstools ist prototypisch. Die REST-API für die Erstellung der Regressionsmodelle (und iterative Erstellung) ist in einem finalen Zustand. Die Entwicklung des Frontends kann daher unabhängig von weiteren Änderungen im Backend voranschreiten.

Für die Erstellung der Regressionsmodelle ist die Scikit-learn Bibliothek zum Einsatz gekommen. Das Trainieren der Modelle wurde in Jupyter-Notebooks mit realen Testdaten geprüft, um Methoden und Parameter zu finden, welche eine hohe Genauigkeit der Modelle erzielen. Die Schnittstellen zur Regressionsanalyse wurden mit verschiedenen Testsdatensätzen mit Messungen von halben Jahren bis zu zehn Jahre getestet. Die trainierten Modelle weisen beim Vorhersagen von Energieverbräuchen auf Basis der abhängigen Gradtagszahlen eine Bestimmtheitsmaß von 0.75 bis 0.9 auf (Siehe Das Bestimmtheitsmaß $\boldsymbol{R^2}$). Die Güte des Modells ist stark abhängig von der Wahl der Basistemperatur. Die iterative Regressionsanalyse kann, durch den Vergleich der erstellten Modelle, bei der Wahl einer geeignete Basistemperatur unterstützen.

5.3 Betrieb des Systems

Die Verteilungssicht bildet den Systemzustand so ab, wie er zum Abschluss dieser Arbeit im Live-Betrieb eingesetzt wird. Die Konfiguration der Proxy Server ist über die statischen Konfigurationsdateien der Nginx und Gunicorn Applikationen eingerichtet und sorgt unter anderem für das korrekte Routing der Aufrufe.

Es wird HTTPS über TLS eingesetzt, um eine sichere Kommunikation der Clients mit dem Server zu ermöglichen. Die Nginx Instanz hört dafür die Ports 80 und 443 ab und leitet alle http Anfragen auf HTTPS um. Die initiale http Anfrage wird mit einem Statuscode 307 (Temporary Redirect) an die HTTPS Version weitergeleitet. Für die Zertifizierung kommt das Open Source Programm „Certbot“ zum Einsatz.

Der Reverse Proxy Server schützt das Django-Backend vor den gängigsten Cyberangriffen, weshalb die Sicherheit das Backends als ausreichend eingestuft werden kann. Durch den Einsatz von HTTPS ist die Verschlüsselung der Aufrufe gesichert.

Die NFA.4 Dateigrößenlimit kann über Nginx geprüft werden und ist daher ebenfalls erfüllt. Anfragen über dem Default Limit werden von Nginx automatisch mit dem Status-Code 413 (Payload Too Large) abgelehnt.

Datenbank Administration

Der Docker Container ist auf dem Ubuntu Server eingerichtet und ist direkt an die Postgres Instanz über TCP/IP verbunden und nutzt den dafür vorgesehen Port 5432 der Postgres Instanz. (Postgres, 2022). Das Routing übernimmt der Nginx Proxy Server. Damit ist die Datenbank Administration über die URL /pgadmin4/ im Live-Betrieb administrierbar.

5.4 Qualitätssicherung

5.4.1 Dokumentation

Die Softwarequalität in Bezug auf die Dokumentation ist sehr hoch und sollte die Einarbeitung von zukünftigen Entwicklern sowie die Wartung des Systems erleichtern. Die REST-API wurde dokumentiert alle Schnittstellen mit technischen Details sin dfür API-Nutzer zu-

gänglich. Der Programmcode wurde zur Förderung der Leserlichkeit bei nicht-trivialen Implementierungen konsistent kommentiert.

5.4.2 Testabdeckung

Die Testabdeckung ist im bisherigen Projektzustand als geringfügig einzuordnen. Es sind Use Case Tests für die Normalisierung implementiert, wobei auch Sonderfälle abgedeckt sind. Die REST-API für Upload und Formatierung sind großflächig getestet mit verschiedenen Sonderfällen. Dennoch besteht bisher keine Testabdeckung innerhalb des Systems, um Funktionen auf ihre Korrektheit zu überprüfen. Die Tests sind derzeit nicht vollständig funktionsfähig und müssen korrigiert werden.

Entwicklungsumgebung und GitHub-Actions

Die Umsetzung der Tests in Verbindung mit GitHub-Actions hat sich als herausfordernd herausgestellt. Zur automatischen Ausführung der Tests bei einem Push auf das Repository müssen Datensätze eingelesen werden. Das Problem besteht, dass die Entwicklungsumgebung Windows war und GitHub Actions auf Ubuntu ausgeführt werden. Im jetzigen Zustand funktionieren die Tests nicht auf beiden Betriebssystemen. Diese Problemstellung wurde erstmal auf einen späteren Zeitpunkt verschoben. In Zukunft sollte eine möglichst automatisierte Lösung mittels Testframeworks gefunden werden, um die Korrektheit von Programmcode zu gewährleisten.

5.4.3 Performance

Die Operationen sind möglichst performant gestaltet worden und die nicht-funktionalen Anforderung NFA.3 Performance zu erfüllen. Dennoch sind einige Operationen über große Datenmengen nicht sehr performant umgesetzt und haben teilweise mehrere Sekunden Antwortzeit.

Ein Beispiel für eine performante Umsetzung ist das Speichern großer Anzahlen von Messdaten. Hier ist Djangos *bulk_create* Funktion zum Einsatz gekommen. Diese Art der Speicherung ist insbesondere bei vielen Dateneinträgen effizient, wie es bei dem Verarbeiten von Energieverbrauchsdatensätzen der Fall ist. Diese Umsetzung hat eine deutlich geringere Ant-

wortzeit, als eine Umsetzung ohne die Verwendung von *bulk_create*. Der Grund ist, dass beim „Bulk Create“ nicht für jeden Eintrag eine neue Datenbank Query genutzt wird (Siehe (Django, 2021, pp. Abschnitt Bulk-Create).

Identifizierte Bottlenecks

Wie bereits in der Anforderungsanalyse festgestellt, besteht von vielen Komponenten eine Abhängigkeit zu dem Degreeday-API Service. Wenn dieser ausfällt, dann können auch Teile der neuen EnergyData-Tools nicht arbeiten.

Die Antwortzeiten der Degreeday-API liegen bei der Abfragen von größeren Zeiträumen und Durchschnittswerten im Bereich von mehreren Sekunden, was auf die Verarbeitung von großen Wetterdatenmengen zurückzuführen ist. Insbesondere bei Operationen, welche wiederholt Aufrufe bei der Degreeday-API machen müssen, beeinträchtigen die Antwortzeiten die User Experience. Die allgemeine Optimierung der Antwortzeit würden Erweiterungen und Veränderungen der Degreeday-API erfordern. Ein Lösungsvorschlag wäre die Erweiterung der Degreeday-API durch eine Operation zur gleichzeitigen Abfrage von mehreren Zeiträumen, um wiederholte Abfragen effizienter gestalten zu können.

5.4.4 Verbesserungspotentiale

Verwaltungsoperationen

Klarheit über die projektartigen Verwaltungen von verschiedenen Arten von Datensätzen ist erst im späteren Verlauf der Anforderungsanalyse entstanden. Daher könnte die Kompaktheit des Systems mit einer Erweiterung einer Projektklasse verbessert werden. Diese könnte Verwaltungs- und Filterungsoperationen kapseln und so Redundanz vermeiden.

Ein Verbesserungsvorschlag ist, die Projektinformationen aus den Entitäten „Rawdataset“, „Normdataset“ und „RegressionAnalysis“ zu entfernen und in einer einzigen Entität zu kapseln. Es müsste anschließend zwischen den verschiedenen Typen von Datensätzen unterschieden werden. Diese Verwaltung der Projekteartigen Struktur würde dann nur noch in einer Komponente liegen.

User Entität und API Key

Derzeit können Benutzer von der Website ihren API Key manuell angeben oder es wird ihnen automatisch von der Authorisierungs Komponente ein API Key zugewiesen. Um nachzuvollziehen, mit welchen API Keys bestimmte Operationen ausgeführt wurden und entsprechende Filterabfragen zu machen, wird der Key derzeit in den Projektentitäten gehalten. Dies bedeutet, dass der API Key derzeit an mehreren Stellen im System gehalten und verarbeitet wird. Es wäre aus Sicherheitsaspekten vorteilhaft, wenn der API Key ausschließlich in einer Entität (z. B. der User Entität) gehalten werden würde. Dadurch würde die Wahrscheinlichkeit verringert, dass bei der Entwicklung Fehler gemacht werden und der API Key nach außen gelangt. Ähnliches gilt für Session, welche in den Projektentitäten gehalten wird. Die Gültigkeit einer Session verfällt nach einer festgelegten Zeit. Der User könnte dann nicht mehr effektiv nach Sessions filtern. Ein Lösungsvorschlag wäre, in der User Entität zusätzlich Listen von Sessions zu halten, mit der Sessions und Projekte in Verbindung gebracht werden. Eine Listenstruktur in der User Entität könnte auch für den API Key umgesetzt werden.

Sicherheitsbeeinträchtigung API Key

Der Ansatz für die Authentifizierung und Autorisierung ist derzeit als bedenkliche Sicherheitslücke einzustufen. Insbesondere die Abfrage von Datensätzen mittels API Key sollte verhindert werden, um die Daten aller Nutzer vor Missbrauch zu schützen.

Im Projektverlauf ist diese Sicherheitslücke aufgefallen. Aufgrund der Authentifizierung der Nutzer durch die WordPress-Instanz und der überschaubaren Anzahl an derzeitigen Nutzern wurde diese Sicherheitslücke vorerst ignoriert. In Zukunft sollten Ansätze gefunden und umgesetzt werden, welche geringe Sicherheitsrisiken darstellen.

6 Fazit

Das Ziel dieser Arbeit war der Entwurf und die prototypische Realisierung eines Data-Service zur Verarbeitung und Regressionsanalyse von Energieverbrauchsdatensätzen. Verschiedene Use Cases aus dem Energiedatenmanagement sind in der Anforderungsanalyse entstanden und in den Softwareentwurf überführt worden. Diese haben sich über den Projektzeitraum agil weiterentwickelt. Es wurde der Ist-Zustand der bestehenden Infrastruktur analysiert und spezielle Anforderungen an das System spezifiert. Des Weiteren wurden zur Qualitätssicherung nicht-funktionale Anforderungen festgelegt. Der Softwareentwurf konkretisierte anschließend die Anforderungen als Bauplan für die Realisierungsphase. Architekturentscheidungen und Technologieauswahl wurden mittels Entwurfsentscheidungen begründet. Es wurden der Entscheidungsprozess und möglichen Vor- und Nachteile der Entscheidungen beleuchtet. Der Architekturentwurf gab Regeln für die Implementierung vor und hat gleichzeitig möglichst wenig Implementierungsaspekte vorgegeben. Die komponentenorientierte Entwurfsweise hat das System in Use Case und ihre Subkomponenten unterteilt, welche systemübergreifende Eigenschaften teilen, um somit eine übersichtliche und erweiterbare Softwarearchitektur zu schaffen. In der Realisierung wurden die Implementierungsaspekte beleuchtet und der Stand der Entwicklung und der Betrieb des Systems beschrieben. Die Erfüllung von funktionalen Anforderungen sowie nicht-funktionalen Anforderungen wurden in tabellarischer Form dargestellt. Für die Qualitätssicherung wurden Aspekte der Testabdeckung, sowie der Performance des Systems beleuchtet und Performance-Bottlenecks identifiziert. Es wurden Vorschläge für die Weiterentwicklung und fortlaufende Verbesserung gemacht.

In der abschließenden Evaluierung soll der Entwurfs- und Realisierungsprozess kritisch beleuchtet werden und Entwurfsentscheidungen hinterfragt werden. Sollten Fehlentscheidungen aufgedeckt worden sein, werden konstruktive Verbesserungsvorschläge gemacht.

6.1 Evaluierung

6.1.1 Softwarequalität vs. Nutzbarkeit

Die Anforderungen an das System sind während der agilen Entwicklung entstanden und haben sich über den Entwicklungszeitraum verändert. Die Klarheit über die Gesamtheit des Systems ist mit dem Voranschreiten des Projektes gewachsen. Während der agilen Sprints wurden Komponentenentwürfe auf Grundlage der ermittelten Anforderungen erstellt. Die groben Architekturentscheidungen wurden zu einem frühen Zeitpunkt festgelegt. Die Unklarheit war jedoch während des ersten Drittels der Entwicklungszeit deutlich und hat Implementierungsentscheidungen negativ beeinflusst und zu einer gewissen Nachlässigkeit geführt.

Trotz des strukturierten Vorgehens ist der Blick auf die Gesamtheit des Systems und der Softwarequalität während der Realisierung nicht stets beachtet worden. Die Entwicklung hatte vorrangig das Ziel, die Realisierung der Schnittstellen und die Abbildung der Use Cases fertigzustellen und für die Frontend-Entwickler zugängig zu machen. Die möglichst schnelle Nutzbarkeit des Systems hatte während der Realisierung eine erhöhte Priorität. Die Softwarequalität in Bezug auf die Einhaltung der Schichtenarchitektur hat unter diesem Umstand gelitten. Der jetzige Projektzustand entspricht daher nicht allen Anforderungen, die der Entwurf an das System stellt.

Es wurden wiederholt Refactoring-Sprints eingelegt, in denen die Kapselung und Verantwortlichkeiten der Komponenten strenger umgesetzt wurden. Die Einhaltung der Regeln zur Trennung der Schichten muss jedoch noch in zukünftigen Sprints nachgezogen werden. In dem jetzigen Projektzustand folgen insbesondere die zu Projektbeginn realisierten Views nicht den Architekturregeln. Die Views haben zu viel Verantwortung und beinhalten Verarbeitungslogik oder greifen teilweise direkt auf die Repository-Schicht zu. In dem Refactoring müssen die Implementierungsdetails aus den Views in Methoden in die Processing-Komponenten verschoben werden. Diese Methoden müssten nach den Use Cases bezeichnet werden und dort abgebildet werden. Anschließend müsste der View so erweitert werden, dass die Antwort der Methodenaufrufe zu dem richtigen Response-Code umgewandelt wird.

Durch die Angleichung von dem Systemzustand an den Entwurf entstehen technische Schulden, die das Frontend nicht betreffen. Die REST-API ist so entworfen und realisiert worden, dass diese unverändert bleibt. Daher ist die Diskrepanz, die derzeit zwischen dem Entwurf und dem Systemzustand besteht, mit relativ geringen Kosten anzugleichen. Es sollte jedoch vor der Projektübergabe passieren, damit bei der Einarbeitung von neuen Entwicklern ein Systemzustand besteht, welches konsistent die festgelegten Architekturentscheidungen befolgt.

6.1.2 Architekturentscheidungen

Die Entwurfsentscheidungen zur Nutzung von Django und die Schichtenarchitektur haben sich als übersichtlich und erweiterbar herausgestellt. Die Use Case Komponenten bildeten klar abgegrenzte Verantwortlichkeiten und stellten in der Realisierung überschaubare Arbeitspakete dar. Der mehrwertschaffende Aspekt des Systems war durch diese Arbeitsweise stets priorisiert und transparent für alle Stakeholder erkennbar.

Die Entscheidung Pandas als zentrale Verarbeitungsbibliothek zu benutzen, hat sich nicht als ausschließlich vorteilhaft herausgestellt. Die Anforderungen an die Performance konnte Pandas nicht voll erfüllen. In Retrospektive hätte die Verarbeitung auch mit numpy-Arrays oder alternativen, effizienteren Datenstrukturen umgesetzt werden können.

Die Pandas-Bibliothek hat den Entwicklungsprozess beschleunigt, insbesondere durch die Funktionen zur Bereinigung und Umformung von Datensätzen. Ein Nachteil entstand durch die unvermeidbare Nutzung von iterativen Veränderungen auf den Dataframes. Im Vergleich zu den vektorisierten Methoden von Pandas ist die iterative Verarbeitung von Dataframes imperformant. Viele Use Cases sind auf iterative Verarbeitung mit verschiedenen Fallunterscheidungen angewiesen. Daher konnte das volle Potential von der Pandas-Bibliothek nicht häufig genutzt werden.

Die Programmiersprache Python und die daraus entstandenen Architekturentscheidungen wurden wegen dem Bezug zur Data-Science gewählt. Es hätte sich als Alternative eine Microservice-Architektur angeboten, bei der die Technologiewahl eines Services nur von dessem Use Case abhängt. Somit wäre nicht das gesamte System durch den Umstand beein-

flusst, dass z.B. für das Erstellen von Regressionsmodellen häufig Python-Bibliotheken verwendet.

Insgesamt sind große Teile der geplanten Features umgesetzt und bereits im Live-Betrieb getestet worden. Das Systems ist zuverlässig im Betrieb und die Wartungsvorgänge sind umfangreich dokumentiert. Die nicht erfüllten Anforderungen an das System werden in zukünftige Iterationen angeglichen. Abschließend ist festzustellen, dass die Definition of Done erfüllt ist und die Übergabe des Projektes für die zukünftige Erweiterung erfolgen kann.

Literaturverzeichnis

Allan Cooper, R. R. a. D. C., 2007. *About face 3 : the essentials of interaction design.* https://katalog.haw-hamburg.de/vufind/Record/523919956 Hrsg. Indianapolis: Wiley.

Bloch, J., 2006. How to Design a Good API and Why it Matters. *Proc. 21st ACM SIGPLAN Conference (OOPSLA)*, Issue http://portal.acm.org/citation.cfm?id=1176617.1176622, pp. 506--507.

Django, 2021. *Django Documentation - Bulk Create.* [Online]
Available at: https://docs.djangoproject.com/en/4.1/ref/models/querysets/#bulk-create
[Zugriff am 01 10 2022].

Django, 2022. *Django Project.* [Online]
Available at: https://www.djangoproject.com/start/overview/
[Zugriff am 03 09 2022].

Doty, W. C. T. &. S., 2007. *Energy management handbook.* 6th Edition, ISBN: 0-88173-542-6 (print) — 0-88173-543-4 (electronic) ed. 700 Indian TrailLilburn, GA 30047: The Fairmont Press, Inc..

Energy Data GmbH, 2022. *Energy Data GmbH Website.* [Online]
Available at: https://energy-data.io/about-us/
[Zugriff am 7 11 2022].

Fredrich, T., 2013. *REST API Tutorial.* [Online]
Available at:
https://github.com/tfredrich/RestApiTutorial.com/raw/master/media/RESTful%20Best%20Pr

actices-v1_2.pdf
[Zugriff am 01 07 2022].

Gang of Four, E. G. J. V. R. J. R. H., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* 1 Hrsg. s.l.:Addison-Wesley Professional.

GEFMA, 2020. *Energiemanagement.* Bonn: Energie, German Facility Management Association - GEFMA-Arbeitskreis.

Geilhausen, M., 2020. *Kompakter Leitfaden für Energiemanager - Energiemanagementsysteme nach DIN EN ISO 50001:2018.* https://doi.org/10.1007/978-3-658-28853-2 Hrsg. s.l.:Springer Fachmedien Wiesbaden GmbH.

Grus, J., 2020. *Einführung in Data Science.* 2. Auflage, Deutsche Übersetzung Hrsg. s.l.:O'Reilly.

James, G., 2013. *An Introduction to Statistical Learning..* https://doi.org/10.1007%2F978-1-4614-7138-7 Hrsg. New York: Springer New York.

Petermann, J. P. E., 2018. Erfolgreiches Energiemanagement im Betrieb. In: *Erfolgreiches Energiemanagement im Betrieb.* Wiesbaden: Springer Vieweg, p. 192.

Postgres, 2022. *Postgres Dokumentation.* [Online]
Available at:
https://www.postgresql.org/docs/current/protocol.html#:~:text=PostgreSQL%20uses%20a%20message%2Dbased,also%20over%20Unix%2Ddomain%20sockets
[Zugriff am 01 09 2022].

Publications Office of the European Union, 9.7.2021. *European Climate Law Regulation (EU) 2021/1119.* PE/27/2021/REV/1 https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32021R1119&from=EN Hrsg. s.l.:s.n.

Rossum, G. v., 2022. *PEP 8 – Style Guide for Python Code.* [Online]
Available at: https://peps.python.org/pep-0008/
[Zugriff am 01 07 2022].

Simone Brugger-Gebhardt, G. J., 2019. *Die DIN EN ISO 50001:2018 verstehen.* https://doi.org/10.1007/978-3-658-26266-2 Hrsg. Wiesbaden: Springer Gabler.

Sommerville, I., 2018. Software Engineering. In: *Software Engineering 10., aktualisierte Auflage.* s.l.:Pearson.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original