Fachhochschule Bielefeld

Campus Minden

Informatik

Projektbericht in dem Modul Webengineering:

**SmartMonitoring**

Webportal für PV-Kennlinien

Eingereicht von:

|  |  |  |  |
| --- | --- | --- | --- |
| Hannes Rüffer  Matrikelnummer: 1151954  Fachsemester: 5 | Michelle Vorwerk  Matrikelnummer: 1234567  Fachsemester: 5 | Moritz Pit Withöft  Matrikelnummer: 1151206  Fachsemester: 5 | Christiane Zolkin  Matrikelnummer: 1151404  Fachsemester: 5 |

Abgabedatum: 28.01.2020

Gutachter:

Prof. Dr.-Ing. Grit Behrens

Cem Basoglu

Florian Fehring

Inhaltsverzeichnis

[1 Motivation und Themenbeschreibung 1](#_Toc31050297)

[1.1 Motivation 1](#_Toc31050298)

[1.2 Themenbeschreibung 1](#_Toc31050299)

[2 Vorstellung der Gruppenmitglieder 1](#_Toc31050300)

[2.1 Vorstellung der Gruppenmitglieder 1](#_Toc31050301)

[2.2 Aufgabenverteilung im Team 1](#_Toc31050302)

[3 Theoretische Grundlagen 2](#_Toc31050303)

[3.1 Photovoltaik-Zelle 2](#_Toc31050304)

[3.2 MPP – Maximum Power Point 4](#_Toc31050305)

[3.3 UL – Leerlaufspannung 4](#_Toc31050306)

[3.4 IK – Kurzschlussstrom 5](#_Toc31050307)

[3.5 Kennlinien 5](#_Toc31050308)

[3.6 Hellkennlinien 6](#_Toc31050309)

[3.7 Dunkelkennlinien 7](#_Toc31050310)

[3.8 Nutzen des Modulportals 8](#_Toc31050311)

[4 Konzeptionelle Arbeiten 8](#_Toc31050312)

[5 Code-Implementierungsdetails 8](#_Toc31050313)

[6 Ausblick und Fazit 9](#_Toc31050314)

[6.1 Ausblick 9](#_Toc31050315)

[6.2 Fazit 9](#_Toc31050316)

[7 Installationshinweise und Benutzerhandbuch 9](#_Toc31050317)

[7.1 Installationshinweise 9](#_Toc31050318)

[7.2 Benutzerhandbuch 9](#_Toc31050319)

[8 Quellenverzeichnis 9](#_Toc31050320)

[9 Anlagen 9](#_Toc31050321)

# Motivation und Themenbeschreibung

Moritz

## Motivation

## Themenbeschreibung

# Vorstellung der Gruppenmitglieder

Christiane

## Vorstellung der Gruppenmitglieder

Hier noch nicht auf deren Aufgaben eingehen, sondern allgemein vorstellen (z.B. seit wann an der FH, Interessen im Bereich Informatik, Stärken etc.)

## Aufgabenverteilung im Team

Hier die Gruppenmitglieder auf das Projekt beziehen und deren Rollen im Projekt erläutern.

# Theoretische Grundlagen

Unser Projekt ist ein Webportal zum Anzeigen und Vergleichen von Photovoltaik-Modulen und deren Daten. Um zu verstehen, wofür die Abkürzungen und Grafiken, die in unserem Portal überall wiederzufinden sind, stehen, und was diese bedeuten, soll in diesem Abschnitt erläutert werden.

## Photovoltaik-Zelle

Es gibt viele verschiedene Arten von Photovoltaik-Zellen, zum Beispiel

* Monokristallines Silizium,
* Polykristallines Silizium (Si),
* Amorphes Silizium (a-Si),
* Kadmium-Tellurid (CdTe) und
* Kupfer-Indium-(Gallium-)Diselenid (CIS/CIGS).

Das am weitesten verbreitete und bekannteste Material ist Silizium, da es vergleichsweise günstig, einfach zu bekommen und einfach zu verarbeiten ist. Silizium hat eine besondere Eigenschaft, welche es zu dem zentralen Bestandteil einer Photovoltaik-Zelle macht: Es ist ein Halbleiter. In Halbleitern können durch die Zufuhr von Energie, zum Beispiel in Form von energiereichem Licht oder elektromagnetischer Strahlung, in ihrem Inneren freie Ladungsträger erzeugen. Das ist allerdings nicht ohne einen weiteren kleinen Trick möglich.

Eine Silizium-Photovoltaik-Zelle hat mehrere Schichten. Die obere Schicht ist mit einem Elektronenspender (zum Beispiel Phosphoratomen) durchsetzt, während die untere Schicht mit einem Elektronenakzeptor (zum Beispiel Boratomen) durchsetzt ist. Die obere Schicht hat somit das Verlangen, Elektronen abzugeben, während die untere Schicht Elektronen aufnehmen möchte. Durch diese Diskrepanz entsteht ein ständig vorhandenes elektrische Feld. Bis zu diesem Punkt findet der Prozess auch ohne jegliche Sonneneinstrahlung statt, aber nur durch das vorhandene elektrische Feld kann noch kein Strom gewonnen werden. Trifft nun Sonnenlicht (oder anderes energiereiches Licht) auf die Photovoltaik-Zelle, übertragen die Lichtphotonen ihre Energie an die locker gebundenen Elektronen in der Schicht des Elektronenakzeptors. Die getroffenen Elektronen lösen sich aus ihrer Bindung und begeben sich in das Leitungsband, eine Schicht zwischen den beiden Siliziumschichten. Viele der Elektronen verschwinden nach kurzer Zeit wieder, da sie sich mit immer noch vorhanden Teilchen rekombinieren. Einige Elektronen werden durch das vorhandene elektrische Feld auf die gegenüberliegende Schicht gedrängt, während die „Löcher“, dort, wo die Elektronen herkamen, leer bleiben. Es entsteht ein Ladungsunterschied, eine Spannung. Solange weitere Lichtphotonen kontinuierlich weitere Elektronen aus ihren Bindungen lösen, entsteht ein nutzbarer Strom, der über Leiterbahnen an der oberen und unteren Siliziumschicht wieder auf die ursprüngliche Seite geführt wird.

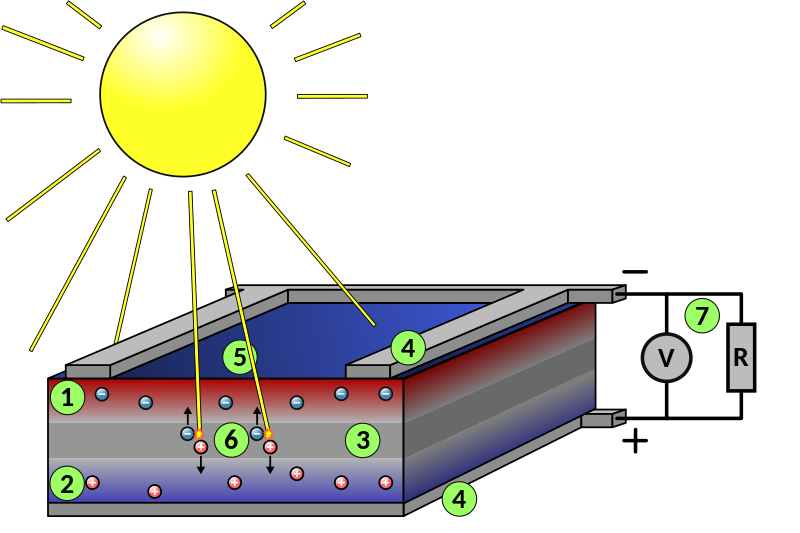


Abbildung : Aufbau einer Photovoltaik-Zelle

## MPP – Maximum Power Point

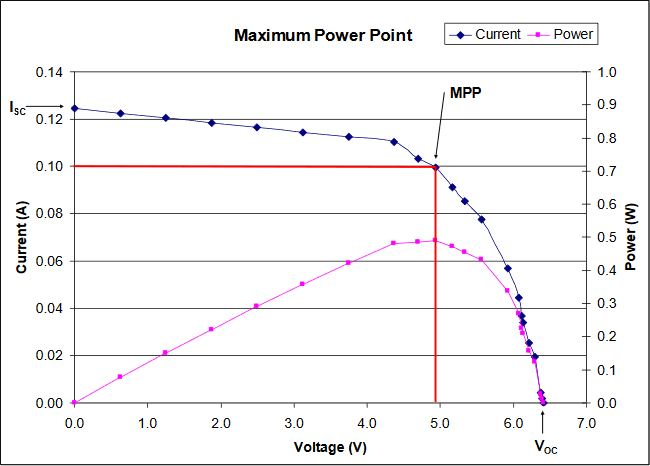
Der Punkt der maximalen Leistung (eng. „Maximum Power Point“, kurz „MPP“) ist der Punkt, an dem die Leistung einer Photovoltaikanlage unter den Standard-Testbedingungen (d.h. 25°C Modultemperatur, 1000 W/m² Bestrahlungsstärke und einer Luftmasse von 1,5 AM) maximal wird. Durch systematisches Ausprobieren von verschiedenen Widerständen auf der Leiterbahn des Verbrauchers, kann eine Folge von Punkten gemessen werden. Diese Punkte bestehen aus einer Spannung und einer Stromstärke. Da die Leistung in Watt gleich der Spannung mal die Stromstärke ist, kann für jeden der eingesetzten Widerstände die maximal mögliche Leistung berechnet werden. Der Punkt, an dem die Leistung am höchste ist, ist der MPP – der Punkt der maximalen Leistung. Zeichnet man die Folge der Punkte in ein Diagramm und verbindet die Punkte, so erhält man den geometrischen Zusammenhang: Der Punkt der maximalen Leistung ist der Punkt, an dem die Fläche unter dem Graphen (der blaue Graph in **Abbildung 2**) maximal wird. Zeichnet man das Produkt aus Spannung und Stromstärke zusätzlich, so kann man den MPP direkt ablesen, da dieser dann der Hochpunkt des neuen Graphen (pink in **Abbildung 2**) ist.

Abbildung : Kennlinie zum MPP

## UL – Leerlaufspannung

Die Leerlaufspannung UL, im Englischen abgekürzt als VOC („open circuit voltage“), wird dann erreicht, wenn keine externe Verbindung zwischen den Polen der Photovoltaik-Zelle angeschlossen ist, und die Photovoltaik-Zelle trotzdem von der Sonne bestrahlt wird. An diesem Punkt ist die höchste Spannung der Photovoltaik-Zelle erreicht, allerdings fließt dann auch kein Strom mehr. Ohne Verbindung zwischen den Polen kann kein Strom fließen, jedoch sorgt die Sonneneinstrahlung weiterhin dafür, dass Elektronen in der Siliziumschicht ihre Bindungen lösen. Da die Elektronen keinen anderen Weg mehr gehen können, um das entstandene Potenzial auszugleichen, ist die höchstmögliche Spannung erreicht.

In der **Abbildung 2** liegt die Leerlaufspannung VOC bei ca. Voltage = 6,4V, am rechten, unteren Rand des Diagramms.

## IK – Kurzschlussstrom

Der Kurzschlussstrom IK, im Englischen abgekürzt als ISC („short circuit current“), wird dann erreicht, wenn eine widerstandslose externe Verbindung zwischen den Polen der Photovoltaik-Zelle besteht. An diesem Punkt fließt der größtmögliche Strom, allerdings existiert keine Spannung mehr. Wenn kein Widerstand zwischen die Pole geschaltet ist, führt das dazu, dass jegliche Spannung, die erzeugt wird, sofortig die gegenüberliegende Schicht erreicht.

In **Abbildung 2** liegt der Kurzschlussstrom ISC bei ca. Current = 1,25A, am linken, oberen Rand des Diagramms.

## Kennlinien

Eine Kennlinie ist eine Abbildung von der Spannung, die in einer Photovoltaik-Zelle herrscht, zu der Stromstärke, die die Photovoltaik-Zelle erzeugt. Wie in 3.2 schon angesprochen, kann man durch das kontinuierliche Ändern des Widerstandes, der an der verbindenden Leiterbahn zwischen den zwei Polen liegt, eine Änderung der Spannung sowie der Stromstärke messen. Fängt man also mit kaum einem Widerstand an, erhält man einen Messpunkt nahe des Kurzschlussstroms IK. Anschließend wiederholt man die Messung mit immer größeren Widerständen und merkt sich auch diese Messpunkte. Hat man dann einen zu hohen Widerstand erreicht, fängt die Photovoltaik-Zelle an, rapide an Stromstärke zu verlieren (ungefähr um den MPP herum). Je höher daraufhin der Widerstand wird, desto näher kommt man der Leerlaufspannung UL, da ein zu hoher Widerstand den gleichen Effekt hat, wie als hätte man überhaupt keine Verbindung zwischen den Polen aufgebaut: Die Elektronen nehmen den Weg mit dem geringsten Widerstand, welcher dann eher durch das Innere der Photovoltaik-Zelle verläuft, als durch die Luft oder eine Verbindung mit zu hohem Widerstand.

Alle Messwerte, die während der gesamten Zeit aufgezeichnet wurden, ergeben dann die Kennlinie. Es gibt allerdings auch unter den Kennlinien verschiedene Arten: Hellkennlinien und Dunkelkennlinien.

## Hellkennlinien

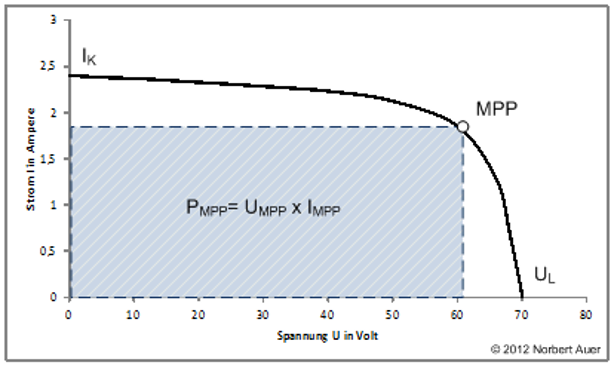


Abbildung : Eine Hellkennlinie

Abgebildet in **Abbildung 3** sieht man eine exemplarische Hellkennlinie. Sie enthält alle wichtigen Punkte IK, MPP und UL.

Der größte Unterschied zwischen den Kennlinienarten ist, wie sie gemessen werden. Eine Hellkennlinie wird unter den Standard-Bedingungen (25°C Modultemperatur, 1000 W/m² Sonneneinstrahlung, 1,5 AM) gemessen. Diese Bedingungen zu erreichen ist nicht immer einfach, da diese sehr spezifisch und abhängig vom lokalen Wetter sind. In manchen Ländern der Erde mag es sogar auf natürlichen Weg nicht möglich sein, die Standard-Bedingungen einzuhalten. Angefertigt werden Hellkennlinien vom Hersteller selbst immer nach der Produktion von einer Photovoltaik-Zelle, am besten sollten die Messungen aber in gewissen Zeitabständen von dem Endnutzer wiederholt werden. Hellkennlinien bieten nämlich neben der Möglichkeit, die maximale Leistung leicht ablesen zu können, auch noch die Möglichkeit, gewisse Mängel an den Photovoltaik-Zellen festzustellen. So kann zum Beispiel ein hoher gemessener serieller Widerstand bedeuten, dass eine Steckverbindung fehlerhaft ist. Ein zu geringer Widerstand hingegen kann heißen, dass lokale Kurzschlüsse existieren. Solche Mängel sollten schnell behoben werden, denn sie wirken sich unvermeidbar negativ auf die Gesamtleistung einzelner Photovoltaik-Zellen, und gegebenenfalls dann auch auf ganze Stränge aus.

## Dunkelkennlinien

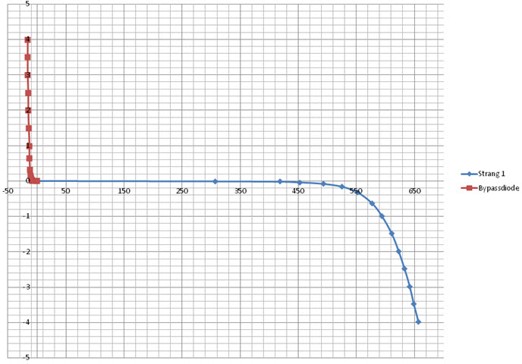
Im Gegensatz zu Hellkennlinien werden Dunkelkennlinien nicht unter den Standard-Bedingungen aufgenommen. Analog zu ihrem Namen werden Dunkelkennlinien nur bei einer Sonneneinstrahlung von 0 W/m² gemessen. Sie sind sehr beliebt, da die Bedingungen für die Aufzeichnung vereinfacht sind – nicht mehr wetter-, ort- und landabhängig. Tatsächlich bieten Dunkelkennlinien auch manche Vorteile, die Hellkennlinien bieten: Durch eine Transformation der Dunkelkennlinie entlang der Y-Achse soweit, bis die Leerlaufspannung UL bei Stromstärke A = 0 liegt, erhält man wieder eine Hellkennlinie. So kann auch von einer Dunkelkennlinie der Punkt der maximalen Leistung abgelesen werden.

Abbildung : Eine Dunkelkennlinie

Auch in der Fehleranalyse sind Dunkelkennlinien beliebt. Durch das Heranziehen von Elektrolumineszenzbildern von den Strängen, können zusammen mit der Dunkelkennlinie schnell fehlerhafte Stränge identifiziert werden. In gewissen Fällen kann durch ein Elektrolumineszenzbild auf einen Blick gesehen werden, wo das Problem des Strangs ist. Aber auf eine Hellkennlinie kann man trotzdem nicht immer verzichten. Ist zum Beispiel das Aufnehmen der Dunkelkennlinie nicht möglich, aber die Bypassdiode kann gemessen werden, liegt eine Unterbrechung zwischen den Photovoltaik-Zellen oder zwischen der Anschlussdiode und Photovoltaik-Zellen vor. In dem Fall muss man am Tag und unter den Standard-Bedingungen nochmal eine Messung durchführen, um herauszufinden, wo genau die Unterbrechung liegt.

Abbildung : Ein Elektrolumineszenzbild

## Nutzen des Modulportals

Das Modulportal bietet neben der großartigen Übersicht und Navigierbarkeit für erfahrene Nutzer und Photovoltaik-Enthusiasten auch eine Hilfestellung für Neu- und Noch-Nicht-Kunden. So gibt es in der Übersicht einen Bereich für Module nach ihrem Beliebtheitsgrad, wodurch Neukunden sich gezielt nach guten Modulen erkundigen können. Außerdem gibt es einen Bereich für die am meisten kommentierten Module. Diese sind sehenswert, da eine rege Diskussion höchst wahrscheinlich gute sowie schlechte Aspekte von Photovoltaik-Modulen aufdecken wird. Durch den zeitlichen Filter von Kennlinien kann auch der Performanceabfall von einzelnen Modulen betrachtet werden, wodurch Neukunden ein Modul schon aussortieren können, falls es bei einer Vielzahl von Leuten schnell an Leistung verliert. Natürlich können jegliche Benutzer auch Fragen unter Module schreiben, die von erfahrenen Benutzern beantwortet werden können.

Insgesamt bietet das Modulportal einen großen Nutzen für erfahrene Benutzer, die ihre Erfahrungen dokumentieren, austauschen und verewigen wollen, aber auch für neue Benutzer, die nur nach Radschlägen und Hilfe suchen.

# Konzeptionelle Arbeiten

## Kommentare: Klassendiagramm

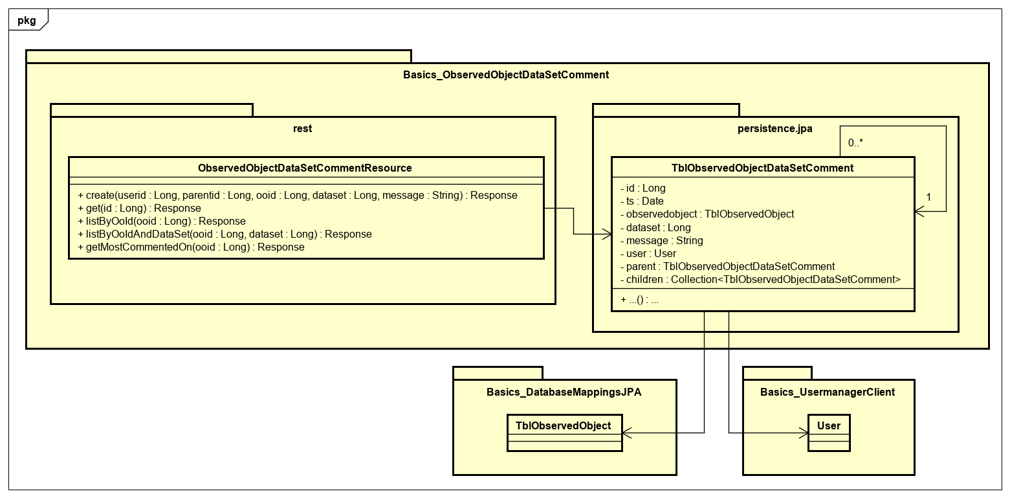


Abbildung : Klassendiagramm für die Kommentare

Die ursprüngliche Planung für die Kommentarfunktion im Backend sah so aus, dass ein neues Projekt erstellt wird, in dem es jeweils ein Paket für die REST-Schnittstelle und die JPA-Klasse gibt. Die Idee war, an den Trend der Microservices anzuknöpfen, und so ein eigenständiges Projekt zu haben, welches so unabhängig wie möglich von den anderen Projekten ist. So sieht man in der obigen Abbildung, dass nur zwei Referenzen benötigt werden, um das Projekt zu realisieren: Eine Referenz auf die Speicherklasse für die ObservedObjects und eine auf die Benutzer Klasse.

So kann die Speicherklasse der Kommentare alle Informationen erreichen, die benötigt werden: „ts“ als Zeitstempel, zu dem der Kommentar angelegt wird, „observedobject“ als Referenz auf das entsprechende ObservedObject, „dataset“ als optionaler Zusatz, zu welchem Datensatz der Kommentar gehört, „message“ als Nachricht des Kommentars, „user“ als Referenz auf das User Objekt, von welchem aus der Kommentar angelegt wurde, „parent“ als Referenz auf einen weiteren Kommentar, um eine hierarchische Abbildung der Kommentare zu schaffen, „children“ als Gegenstück zu der „parent“-Referenz.  
Die REST-Schnittstelle für die Kommentare beinhaltet dann die Standardfunktionalität „create“ und „get“ (welche in der Implementierung noch um „list“ und „delete“ erweitert wurden), sowie diverse Schnittstellen, um spezifische Gruppen von Kommentaren zu laden.

In der Umsetzung von dem Diagramm hat es leider ein paar unlösbare Probleme gegeben: Die REST-Schnittstelle benötigt eine „PersistenceUnit“ um von außen erreichbar zu sein, jedoch kann keine neue „PersistenceUnit“ für die Kommentare angelegt werden, da man dann keinerlei Zugriff mehr auf jegliche vorhandenen Objekte hätte, das heißt auch keinen Zugriff auf ObservedObjects, die aber benötigt werden. Dazu kam noch, dass man auch das Hauptprojekt „SmartMonitoringBackend“ in dem neuen Projekt nicht verlinken konnte, da sonst eine „zyklische Referenz“ entstände, die allerdings von Netbeans verboten ist. Man kann aber das Hauptprojekt auch nicht **nicht** verlinken, weil das Projekt für die Kommentare sonst nicht zur Runtime des Servers verfügbar ist.

Gelöst wurde das Problem auf eine weniger schöne, aber funktionierende Art, indem die REST-Schnittstelle in der Hauptprojekt bewegt wurde. Die JPA-Klasse konnte in dem neuen Projekt verbleiben. So gab es keine zyklische Referenz mehr und der Server hat zur Runtime alle Resourcen und Klassen gefunden.

# Code-Implementierungsdetails

## Suche: Backend

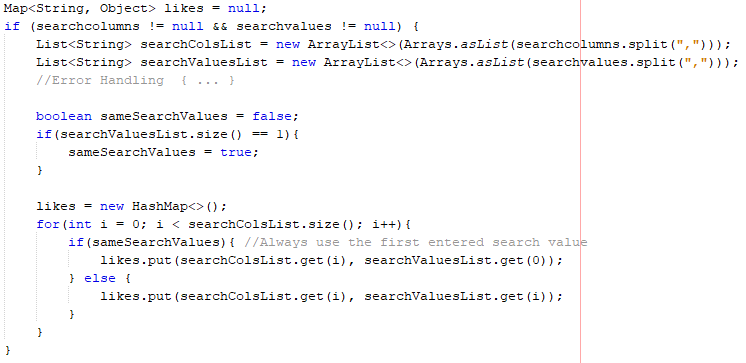
Die Suchfunktionalität ist im Backend in zwei Schritten implementiert: Parsen der Anfrage und Korrektes einfügen in die Datenbankabfrage.

Abbildung : Codeausschnitt: Parsen der Suchanfrage

Abbildung 6 ist ein verkürzter Ausschnitt der Methode „getSets(…)“ der Klasse „DataResource“. Beim Parsen der Anfrage ist zu beachten, dass die Suche auf zwei Parameter aufgeteilt ist. Der erste Parameter „searchcolumns“ ist eine kommaseparierte Folge von Spaltennamen, in denen man Suchen möchte. Der zweite Parameter „searchvalues“ ist entweder ein einzelner Wert (d.h. er enthält keine Kommata), nach welchem dann in allen Spalten gesucht wird, oder auch eine kommaseparierte Folge von Suchbegriffen. In dem letzteren Fall muss die Anzahl der Spaltennamen gleich der Anzahl der Suchbegriffe sein, ansonsten gibt die REST-Schnittstelle einen Fehler zurück. Anschließend werden die Suchspalten und -begriffe in einer „HashMap<String, String>“ einander zuegordnet.

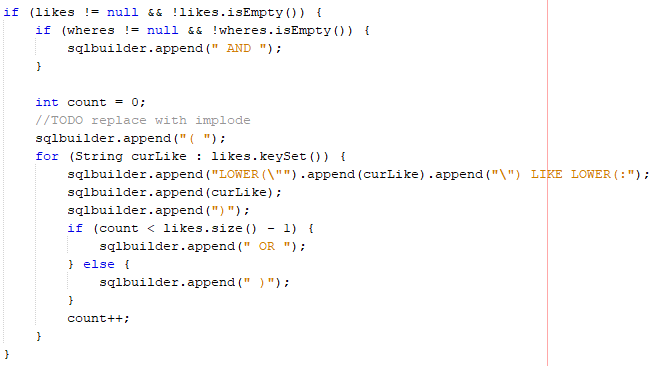


Abbildung : Codeausschnitt: Einfügen der Suchbegriffe in die PreparedQuery

Ist das Parsen der Anfrage erfolgreich, wird die Anfrage an die „DynamicTable“ des angefragten ObservedObjects weitergeleitet. **Abbildung 7** zeigt den Ausschnitt der Methode „getPreparedQuery(…)“ der Klasse „DynamicTable“, in dem die Suchparameter in den SQL-String geschrieben werden. Es wird für jeden Eintrag der Such-„HashMap“ lediglich der Spaltenname in dieser Form eingesetzt: „LOWER(<spaltenname>) LIKE LOWER(:<spaltenname>)“. Der erste <spaltenname> wird aufgelöst zu dem tatsächlichen Spaltennamen, während der zweite <spaltenname> zusammen mit dem Doppelpunkt ein Platzhalter für die Bindung zu dem Wert zu einem späteren Zeitpunkt ist. Beide Spaltennamenersetzungen sind von einem „LOWER()“ umschlossen, das heißt, dass die Suche sowohl bei der Spalte als auch bei dem Wert die Groß- und Kleinschreibung nicht berücksichtigt. Dazwischen steht ein „LIKE“, was in SQL-Sprachen für einen Ähnlichkeitsvergleich mit möglichen „Wildcards“ steht. Nach jedem, außer dem letzen, Eintrag in der Such-„HashMap“ wird ein „OR“ angehangen, was dazu führt, dass ein Datensatz gefunden wird, wenn auch nur eine Spalte aus der angefragten Suche den entsprechenden Suchbegriff beinhaltet.  
Es wurde überlegt, ob man die Suche dynamischer gestalten sollte, das heißt die „LOWER(…)“ abschaltbar, bzw. das „OR“ durch ein „AND“ tauschbar macht, was in der Theorie auch nicht zu schwer zu realisieren gewesen wäre, allerdings haben wir uns dagegen entschieden, weil jede weitere Option als Parameter mit an die Funktion übergeben werden muss. Da die Funktion bereits 11 Parameter hat, würden zwei weitere Parameter, die bei den meisten Aufrufen der Methode sowieso „null“ sind, der Übersichtlichkeit, und damit auch der Wartbarkeit und Erweiterbarkeit des Programms schaden.

Die „Wildcards“ bei dem Ähnlichkeitsvergleich mit dem „LIKE“ Schlüsselwort in SQL haben unvorhergesehene Probleme ausgelöst. Es gibt die zwei Arten von „Wildcards“: Ein „.“ (Punkt) steht für ein einzelnes beliebiges Zeichen, ein „%“ (Prozent) für eine beliebig lange Folge von Zeichen, auch keine. Somit könnte auf die Suche „D%“ in der Spalte „land“ die Antworten „Deutschland“, „Dänemark“ oder auch einfach nur „D“ zurückkommen. („D“ ist kein Land, aber es würde von dem Suchbegriff gefunden werden.) Problematisch wird es dadurch, dass die Suche über die GET-Parameter der Anfrage gehandhabt wird und GET-Parameter von allen modernen Browsern von jeglichen nicht-ASCII-Zeichen bereinigt werden. Anstelle der nicht-ASCII-Zeichen ersetzt ein Browser eine Kodierung beginnend mit einem Prozentzeichen, gefolgt von einer Zahl, die für das entsprechende Zeichen steht. Das Zeichen „Ü“ würde in dieser Kodierung die Zeichenfolge „%C3%9C“ haben. Wenn der Benutzer aber nach einem Modul sucht, wird sein Suchbegriff von Prozentzeichen umschlossen. Der Suchbegriff „C3%9C“ würde also umgeformt werden zu „%C3%9C%“. Auf der Serverseite kann diese Kodierung wieder rückgängig gemacht werden, allerdings kann der Server nicht unterscheiden, ob der Browser oder der Benutzer ein Zeichen kodiert hat. Der Server dekodiert alle Sonderzeichen, die er findet. Das führt dazu, das gewisse Zeichen auf der Serverseite dekodiert werden, die dazu führen, dass der Server abstürzen kann. Um das zu umgehen, wird sowohl im Frontend als auch im Backend das „Wildcard“ Zeichen durch das Paragraphenzeichen ersetzt, und nach der Übertragung wieder zu dem Prozentzeichen zurückersetzt.

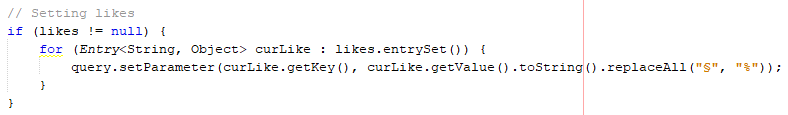


Abbildung : Codeausschnitt der Wildcard-Ersetzung im Backend

## View-Requests: Backend

Die größten Teile des Backends für die View-Requests ist simpel: Eine Speicherklasse für die Daten, eine Schnittstelle für die Standardoperationen und der Rest wird von JAX-RS und Hibernate geregelt. Zwei Problemstellen gab es allerdings, die erste davon bei der Abfrage, welcher Datensatz eines ObservedObjects am meisten angeschaut wurde. Hibernates Methoden um Queries zu erzeugen können nur typisiert sein, wenn genau **ein** Typ zurückgegeben wird, sei es ein primitiver Typ oder eine Klasse. In diesem Fall müssen allerdings zwei Werte zurückgegeben werden: Die Datensatz ID, nach der **gruppiert** wird, und die Anzahl der Besuche, nach der **sortiert** wird.

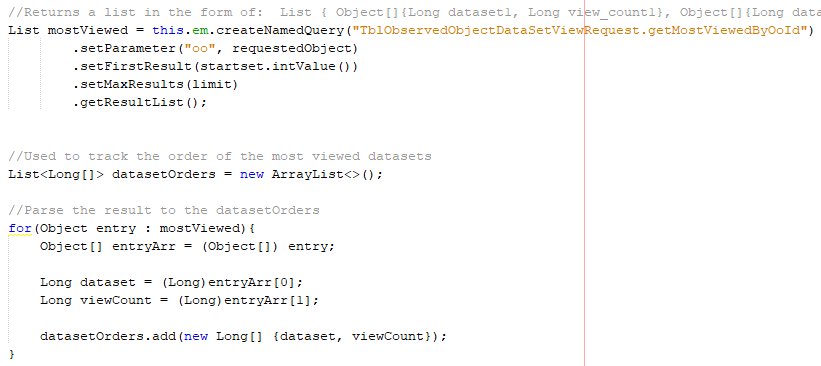


Abbildung : Codeausschnitt: Abfrage der meist besuchten Module aus der Datenbank

Um das zu realisieren, muss man ein nicht-generische Query erstellen und ausführen. Dadurch bekommt man eine nicht-generische Liste zurück, über die man manuell iterieren und die einzelnen Werte von „Object“ zu, in diesem Fall, „Long“ parsen muss. Ein unschöner Weg, aber leider das einzige, was Hibernate in diesem Fall anbietet.

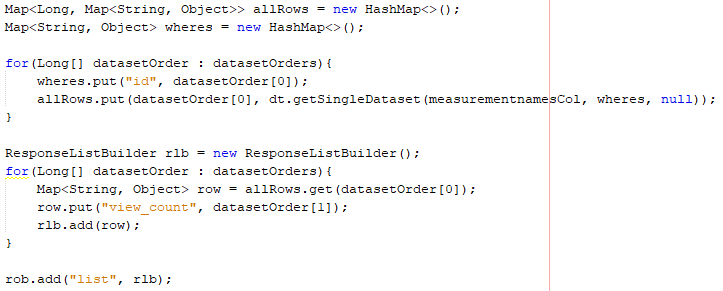
Die zweite Problematik bestand darin, dass mehrere Datensätze, die nicht unbedingt hintereinander liegen, zurückgegeben werden müssen. Würden diese hintereinander liegen, könne man durch ein gut gewähltes „OFFSET“ und „LIMIT“ seine Datensätze bekommen, das findet hier aber keine Anwendung. Man müsste also in die SQL-Abfrage in die „WHERE“ Bedingung die IDs der Datensätze mit aufnehmen, zum Beispiel so: „… WHERE id = ‚1‘ OR id = ‚40‘ OR id = ‚19001‘ OR …“. Allerdings ist die “WHERE” Bedingung in der Schnittstelle der “DynamicTable“ bereits implementiert und die „WHERE“ Bedingungen werden mit „AND“ getrennt, nicht mit „OR“. Dadurch kann man immer nach maximal einem Datensatz gleichzeitig fragen, was in unserer Umsetzung auch genau so passiert ist.

Abbildung : Codeausschnitt: Einzelabfragen aller gefundenen Datensätze

## Media: Backend

Eine Teilaufgabe des Backends war es, die bereits vorhandene Media-Schnittstelle umzuschreiben. Die vorherige Media-Schnittstelle hat eine „1:n“-Beziehung implementiert, diese sollte nun zu einer „n:m“-Beziehung umgeschrieben werden. Die Kritik an der alten Version war, dass wenn man das gleiche Bild zu verschiedenen ObservedObjects hochladen möchte, dieses auch tatsächlich zweimal in der Datenbank ablegen muss. Das führt dazu, dass unnötig Ressourcen von dem Server verbraucht werden. Die neue „n:m“-Beziehung löst dieses Problem, indem beliebig viele Verbindungen zwischen einzelnen Medien und ObservedObjects angelegt werden können, ohne doppelten Platz zu verbrauchen.

Dazu kommt, dass das Modulportal eine neue Anforderung an Medien hatte: Es sollte möglich sein, zu einem Bild nicht nur ein ObservedObject, sondern auch eine Datensatz ID hinterlegen zu können. Ein Modultyp in unserer Datenbank ist lediglich ein Datensatz eines ObservedObjects, jedoch sollen verschiedene Modultypen auch verschiedene Bilder haben können. Umgesetzt wurde das dadurch, dass sowohl in der „TblMedia“-Klasse, als auch in der „TblObservedObject“-Klasse die Referenzen auf die jeweils andere Ressource entfernt wurde und eine neue Verbindungsentität angelegt wurde. „TblMedia“ und „TblObservedObject“ enthalten dann jeweils eine Liste von Objekten der Verbindungsentität.

Abbildung : Codeausschnitt: Neues Feld in der "TblMedia"-Klasse

Die Verbindungsentität namens „TblMediaJoinObservedObjectDataSet“ hat dann jeweils einen Eintrag für ein Media- und ein ObservedObject-Objekt. Zusätzlich hat die Verbindungsentität ein optionales Feld für die ID des Datensatzes, an den das Bild gebunden sein soll.

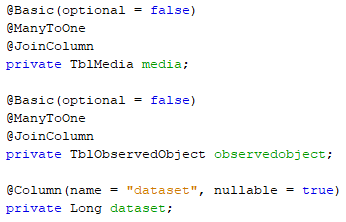


Abbildung : Codeausschnitt: Felder in der Verbindungsentität zwischen Media und ObservedObject

# Ausblick und Fazit

Michelle

## Ausblick

## Fazit

# Installationshinweise und Benutzerhandbuch

## Installationshinweise

Die Installation und Benutzung kann nur einwandfrei auf einem Server mit einer frisch aufgesetzten Datenbank funktionieren, da die IDs der benötigten ObservedObjects hardkodiert sind. Die Möglichkeit, diese IDs dynamisch, auf eine Art, die nicht jede Menge Programmieraufwand hat, abzufragen, besitzt SWAC noch nicht.

Die Installation des Modulportals folgt nach der normalen Installation des SmartMonitoring-Frontends und -Backends. Es muss lediglich folgende drei Punkte gemacht werden:

* Das zusätzliche Projekt „Basics\_ObservedObjectDataSetComment“ aus GitLab klonen und in den Ordner „projects“ im SmartMonitoring Ordern ablegen
* In der Datenbank manuell die kryptische Einschränkung in der Tabelle „tbl\_measurement\_type“ löschen
* Zu der Seite „/SmartMonitoring/sites/modulesetup.html“ navigieren, auf den Knopf „Alle Ausführen“ klicken und ca. 20 Sekunden warten, bis alle Schritte der Installation automatisch abgeschlossen werden

Eine Installationsanleitung ist auch nochmal unter diesem Link zu finden:  
<http://git01-ifm-min.ad.fh-bielefeld.de/Forschung/scl/2015_03_SCL_SmartMonitoring_Frontend/-/wikis/Installationsanleitung%20PVModulPortal>

## Benutzerhandbuch

Moritz

# Quellenverzeichnis

Abbildung 1: Aufbau einer Photovoltaik-Zelle: [https://de.wikipedia.org/wiki/Photovoltaik, Zugriff 26.01.2020]

Abbildung 2: <https://www.teachengineering.org/content/cub_/lessons/cub_pveff/Attachments/cub_pveff_lesson03_fundamentalsarticle_v11_tedl_dwc.pdf>

Abbildung 3: <https://photovoltaiksolarstrom.com/photovoltaiklexikon/maximum-power-point-mpp-tracking/>

Dunkelkennlinien: <https://photovoltaikbuero.de/pv-know-how-blog/dunkelkennlinien-von-photovoltaikanlagen-messen/>

Bild 2 & Inhalt: https://photovoltaikbuero.de/pv-know-how-blog/was-bringt-eine-kennlinienmessung-bei-der-fehlersuche-an-pv-anlagen/

# Anlagen