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](#_Toc30585855)

[1.1 Motivation 1](#_Toc30585856)

[1.2 Themenbeschreibung 1](#_Toc30585857)

[2 Vorstellung der Gruppenmitglieder 1](#_Toc30585858)

[2.1 Vorstellung der Gruppenmitglieder 1](#_Toc30585859)

[2.2 Aufgabenverteilung im Team 1](#_Toc30585860)

[3 Theoretische Grundlagen 2](#_Toc30585861)

[4 Konzeptionelle Arbeiten 2](#_Toc30585862)

[5 Code-Implementierungsdetails 2](#_Toc30585863)

[6 Ausblick und Fazit 3](#_Toc30585864)

[6.1 Ausblick 3](#_Toc30585865)

[6.2 Fazit 3](#_Toc30585866)

[7 Installationshinweise und Benutzerhandbuch 3](#_Toc30585867)

[7.1 Installationshinweise 3](#_Toc30585868)

[7.2 Benutzerhandbuch 3](#_Toc30585869)

[8 Quellenverzeichnis 3](#_Toc30585870)

[9 Anlagen 3](#_Toc30585871)

# Motivation und Themenbeschreibung

Moritz

## Motivation

## Themenbeschreibung

# Vorstellung der Gruppenmitglieder

## Vorstellung der Gruppenmitglieder

In unserem Projektteam waren vier Mitglieder beteiligt – Hannes Rüffer, Moritz Withöft, Michelle Vorwerk und Christiane Zolkin. Jedes Teammitglied hat im Wintersemester 2017 mit dem Informatik Studium begonnen, wodurch sich eine gemeinsame Grundwissensbasis ergab.

Hannes Rüffer, der ein gutes technisches und logisches Verständnis mit sich bringt, ist in der Lage einen Überblick über alle Projektanforderungen zu bewahren und Ansätze für die Entwicklung zu finden. Durch seine Arbeitsmoral ist er ein guter Helfer und Problemlöser und seine Stärke liegt darin, das Projekt auch bei Schwierigkeiten voranzutreiben.

Moritz Withöft ist ebenfalls ein sehr motivierter Mitarbeiter und Projektleiter. Seine Stärke besteht in seiner Zuverlässigkeit, aufgetragene Aufgaben effektiv zu bewältigen. Durch seinen Nebenberuf durfte er bereits Erfahrungen im Bereich Webengineering sammeln, und konnte dies in diesem Projekt gut einsetzen.

Auch Michelle Vorwerk konnte bereits durch beruflicher Erfahrung viel zum Projekt beitragen. Durch kreieren von Weboberflächen hat sie ein Auge für Ästhetik und Pragmatik von Webseiten entwickelt, was sehr hilfreich für die Projektentwicklung war.

Christiane Zolkin ist eine zielorientierte Persönlichkeit und außerdem lernwillig, um neue Aufgaben souverän bewältigen zu können. Dadurch konnte sie zu dem Projekt einige Aspekte beitragen und Teilbereiche auf sich nehmen.

## Aufgabenverteilung im Team

Durch die Stärken, Schwächen und unterschiedlichen, individuellen Eigenschaften aller Teammitglieder, konnten wir die Rollen innerhalb des Projektes gut einteilen.

Die Projektleiterrolle und Rolle des Dokumentationsmanagers wurden an Moritz Withöft übergeben, aufgrund seiner Zuverlässigkeit, Motivation und Sprachbegabung.

Wegen ihres Überblicks des Projektgeschehens, war Michelle Vorwerk Schnittstellen-Manager, um alle Teilbereiche der Entwicklung strukturiert zusammenzuführen.

Da das Projekt sehr Frontend-orientiert war, entwickelten Moritz, Michelle und Christiane gemeinsam an diesen Teilbereich.

Das Backend war auch sehr arbeitsintensiv, aber durch Hannes gutes Verständnis für diesen Bereich, konnte er diese Aufgabe gut übernehmen.

Alles in allem herrschte aber eine gute Gruppendynamik, und die Rollenverteilung innerhalb des Projektes war nicht statisch, sondern eher interaktiv, wodurch viele Probleme gelöst werden konnten, indem man sich einander half.

# Theoretische Grundlagen

Hannes

# C:\Users\Christiane Zolkin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\UseCase_PVModul.pngKonzeptionelle Arbeiten

Abbildung

Um alle Anforderungen übersichtlich darzustellen, wurde zunächst ein Use-Case Diagramm angelegt (Abbildung 1). Damit konnte man abgleichen, was der Auftraggeber erwartet, und somit was die Entwickler zu implementieren haben. Missverständnisse wurden somit schnell identifiziert und die Anforderungen konnten gut geändert werden.

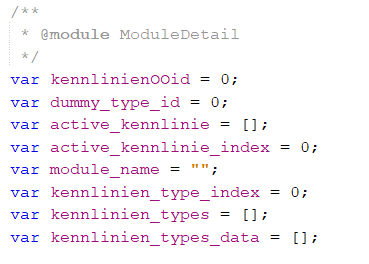
Wichtig sind dabei eine Modulübersicht, sowie eine Modulabfrage über ein Suchfeld. Es ist möglich nach speziellen Modultypen zu suchen, allerdings aber auch nach Herstellern und deren Produkte. Außerdem möchte man Daten mehrerer Module vergleichen können, indem man Module auswählen kann und auf eine Detailseite weitergeleitet wird. Dort werden technische Daten des Moduls und Kennlinien übersichtlich angezeigt. Außerdem gibt es eine Kommentarsektion, in der Diskussionen und Fragen von Kunden geklärt werden können, indem man Kommentare lesen, aber auch schreiben kann.

Des Weiteren sollten auch neue Modultypen von angemeldeten Nutzern hochgeladen werden können.

# Code-Implementierungsdetails

## Kennlinien eines Moduls anzeigen und neue Kennlinien hochladen

### Kennlinien anzeigen

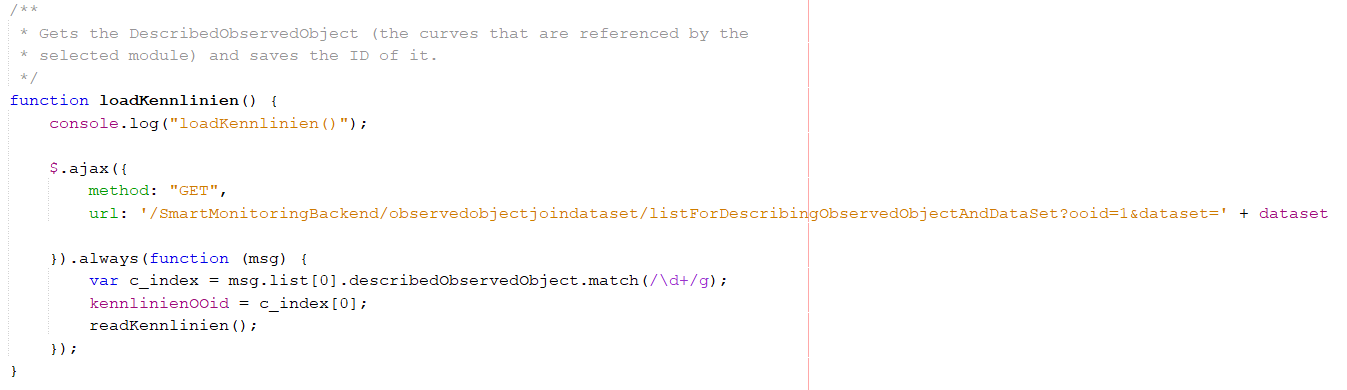


Code Abschnitt - Kennlinien anzeigen

Um Kennlinien in einem Diagramm anzeigen zu können, werden Datenstrukturen gebraucht, um die von der Datenbank geholten Daten zu speichern und somit verarbeiten zu können. Hierzu werden Variablen angelegt.

Zur Speicherung der Kennlinien zugehörig zu einem Modultypen, wurde ein Dummy Type ObservedObject angelegt. Dessen ID (dummy\_type\_id) wird gefetcht, um das referenzierte ObservedObject – die zugehörigen Kennlinien also – herauszufinden (kennlinienOOid).

Die restlichen Variablen sind Datenstrukturen für die Speicherung der von der Datenbank geladenen Kennlinien. Dabei handelt es sich bei „kennlinien\_types\_data“ um ein dreidimensionales Array, welches die Kennlinien-Typen, die Kennlinien-Datensätze und die Punkte einer einzelnen Kennlinie enthält. „active\_kennlinie“ stellt dabei die aktuell angezeigte Kennlinie im Diagramm dar. Sie wird ermittelt über „kennlinien\_type\_index“ und „active\_kennlinie\_index“.



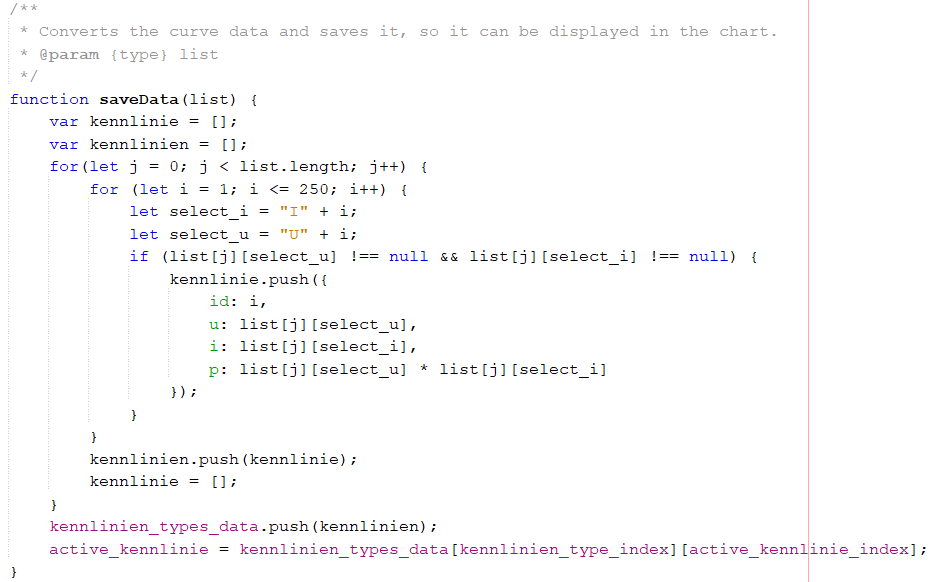
Code Abschnitt - Kennlinien anzeigen

Mit dem Laden der Seite wird die Funktion „loadKennlinien()“ initiiert. Mit der Modul ID werden dann die DescribedObservedObjects geladen und die Kennlinien ID gespeichert. Nun können die Kennlinien ausgelesen werden.



Code Abschnitt - Kennlinien anzeigen 3

Die Kennlinien ObservedObject ID ermöglicht das Auslesen aller Kennlinien eines Modultypen. Mit der Schnittstelle „observedobject/listChilds“ werden alle in der Datenbank gespeicherten Kennlinien Typen zurückgegeben – Hellkennlinien, Dunkelkennlinien und Laborkennlinien. Jeder Eintrag in der zurückgelieferten Liste stellt einen Kennlinientypen dar, und bei jedem Eintrag handelt es sich um ein Array mit mehreren Kennlinien Datensätzen. Diese Daten werden nur mit der Funktion „saveData()“ gespeichert und für das Diagramm verfügbar gemacht.

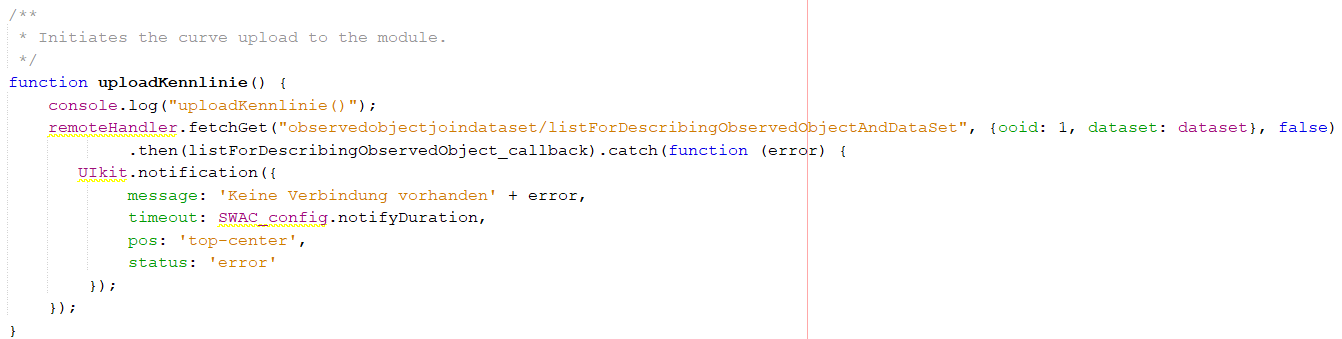


Code Abschnitt - Kennlinien anzeigen 4

In „saveData()“ werden die zurückgelieferten Kennliniendaten mit einer doppelten for-Schleife durchiteriert. Die äußere for-Schleife ist für die Iteration durch die Kennlinien-Typen verantwortlich, die innere für das Durchlaufen aller Datensätze des aktuellen Kennlinien-Typs.

Dementsprechend wird ein dreidimensionales Array zur Speicherung aller Daten herangezogen – um Kennlinien-Typen, Kennlinien-Datensatz und Kennlinien-Punkte abspeichern zu können und die Daten an das Diagramm zu überliefern. Angezeigt wird dann die „active\_kennlinie“, welche über den Kennlinien-Type-Index und den Aktiven-Kennlinie-Index (beide anfangend bei 0) ermittelt wird.

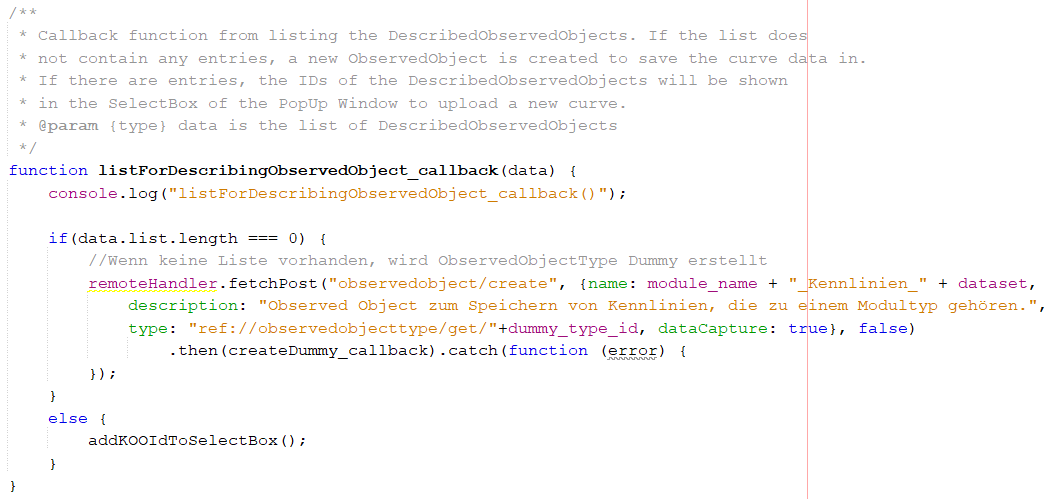
### Kennlinien hochladen



Sobald „Neue Kennlinie hinzufügen“ auf der Nutzeroberfläche angeklickt wird,

wird die Funktion „uploadKennlinie()“ ausgeführt.

Der Upload besteht aus mehreren Schritten: Zunächst wird über die Schnittstelle „observedobjectjoindataset/listForDescribingObservedObjectAndDataset“ die Liste aller vom Modultypen referenzierten Kennlinien zurückgegeben, mit der über die Callback-Funktion „listForDescribingObservedObject\_callback()“ gearbeitet werden kann.



In der Callback-Funktion wird nun geprüft, ob in dieser Liste Einträge vorhanden sind – also, ob bereits ein Kennlinien-ObservedObject vorhanden ist, bei dem neue Datensätze hinzugefügt werden können. Falls nicht, wird dieses ObservedObject erstmals angelegt. Ansonsten wird der User mit der zurückgegebenen Kennlinien-OOID zur Upload-Seite weitergeleitet.



Wenn das Kennlinien-ObservedObject erfolgreich angelegt werden konnte, wird die „createDummy\_callback()“-Funktion ausgeführt, welche eine Referenz zwischen dem aktuellen Modultypen und dem neu kreierten Kennlinien-ObservedObject schafft. Falls auch dies erfolgt, kann der User zur Upload-Seite des Kennlinien-Dummys weitergeleitet werden. Nach dem Hochladen können nun die hochgeladenen Kennlinien beim nächsten Aufruf der Moduldetail-Seite angezeigt werden.

# Ausblick und Fazit

Michelle

## Ausblick

## Fazit

# Installationshinweise und Benutzerhandbuch

## Installationshinweise

Hannes

## Benutzerhandbuch

Moritz

# Quellenverzeichnis

# Anlagen