



Hochschule für Angewandte Wissenschaften Hamburg  
*Hamburg University of Applied Sciences*

Development of a Computer-Aided Assistance for the  
Maintenance of an Energy Management System According to  
DIN EN ISO 50001

Gerrit Schutz

Gerrit Schutz

Development of a Computer-Aided Assistance for the Maintenance of an  
Energy Management System According to DIN EN ISO 50001

Bachelorarbeit eingereicht im Rahmen des Umwelttechnik Studiums  
am Department Umwelttechnik  
der Fakultät Life Sciences  
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Herr Prof. Dr. Kay Förger

Zweitgutachter: Herr Dipl.-Ing. (FH) Nils Heinrich

Abgegeben am 14.02.2018

Gerrit Schutz

Thema der Arbeit

Entwicklung einer rechnergestützten Assistenz zur Aufrechterhaltung eines Energiemanagementsystems nach DIN EN ISO 50001

Stichworte

Energiemanagement, ISO 50001, Energiemonitoring, JEVIs Software

Kurzzusammenfassung

Das Ziel dieser Arbeit ist es, eine Software-basierte Lösung zu schaffen, um ein Energiemanagementsystem effektiv zu verwalten, anstatt nicht-standardisierte proprietäre Software zu verwenden, um dasselbe zu tun. In der angebotenen Lösung wird das Open Source JEVIs System sowohl als Datenspeicher als auch als Schnittstelle genutzt. Die Präsentation erfolgt über einen Browser mit HTML und JavaScript.

Gerrit Schutz

Title of the Paper

Development of a Computer-Aided Assistance for the Maintenance of an Energy Management System According to DIN EN ISO 50001

Keywords

Energy management, ISO 50001, energy monitoring, JEVIs Software

Abstract

The purpose of this thesis is to create a software-based solution to manage effectively an energy management system, instead of using nonstandard proprietary software to do the same. In the offered solution, the open source JEVIs System is used as a data storage as well as an interface: the presentation is made on a browser using HTML and JavaScript.

# Table of Contents

<b>1 Motivation .....</b>	<b>6</b>
<b>2 Introduction .....</b>	<b>7</b>
2.1 Envidatec GmbH.....	7
2.2 Energy management according to ISO 50001 .....	8
2.3 JEVIS System.....	10
<b>3 Objectives .....</b>	<b>12</b>
<b>4 Analysis .....</b>	<b>13</b>
4.1 JEVIS Components.....	13
4.2 Planning the first steps.....	18
4.3 The development environment.....	19
4.4 Approach.....	20
<b>5 Conception and Design .....</b>	<b>28</b>
5.1 JEVIS Classes for ISO 50001 .....	28
5.2 User Interface .....	38

<b>6</b>	<b>Implementation.....</b>	<b>40</b>
6.1	Java Classes.....	40
6.2	Template files.....	48
6.3	Generation of browser pages on the server .....	50
<b>7</b>	<b>Demonstration .....</b>	<b>53</b>
<b>8</b>	<b>Conclusion and Outlook.....</b>	<b>64</b>
<b>9</b>	<b>Appendices .....</b>	<b>66</b>
9.1	Table of JEVIS Class Structure for ISO 50001 .....	66
9.2	Bibliography .....	66
9.3	List of Tables.....	67
9.4	List of Figures .....	67

# 1 Motivation

Energy issues are gradually emerging in the consciousness of the general public. While reducing the impact on the environment, such as through the reduction of CO<sub>2</sub> emissions, saving energy has become more and more popular. The interests behind it are complex: although the general interest of people is being concerned with saving money, most are concerned with adhering to the ever lower legal limits for environmental, human, animal and natural conservation, while the rest is doing it out of conviction.

Politically, the two core pillars of the energy revolution in Germany are renewable energy and energy efficiency. On the one hand, addressing the generation side to reduce greenhouse gas emissions from fossil fuel sources and, on the other hand, to minimize consumption on the consumer side, as well as increase efficiency in energy production and energy transportation. The topic of this thesis falls into the second pillar, energy efficiency.

The introduction of an energy management system is intended to continuously improve a company's energy efficiency. Through my work as a student trainee, I have made many contacts within this topic and have been able to gather adequate experience in this subject. It is observed that companies use their own system, either for documenting the management system or the way in which energy data is recorded and managed. This gives the need for other systems to merge the data. In order to make the organizational and written procedures associated with the introduction and operation of an energy management system efficient and more standardized, this work tries to build up a computer-aided assistance for the processes involved.

## 2 Introduction

The following three sections provide an introduction to the company Envidatec GmbH, convey a basic understanding of the Energy Management System according to ISO 50001 and lastly give a short introduction to the JEVIS System.

### 2.1 Envidatec GmbH

Envidatec GmbH [1] was founded in 2001 in Hamburg, Germany by a group of specialized engineers from various disciplines. Since then, Envidatec has been in business for over 15 years, and is a well-known national and international partner for many companies.

Envidatec employees have been active in the fields of energy efficiency, energy management, energy data management, energy benchmarking, conducting energy audits and implementing energy optimization measures. Envidatec engineers are specialists in electrical engineering, environmental engineering, system engineering and renewable and process engineering among others. Envidatec's goal is to optimize the energy efficiency and energy consumption of companies and institutions not only partially and temporarily, but holistically and sustainably.

Up to date, Envidatec has successfully completed over 250 energy related projects in more than 30 countries. The flexibility, expanse of knowledge and many years of experience have established Envidatec as a global acting company in all aspects of energy efficiency and the optimization of energy consumption.

In order to ensure the sustainability of optimization measures implemented, Envidatec supports customers with technical solutions, from planning to complete system solutions, for the necessary measurement and continuous analysis of energy flows and processes within the company. Apart from the audit and consultancy services,

Envidatec also provides energy monitoring software and hardware solutions of the JEVIS brand, which is mainly developed and cultivated in the framework of the open source project OpenJEVIS by Envidatec. Envidatec collaborates with Universities around the world to provide state of the art R&D in the topics of energy monitoring, analysis and control.

## **2.2 Energy management according to ISO 50001**

The standard ISO 50001 is designed according to other ISO management system standards, in particular Quality Management Systems according to ISO 9001 and Environmental Management Systems according to ISO 14001.

ISO 50001 focuses on a continual improvement process to achieve the objectives related to the energy performance of an organization. This process follows a Plan-Do-Check-Act cycle also called the PDCA approach:



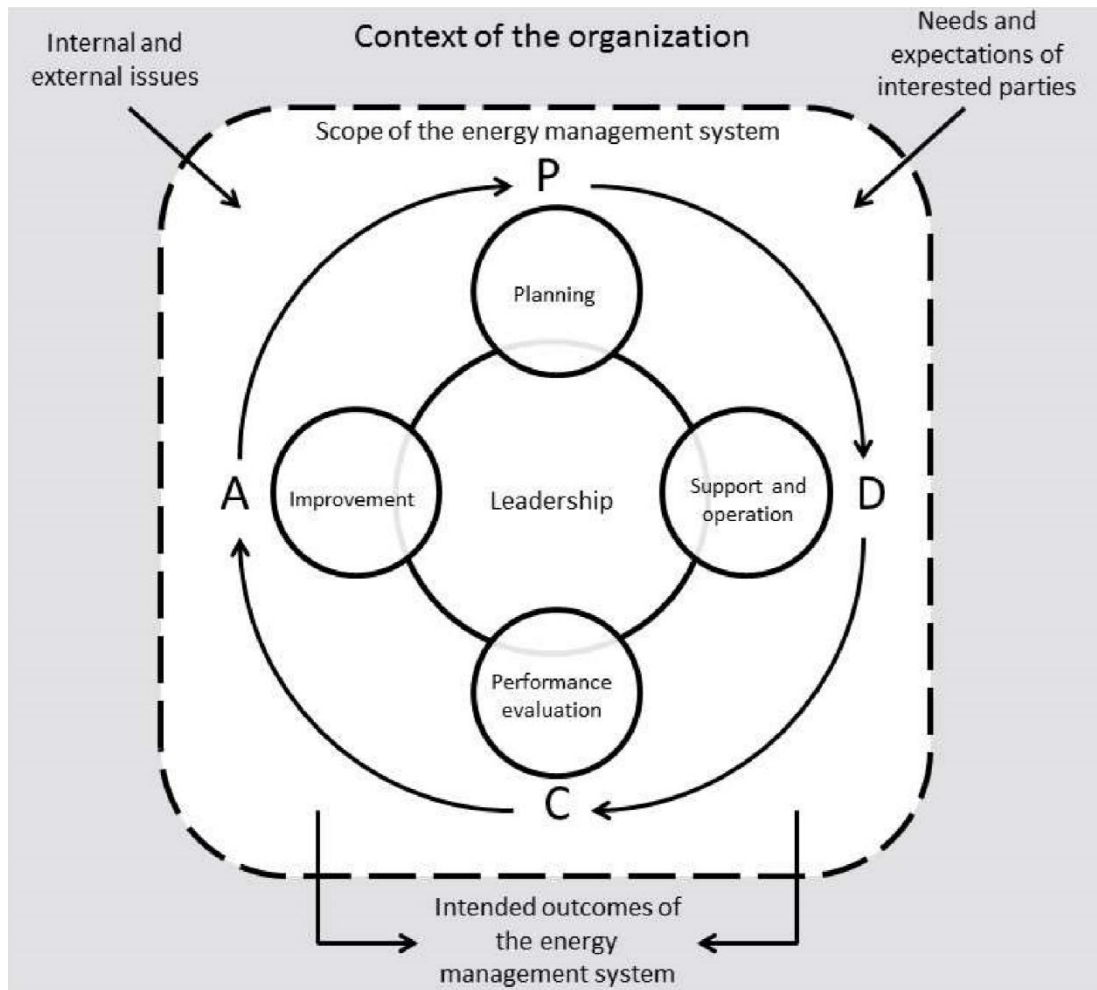


Figure 1 Plan - Do - Check - Act Cycle illustration taken from the ISO 50001 [2]

The *Plan* step (P in Figure 1) consists of conducting the energy review and establishing the energy baseline (EnB), energy performance indicators (EnPIs), objectives, targets and action plans necessary to deliver results that will improve energy performance in accordance with the organization's energy policy.

During the *Do* part (D in Figure 1) the energy management action plans need to be implemented.

Monitoring, measuring and reporting processes determine the key characteristics of operations for energy performance matching to the energy policy and objectives. These actions take place in the *Check* phase (C in Figure 1).

Lastly, taking actions to continually improve the energy performance and the energy management system (EnMS) fall into the *Act* part.

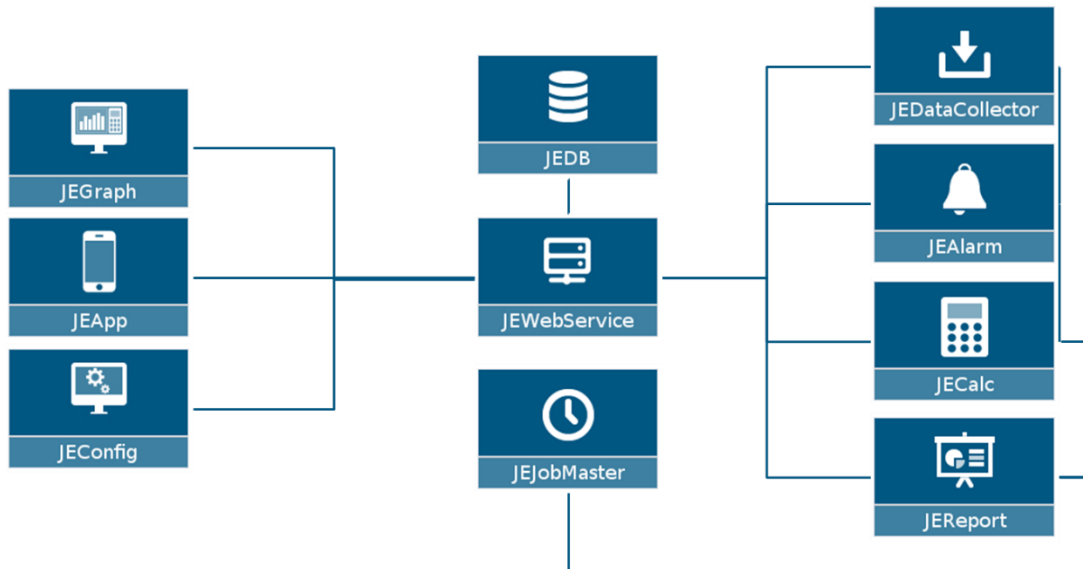
This thesis tries to incorporate the newest revision of the DIN EN 50001 standard. This newest revision was drafted in September 2017, with a planned release date of August, 18<sup>th</sup> of 2017. The final release is expected at the end of 2018 or early 2019. Main changes, of which the adoption of the High Level Structure (HLS) being the most fundamental change, are according to the preface of mentioned revision [2]:

- Adoption of the Annex SL Annex 2, High Level Structure (HLS) text to ensure a high level of compatibility with other management system standards,
- Clarification of language and organization,
- Definitions in Section 3 are in context order,
- Energy Review has been clarified,
- Normalization of EnPI(s) and associated EnB(s),
- Clarification on the energy data collection plan and related requirements (previously energy measurement plan),
- EnPI and EnB have been clarified to provide a better understanding of these concepts.

## 2.3 JEVIs System

The JEVIs System [3] is an open source software designed for energy and operational data storage and monitoring. All the components of the JEVIs System are written in Java and can be installed on any operating system (e.g. Windows, Linux, etc.),

including the graphical user interface (GUI) components, state machine-based processing, and most importantly the JEVIS System database.



**Figure 2 JEVIS System Architecture, Envidatec 2014 – The Open JEVIS System**

The core of the JEVIS System is the JEWService (Figure 2), which accesses a MySQL database system. The database is a relational database and is divided into a system-related and a user-customizable area. It is not limited to any particular energy data format, and nearly every possible data format can be stored.

The JEVIS database design is object oriented with inheriting functions via parent and child objects. The data is stored in rows, while all changes are audited. It uses the relational database to store the data in an object-oriented way. It supports SI units and different time zones. The JEWService is a Representational state transfer, called REST for short, webservice, which provides clear HTTP-based syntax and structures to access the data stored in the JEVIS data base (JEDB). JEVIS is open source and follows the GPL and CC-BY-NC license [3].

## 3 Objectives

The aim of this thesis is to develop of an intuitive and functional graphical user interface (GUI) for a web-based application to manage all data for the implementation of an energy management system according to ISO 50001 and the continuous operation of an EnMS.

This includes managing of the corresponding ISO 50001 documents and their applications:

1. Energy Policy
2. Index of Legal Provisions
3. Energy Flow Chart
4. Energy Baseline of all Energy Sources
5. Energy Performance Indicators (EnPIs)
6. Action Plans
7. Training Records
8. Checklists for Internal Audits and Checklists for Energy Audits
9. Report for Deviations
10. List of Corrective and Preventive Measures
11. Records of Management Reviews

Especially Energy Performance Indicators and Action Plans depending on them can be sourced through the obtained data.

## 4 Analysis

The next part introduces some JEVis components, that are essential for this work. This chapter also analyzes the materials that can be used, the development environment and the approach to start on the project.

### 4.1 JEVis Components

There are some key components of the JEVis System, that need to be understood in the following content. Those are:

**JEDB** - JEVis database, standard MySQL database with a custom scheme (Figure 3).

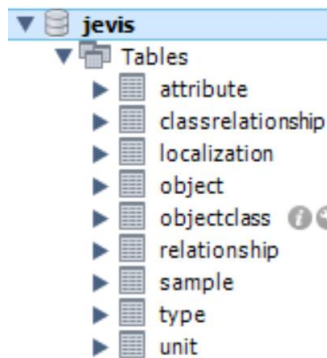


Figure 3 JEVis database scheme

**JEVisClass** – Class of a *JEVisObject* in the *object* table (Figure 4). In the following document mostly abbreviated to class. MySQL command:

```
CREATE TABLE objectclass (  
    name varchar(255) NOT NULL,  
    icon blob,  
    description varchar(1024) DEFAULT NULL,
```

```

isunique tinyint(1) DEFAULT '0',
PRIMARY KEY (name)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

name	icon	description	isunique
Legal Regulation	BLOB	NULL	0

**Figure 4 JEVISClass properties**

The ***classrelationship*** table defines the relationship of class parents and children (Figure 5), which describes the inheritance relations between classes. MySQL create command:

```

CREATE TABLE classrelationship (
  startclass varchar(255) NOT NULL,
  endclass varchar(255) NOT NULL,
  type int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (startclass, endclass, type)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

startclass	endclass	type
01 General	Internal Audit	0

**Figure 5 JEVIS classrelationship properties**

***JEVisObject*** – Standard object, can be a directory or a single object (Figure 6). It can contain *JEVisAttributes* and is of a defined *JEVisClass*. The Create Table command is:

```

CREATE TABLE object (
  id bigint(20) NOT NULL AUTO_INCREMENT,
  name varchar(255) NOT NULL,
  type varchar(255) NOT NULL,
  link bigint(20) DEFAULT NULL,
  deletets date DEFAULT NULL,
  #delete timestamp

```

```

PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=8064 DEFAULT CHARSET=utf8;

```

id	name	type	link	deletets
7913	EDL-G	Legal Regulation	NULL	NULL

Figure 6 JEVisObject properties

The **relationship** table maps the relationship between *JEVisObjects* according to the classrelationship (Figure 7).

```

CREATE TABLE relationship (
  startobject bigint(20) NOT NULL,
  endobject bigint(20) NOT NULL,
  relationtype int(11) NOT NULL,
  PRIMARY KEY (startobject, endobject, relationtype),
  UNIQUE KEY `UNIQUE_CHECK` (startobject, endobject,
relationtype)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

startobject	endobject	relationtype
1	1	3

Figure 7 JEVIs relationship properties

**JEVisAttribute** – Belongs to an *JEVisObject*. It is designed to describe a complete sample of single value. It contains data about timestamps, data input and output type and sample rate (Figure 8). MySQL table structure is as follows:

```

CREATE TABLE attribute (
  name varchar(255) NOT NULL,
  object bigint(20) NOT NULL,
  mints datetime DEFAULT NULL,
  maxts datetime DEFAULT NULL,
  samplecount int(11) DEFAULT '0',

```

```

period varchar(255) DEFAULT NULL,
unit varchar(45) DEFAULT NULL,
altsymbol varchar(255) DEFAULT NULL,
inputunit varchar(1024) DEFAULT NULL,
inputrate varchar(255) DEFAULT NULL,
displayrate varchar(255) DEFAULT NULL,
displayunit varchar(1024) DEFAULT NULL,
opt varchar(10240) DEFAULT NULL,
PRIMARY KEY (name, object)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

name	object	mints	maxts	samplecount	period	unit	altsymbol
01 Januarv	7895	2017-12-08 17:14:14	2017-12-08 17:14:14	1	NULL	NULL	NULL
inputunit		inputrate		displayrate	displayunit		opt
{ "formula": "", "label": "", "prefix": "None" }		PT15M	P1M	{ "formula": "", "label": "", "prefix": "None" }		NULL	

**Figure 8** JEVisAttribute properties

The **type** table further describes *JEVisAttributes*, especially to which *JEVisClass* they belong to and, how they are displayed (Figure 9).

```

CREATE TABLE type (
  name varchar(255) NOT NULL,
  jevisclass varchar(255) NOT NULL,
  displaytype varchar(255) DEFAULT NULL,
  primitivtype int(11) DEFAULT NULL,
  guiposition int(11) DEFAULT NULL,
  defaultunit varchar(45) DEFAULT NULL,
  description varchar(255) DEFAULT NULL,
  validity varchar(45) DEFAULT NULL,
  value varchar(255) DEFAULT NULL,
  altsymbol varchar(255) DEFAULT NULL,
  inheritedt tinyint(1) DEFAULT NULL,

```



```

PRIMARY KEY (name, jevisclass),
UNIQUE KEY attindex (name, jevisclass)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

name	jevisclass	displaytype	primitivtype	guiposition	defaultunit	description	validity	value	altsymbol	inherited
Content Summary	Legal Regulation	Text Area	0	0	NULL	NULL	0	NULL		0
Date of Review	Legal Regulation	Date	0	0	NULL	NULL	0	NULL		0
Issue Date	Legal Regulation	Date	0	0	NULL	NULL	0	NULL		0
Last Amended	Legal Regulation	Date	0	0	NULL	NULL	0	NULL		0

Figure 9 JEVIS type properties

**JEVISample** – the main data storage for values, strings and files, including content (Figure 10).

```

CREATE TABLE sample (
  object bigint(20) NOT NULL,
  attribute varchar(255) NOT NULL,    #table attribute.name
  timestamp datetime NOT NULL,
  value varchar(2048) DEFAULT NULL,
  manid int(11) DEFAULT NULL,
  insertts datetime DEFAULT NULL,
  note varchar(45) DEFAULT NULL,
  file longblob,
  filename varchar(255) DEFAULT NULL,
  PRIMARY KEY (object, attribute, timestamp),
  UNIQUE KEY sampleindex (object, attribute, timestamp)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

object	attribute	timestamp	value	manid	insertts	note	file	filename
1	Domain Name	2002-06-15 00:00:00	108.93	NULL	2014-07-02 18:36:16	CSV Import	NULL	NULL

Figure 10 JEVISample properties

## 4.2 Planning the first steps

Work on the theme of this thesis is rather sparse. There is a thesis from 2015 [4] that deals with similar topics and discusses mostly the procedural part, i.e. primarily process management and energy technology, as well as the political and cultural aspects. This thesis however, will primarily focus on the software part. There needs to be a basic understanding of the ISO 50001 and the JEVIS System, to follow the discussion and development process.

The PDCA-cycle of the ISO 50001 induces the need for a data base and data storage for energy consumption values and a variety of documents recording these values, targets and protocols. There should be at least one overview page giving an overview of the consumption and the energy performance indicators compared to previous years. The organizational process should start with the setting up of a plan for the following steps. There have to be deliberations about the development environment. Questions that arise here, among others, are which software and which programming languages to use or what should be the system environment. There needs to be an evaluation of produced materials so far concerning this subject. However, because of the peculiarity of the theme, most are internal documents, like the previously created theses in the company. Publicly available publications from the libraries should be referenced, too. Considerations about how data is transferred to and from the JEWebService are necessary, which induces the need to think and evaluate how and what data is displayed to the user. Finally, a test with appropriate data to demonstrate and scrutinize the implemented system needs to be conducted.

The following framework plan was subjected to change through the conclusions made and experience gained during the duration of this thesis. Six main steps are necessary to complete the work:

1. Setting up the development environment
2. Viewing, analyzing and evaluating existing Java classes in the JEWebService
3. Creating or updating JEVIS ISO 50001 classes
4. Implementing data input and output for ISO 50001 into JEWebService
5. Creating respective html templates -> Use HTML Template Engine to fill datasets/forms
6. Applying use cases for demonstration, verification and optimization

Login into the Representational State Transfer (REST) service in the JEVIS System is done with a basic authentication. The user name and the password are Base64 encoded and sent to the server. This is no encryption so it is not very safe. This is a step that requires further work in the case of putting this project as a product into the market. This is also the case with regular data transfer. At the moment, for development purposes, it is done via unencrypted http transfer.

### **4.3 The development environment**

An installation of NetBeans 8.2 [5] is used as a development environment and for this work, the JEVIS Application Programming Interface (JEAPI), the MySQL based implementation of the JEAPI (JEAPI-SQL), the collection of common JEVIS functions (JECommons) and the JEVIS REST Webservice (JEWebService) are cloned from GitHub.

In the source package of the web service a new package called `org.jevis.rest.iso` is initialized. For development and testing purposes, all

classes are developed locally on the PC. The JEDB is runs on a standard local MySQL Server.

## 4.4 Approach

As mentioned above, first it is determined where the data will be visualized and in which format. There is a starting point with the JEVIS System, but it should be defined where the project will lead to. First deliberations result in an approach using WordPress plug-ins as a means to input and output data as interface to the user. [6] This is the result from recent experience in the establishment of new company presence in the internet.

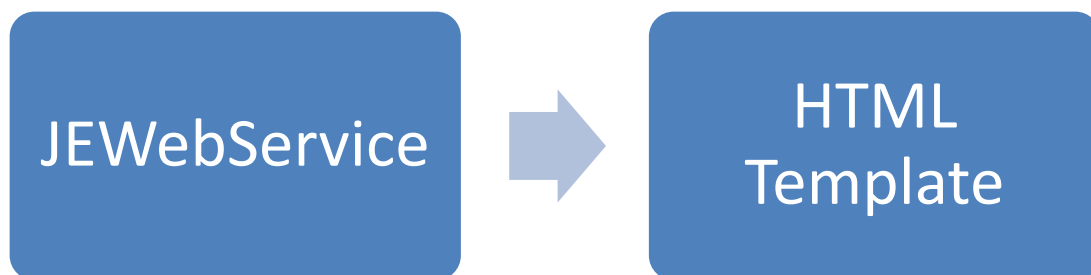


Figure 11 First Deliberations User Interface

After a short discussion with the lead JEVIS developer, the first draft is dismissed in favor of a more open and versatile concept [7]. The above solution (Figure 11) would implement a multi-step access chain. This is to be avoided. In this case, a request first contacts the PHP driver, which in turn requests the JEWService, finally accessing the database via the JEWService. This is an unnecessary step in the request chain. Furthermore, this duplicates requests in both directions, e.g. the user requests data from the database, or the user wants to update or upload some data.

---

From the perspective of performance, it is recommended to integrate the access directly into the JEWebService by means of a template engine. This engine allows customizable html documents, which can be filled with data. This leads to a revised architecture (Figure 12). Further positive effects could be generated, e.g. reports as html, which can be simply converted to pdf. These allows visualization with charts in the JEVIS Control Center.



**Figure 12 Revised Concept of request chain**

Parallel to these considerations, existing materials on this subject are evaluated and analyzed. The first aspect that needs to be determined is the intended document format for the document management aspect of the structure. There are two main distinctions: first the document can be a file, which can be proprietary or open source. The second choice is the document as an JEVIS Object, which would be equivalent to a database object. Because JEVIS is open source, this includes the same advantages as an open source file and more. Each solution has its own advantages and disadvantages. Some general distinctions between open source and proprietary software are listed below (Table 1):

Table 1 Advantages and disadvantages of open source and proprietary software

Open		Proprietary	
Advantage	Disadvantage	Advantage	Disadvantage
Costs	Customer support with exceptions	Customer support, service level agreement	Costs
Customizing of solutions		Appeal	dependency
Fast updates and bugfixes		Continuous updates	Adaption to special needs

The basic data of all the documents are to be realized as JEVis Objects [8], with a few exceptions for printed document files such as the Management Review file and the Energy Flow Chart file. The Management Review needs to be printed and signed and the Flow Chart is an image file, mostly created with different third-party programs.

In a previous study [9], the application area of the structure was differentiated into two parts: first there are steps, which need to be adhered to in the case of implementing an energy management system according to ISO 50001 into an organization: secondly, a different form of approach is required concerning the management of a running system. The JEVis class structure has to be able to manage both applications. This should be realized in such a way, that two instances for the same concept are avoided.

In the class hierarchy, the top most is an "Organization". This is followed downwards by "Energy Management Directory", "Energy Management System Directory", "ISO 50001 Implementation Directory" and "ISO 50001 Steps". This one interleaving at the

beginning is superfluous. To optimize the structure the proposed differentiation of implementation and continuous operation of an energy management system should be avoided. There should be just a single *ISO 50001 Directory* below the organization object. “ISO 50001 Steps” contains the following objects:

Table 2 ISO 50001 Steps from previous study [9]

Document, Implementation Step	File Type, Attributes
First contact	Date, Text documents, descriptions of documents
Meetings <ul style="list-style-type: none"> <li>• Initial Meeting</li> <li>• Follow-Up Meetings</li> </ul>	Project Start, Contact Person, Content of Meeting and Results, Date, Other Participants, Report-File, Project End, Auditor
Gap Analysis <ul style="list-style-type: none"> <li>• Equipment Register <ul style="list-style-type: none"> <li>▪ Air Condition,</li> <li>▪ Compressor,</li> <li>▪ Cooling,</li> <li>▪ Heating,</li> <li>▪ Lighting,</li> <li>▪ Office,</li> <li>▪ Pantry,</li> <li>▪ Production,</li> <li>▪ Ventilation</li> </ul> </li> </ul>	Date Begin, Date End, Report File,

---

Responsibilities	Energy Manager, Energy Team, Contact, File for Assignments
Documents and energy planning <ul style="list-style-type: none"> <li>• EnMS Manual</li> <li>• Procedural Instructions</li> <li>• Energy Flow Charts</li> <li>• Wiki</li> <li>• Worksheets</li> </ul>	All Documents with Date of Creation, Date of Decontrol, Number, Responsible Person for Creation/Decontrol, Title, Version, Content
Trainings <ul style="list-style-type: none"> <li>• Participants</li> </ul>	Date, Description, Schedule File, Training Title, Trainer, Location, Time  Participants: First Name, Last Name, Certificate File
Internal Audit	Auditor, Report File, Date Begin, Date End, Participants, Schedule File, Remarks
Management-Review	Date, Participants, Report File, Notes, optional Auditors
Certification	Date Begin, Certificate File, Date End, External Auditor, Location, Schedule File, Report File, Participants, Remarks

This draft of class structures was created by a previous bachelor thesis [9][5] and needs optimization, partly for the use case, because the software solution presented by this thesis wants to address users on the customer side primarily, not Envidatec



employees and of course for the technical programming and data processing conditions and requirements.

The approach presently used by Envidatec in the day-to-day business is based on a fileserver (Table 3) which should be the starting point for developing and improving the new structure.

**Table 3 Currently existing file types in use-cases, file-based structure**

<b>Document</b>	<b>File Type</b>
Energy policy	Text document
Index of Legal Provisions	Spreadsheet file
Energy Aspects Matrix	Spreadsheet file
Sankey Diagram	Image or PDF from external program
Energy Baseline of all Energy Sources	Spreadsheet file
Energy Performance Indicators (EnPIs)	Spreadsheet file
Action Plan	Combination from Text document and Spreadsheet file
Training Records	Text documents, image files from certificates
Checklists for Internal and External Audits	Spreadsheet file
Checklist for Energy Audits	Spreadsheet file

---

Report for Deviations	Mixed Text and Spreadsheet
List of Corrective and Preventive Measures	Mixed Text and Spreadsheet
Records of Management Reviews	Text documents

On the whole, the process can be split into three parts. One part mainly concerns the document management. At best all texts, dates and versioning take place as JEVIs objects. The possibility to post documents, e.g. text or spreadsheet documents should be provided, too, where it cannot be avoided. The second is concerning energy and production values or other data which allows the determination of performance indicators, and lastly, the energy monitoring and its links to the *Monitored Object Directory*, which can be an interface to the monitoring capabilities of the JEVIs System. The *Monitored Object Directory* contains the objects which are filled by the JEDataCollector with obtained data from a monitoring system.

There are three possible scenarios that need to be taken into account concerning the availability of data for energy consumptions, energy bills and the measurement of values.

1. The simple scenario
  - a. Energy bills each month
  - b. No sub-meters
    - => energy consumption is calculated pro rata on the basis of the load of connected load to the total energy consumption.
2. The mixed scenario
  - a. Energy bills each month
  - b. Some sub-meters
    - => remaining energy consumption is calculated pro rata on the basis of the connected load to the total energy consumption.
3. The optimal scenario
  - a. Energy bills each month
  - b. All equipment has its own sub-meter

# 5 Conception and Design

The following chapter covers the different classes required for the effective maintenance of an energy management. The structure of these classes is created using the JEVIS Control Center. In principle, there are three main types of classes: the first and hierarchical top most class is in the usual case a directory object. Directory classes are usually unique, which means they can only be created once on a level and they have no attributes. Therein are multiple objects of the same class or unique classes, which can be created only once. Section 5.2 describes the final structure. The creative process and why some choices are made has been discussed in the previous chapter 4 - Analysis.

## 5.1 JEVIS Classes for ISO 50001

This section covers the developed class structure in depth. The JEVIS System allows for most of the usual data types, e.g. String, Long, Double. It is further possible to link an object to another with its ID.

For understanding and visualization requirements the following shows an exemplary object structure derived of the class structure. Their Attributes in detail and how they are working together can be understood from the appendix 9.1 Table of JEVIS Class Structure for ISO 50001.


The following examples are anonymized and show an exemplary naming of each object. The names can be chosen arbitrarily and in different languages. The classes and their descriptions are in English. On the right side of the figures their unique object IDs are shown.

To meet the requirements of companies with multiple locations, this directory contains the objects of the newly created class *Site* and an object of the *Superior Level Meetings Directory* class (Figure 13).

▼  Organization	7742
▼  ISO 50001 Directory	7745
▶  Superior Level Meetings	7746
▶  Hamburg	7747

**Figure 13 Organization with top-level objects**

The *ISO 50001 Directory* class can have multiple sites as children, whether they are office sites, production sites or other (Figure 14). The *Site* class enables the children *Documents Directory*, *Energy Planning*, a *Meetings Directory* for Meetings at the site and *Monitoring Register*.

▼  Hamburg	7747
▶  Documents Directory	7752
▶  Energy Planning	7753
▶  Meetings in Hamburg	7748
▶  Monitoring Register	7944

**Figure 14 Site Object with its children**

The *Documents Directory* class contains all “former” Documents, which are now realized as native JEVs objects. Documents have to comply with the requirements of ISO 50001, e.g. a title, date, author or reference number, like a standard document management system. The first level contains only Directories (Figure 15).












▼  Documents Directory	7752
▶  Action Plan Directory	7955
▶  Announcement Directory	7921
▶  Audit Directory	7923
▶  Energy Team Directory	7963
▶  Legal Regulation Directory	7911
▶  Management Manual Directory	7780
▶  Management Review Directory	7939
▶  Procedural Documents Directory	8060
▶  Training Course Directory	7918
▶  Training Directory	7917

Figure 15 Documents Directory with its children

Periodic plans for energy saving actions according to the ISO 50001 are needed. The *Action Plan Directory* (Figure 16) lists all Action Plans of the energy management system. An *Action Plan* extends the *Document* class with the additional Attributes of Action Plan File and Participants. All Action Plans have an *Implemented Actions Directory* and a *Planned Actions Directory*, which can contain objects of the *Energy Saving Action* class.









▼  Action Plan Directory	7955
▼  Action Plan 2016	7956
▼  Implemented Actions Dire...	7957
 Lighting Renewal	7959
▼  Planned Actions Directory	7958
 Heating Renewal	7960










Figure 16 Action Plan Directory with examples for implemented and planned actions

The *Announcement Directory* contains all Announcements, which are relevant to the energy management system and which extend the *Document* class with an Attribute Announcement File, as they are normally a file containing text for printing requirements or similar (Figure 17).

▼  Announcement Directory	7921
 Apointment of Energy Manager	7922

**Figure 17 Announcement Directory**

The ISO 50001 requires regular planned Internal Audits to evaluate the energy management system (Figure 18). Like every other ISO certification there are determined intervals for monitoring and recertification audits. In the *Audit Directory* there can be *External* and *Internal Audit* class objects. External Audits are usually done by external service providers, so the class's most important Attribute is the Report File. Internal Audits allow for a wider range of possible questions defined with the *Audit Question* class.

▼  Audit Directory	7923
 External Audit 2016	7931
▼  Internal Audit 2016	7924
▼  01 General	7925
 Scope and Boundary	8045
 02 Plan	7926
 03 Do	7927
 04 Check	7928
 05 Act	7929

**Figure 18 Audit Directory**

The new revision of the ISO 50001 does not explicitly demand for an Energy Manager as it has been before. This structure still allows to define one, in respect to the

common handling in companies to designate a key executive. It can also be used to define responsibilities for various areas of the energy management system (Figure 19).

▼  Energy Team Directory	7963
 Energy Manager	7964
 Energy Team Member 1	7965
 Energy Team Member 2	7966

**Figure 19 Energy Team Directory and Responsibilities**

The Organization has to evaluate its compliance to legal regulations at planned intervals. The *Legal Regulation Directory* lists all the organization relevant regulations in context with energy (Figure 20). The corresponding class in the Directory is the *Legal Regulation* class.

▼  Legal Regulation Directory	7911
 DIN EN ISO 50001:2011	7912
 EDL-G	7913
 EnEG	7914

**Figure 20 Legal Regulations Directory**

In most cases the management manual of an organization is mapped to the underlying scheme of the ISO 50001. The *Management Manual Directory* orients itself along the chapters of the norm (Figure 21). The preliminary structure applies classes to each chapter and subchapter to the second level, e.g. 10.1. This allows some possibilities of customizing within the limits of the original structure of the ISO 50001.





















▼  Management Manual Directory	7780	▼  07 Support	7788
 01 Scope	7781	 7.1 Resources	7809
 02 Normative References	7783	 7.2 Competence	7810
▼  03 Terms and Definitions	7784	 7.3 Awareness	7811
 3.1 Terms related to the ...	7791	 7.4 Communication	7812
 3.2 Terms related to the ...	7792	▼  7.5 Documented Informat...	7813
 3.3 Terms related to Requ...	7793	 7.5.1 General	7815
 3.4 Terms related to Perfo...	7794	 7.5.2 Creating and Upd...	7816
 3.5 Terms related to Energy	7795	 7.5.3 Control of Docum...	7817
▼  04 Context of the Organizati...	7785	▼  08 Operation	7789
 4 Energy Management Sy...	7796	 8.1 Operational Planning ...	7818
 4.1 Understanding the Or...	7797	 8.2 Design	7821
 4.2 Understanding the Ne...	7798	 8.3 Procurement	7822
 4.3 Determining the Scop...	7799	▼  09 Performance Evaluation	7790
▼  05 Leadership	7786	 9.1 Monitoring, Measure...	7823
 5.1 Leadership and Comm...	7800	 9.2 Evaluation of Complia...	7824
 5.2 Energy Policy	7801	 9.3 Internal EnMS Audit	7825
 5.3 Organization Roles, R...	7802	 9.4 Management Review	7826
▼  06 Planning	7787	▼  10 Improvement	7827
 6.1 General	7803	 10.1 Nonconformity and ...	7828
 6.2 Actions to Address Ris...	7804	 10.2 Continual Improvem...	7829
 6.3 Energy Review	7805		
 6.4 Energy Performance I...	7806		
 6.5 Energy Baseline	7807		
 6.6 Objectives, Energy Tar...	7808		



Figure 21 Management Manual Directory

The ISO 50001 calls for regular evaluations of the energy management system. These are pictured in the *Management Review* class and listed in the *Management Review Directory* class (Figure 22).

▼  Management Review Directory	7939
 Management Review 2016	7940


**Figure 22 Management Review Directory**

Each company has its own procedural documents. Primarily they further elaborate on the various management manual chapters and are to give concrete instructions to the employees. For example, there is a procedural document to consider energy efficiency for procurement of new equipment. These objects of the class *Procedural Document* are in the *Procedural Document Directory* (Figure 23).

▼  Procedural Documents Directory	8060
 Beschaffung	8061

**Figure 23 Procedural Document Directory**






Regularly occurring trainings for energy related topics, as stipulated in the ISO 50001, are in the *Training Course Directory*, with their contents, presentation file, etc., while a *Training* which already took place is saved to the *Training Directory* (Figure 24). These trainings are linked to their corresponding training course via ID.

▼  Training Course Directory	7918
 Jaehrliche Schulung	7919
▼  Training Directory	7917
 Schulung 23.05.2016	7920

**Figure 24 Training and Training Course Directories**










*Energy Planning* contains less documents and more data in its sub-classes. Every company differs in the existing data situation. The possible sub-classes are shown in

Figure 25. The respective objects in this part of the structure contains the most numbers.

▼  Energy Planning	7753
▶  Energy Flow Chart Directory	7951
▶  Energy Sources Directory	8046
▶  Equipment Register	7754
▶  Performance Directory	7941


**Figure 25 Energy Planning Directory**

Energy bills and energy consumption on data basis of the energy supplier is logged into the respective sub-class object of an *Energy Source* class object, which are the classes *Energy Bills* and *Energy Consumption* (Figure 26). These are further differentiated by the type of energy source.

▼  Energy Sources Directory	8046
▼  Electricity	7903
 Energy Bills 2015	8053
 Energy Bills 2016	7906
 Energy Bills 2017	8052
 Energy Consumption 2015	8048
 Energy Consumption 2016	7907
 Energy Consumption 2017	8049
▶  Gas	7908










**Figure 26 Energy Sources Directory**

In the *Energy Flowchart Directory* (Figure 27), the images and/or files are listed, which present all the energy flows within the organization.

▼  Energy Flow Chart Directory	7951
 Energy Flow Chart 2016	7954

**Figure 27 Energy Flow Chart Directory**

The next item is *Equipment Register* class object (Figure 28). This builds onto a class structure that was developed in 2017 [10]. In order to simplify and summarize the energy-relevant objects, this will be modified in some parts. For optimization some documents and worksheets from the day-to-day use of Envidatec concerning energy management are integrated. The object of the *Equipment Register* class, as far as possible all equipment, e.g. production, IT, climate control, etc., needs to be listed. In the low-level classes, e.g. the *Heater* class, a link to objects in the *Monitored Objects Directory* is added.

▼  Equipment Register	7754
▶  Compressor Equipment Dire...	7762
▶  Cooling Equipment Directory	7763
▶  Heating Equipment Directory	7755
▼  Lightning Equipment Direct...	7774
 Beleuchtungsanlage	7883
 Office Equipment Directory	7776
▶  Production Equipment Direc...	7775
▶  Ventilation Equipment Direc...	7767



**Figure 28 Equipment Register**

To determine performance indicators, these energy consumptions need to be linked to the production of the organization. In this context, the data situation and acquisition period are of vital importance. To simplify the process and give at least basic EnPIs a “Performance Directory” is created (Figure 29) with the same resolution as the energy bills with one value per month. This allows a basic calculation to develop an EnPI.

▼  Performance Directory	7941
 Produktion Kalt 2015	8058
 Produktion Kalt 2016	7942
 Produktion Kalt 2017	8056
 Produktion Warm 2015	8059
 Produktion Warm 2016	7943
 Produktion Warm 2017	8057

**Figure 29 Performance Directory**

Back in the *Site* class, there still exists the *Meetings Directory* (Figure 30) for meetings at the site and the *Monitoring Register* class, which will contain the data required for a monitoring system. The *Meetings Directory* class is almost identical to the *Superior Level Meetings* class but without the feature to create an *Initial Contact* object.

▼  Meetings in Zarrentin	7748
 1. Treffen vor Ort	7961

**Figure 30 Meetings Directory**

The *Monitoring Register* lists all the relevant data for an online monitoring system (Figure 31). This includes a list of the installation locations in the *Measuring Point Directory*, a list of the installed meters in the *Meter Directory* and a list of the installed stations, which fetches the data from the meters to a central database, in the *Station Directory*. A *Measuring Point* object in the *Measuring Point Directory* contains two link attributes for the connected *Meter* and *Station*.

---








▼	 Monitoring Register	7944
▼	 Measuring Point Directory	7945
	 Verdichter 1	7950
▼	 Meter Directory	7946
	 MDVH5181-M Nr. 1	7949
▼	 Station Directory	7947
	 Grundstation 1	7948

Figure 31 Monitoring Register

## 5.2 User Interface

The general consensus has been to use normal HTML files to present or input the data. This allows for a wider range of application as most electronic devices can open these files, and convert to pdf or simply print. There are many possibilities to easily visualize data with free and open source libraries, like jQuery, Charts.js and others. Figure 32 shows a first test on how to visualize data with texts, tables and charts with html and JavaScript with the help of some third party open source libraries:

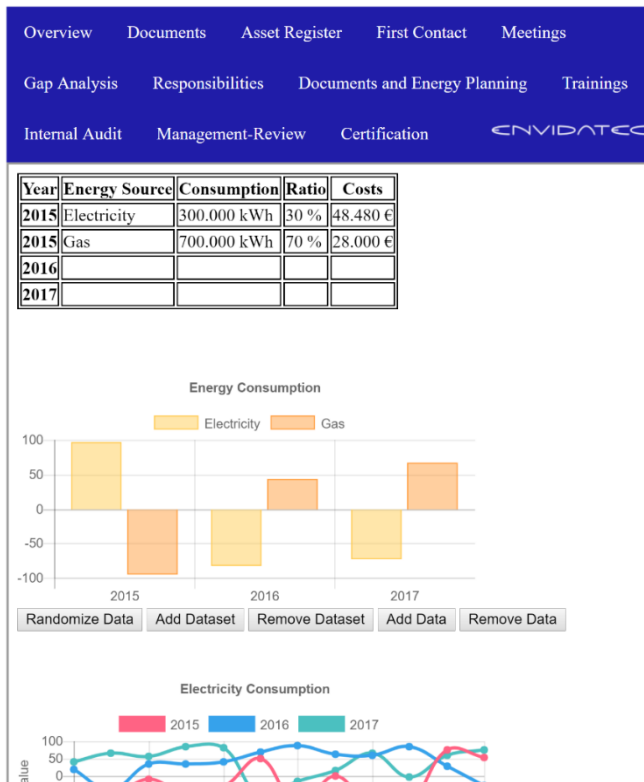


Figure 32 First Design tests with basic HTML and JavaScript

To move the data from the JEVIS System via the REST web service into the HTML file a transporter is needed, which is the template engine.

As it was decided to use a template engine to fill preformatted html pages to offer a wider variety of customization possibilities there are several selection options [11]. The first-choice criteria have been the programming language conforming to the rest of the JEVIS System and open source. The most notable conforming to these requirements are FreeMarker, Hamlets, Mustache and Thymeleaf. All of these are still being kept alive, except Hamlets. After evaluating their features and possibilities of use, FreeMarker has been chosen as the template engine for this project.

The data is prepared in Java, and then FreeMarker fills this data into the prepared HTML files.

# 6 Implementation

In this section of the thesis, the implementation is mainly done in NetBeans IDE [5]. The class structures are implemented and at the same time checked for their functionality. The FreeMarker template files are created by a standard editor. The following shows and describes some selected examples on how data processing and data output is performed. Traditionally the JEWebService output format is a JSON object [12]. The whole source code, available on a CD attached, will especially be moved to the OpenJEVis project on GitHub in due time.

## 6.1 Java Classes

At first the whole ISO 50001 JEVis class structure is converted to Java classes. This keeps the overview over the needed structures and values and increases the accessibility for other people and improves the ease of testing.

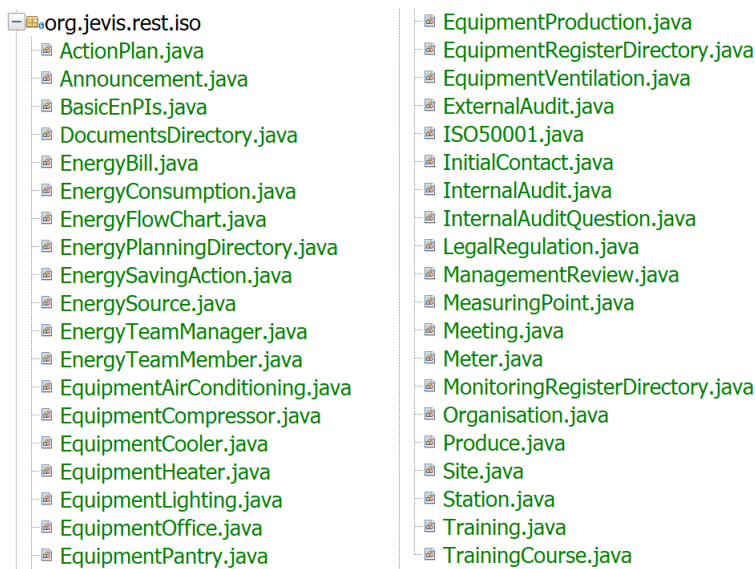


Figure 33 List of `org.jevis.rest.iso` classes



However, this adversely affects the performance and should therefore be further reviewed. There are two approaches: on the one hand the contents of the constructors can be moved to the get-Methods (getter), on the other hand it is possible to insert the code directly into the REST classes. There are mainly two types of classes, which can be differentiated into directories and objects in those directories. As an example, the *EnergySource* class is presented in the following paragraphs. The following Figure 34 shows the early constructor of the *EnergySource* class.

```
public class EnergySource {  
  
    private static final org.apache.logging.log4j.Logger logger = LogManager.getLogger(Ene  
  
    private String Name = new String();  
    private long ID = 0L;  
    List<EnergyBill> EnergyBills = new ArrayList<>();  
    List<EnergyConsumption> EnergyConsumptions = new ArrayList<>();  
  
    public EnergySource(JEVisObject input, JEVisClasses jc) {  
  
        try {  
            this.ID = input.getID();  
            this.Name = input.getName();  
  
            for (JEVisObject obj : input.getChildren(jc.getEnergyConsumption(), true)) {  
  
                EnergyConsumption ec = new EnergyConsumption(obj);  
                ec.setType(Name);  
  
                EnergyConsumptions.add(ec);  
            }  
  
            for (JEVisObject obj : input.getChildren(jc.getEnergyBills(), true)) {  
  
                EnergyBill eb = new EnergyBill(obj);  
  
                EnergyBills.add(eb);  
            }  
  
            for (EnergyConsumption ec : EnergyConsumptions) {  
                for (EnergyBill eb: EnergyBills) {  
                    if (eb.getYear() == ec.getYear()) {  
                        ec.setCosts(eb.getcosts());  
                    }  
                }  
            }  
        }  
    } catch (Exception ex) {  
        logger.catching(ex);  
    }  
}
```

Figure 34 First *EnergySource* class constructor

The *EnergySource* class contains two attributes, namely the *Name* and the *ID* and two lists, containing the energy bills and the energy consumption. To improve the performance for data requests, the constructor is changed slightly (Figure 35 EnergySource class, revised constructor): the loops are moved to respective getter (Figure 36). A new local variable for the *JEVisObject* *input* and the *JEVisClasses* *jc* has been created, which can be accessed by their getter.

```
public EnergySource(JEVisObject input, JEVisClasses jc) {  
    try {  
        this.ID = input.getID();  
        this.Name = input.getName();  
  
        this.object = input;  
        this.jc = jc;  
  
        List<JEVisAttribute> listEnergySourceAttributes = new ArrayList<>();  
  
        listEnergySourceAttributes = input.getAttributes();  
  
        for (JEVisAttribute att : listEnergySourceAttributes) {  
            String name = att.getName();  
  
            JEVisType jt = att.getType();  
            int primitiveType = jt.getPrimitiveType();  
            String GUIDisplayType = jt.getGUIDisplayType();  
            FormAttributeDisplayType fadt = new FormAttributeDisplayType(primitiveType, GUIDisplayType);  
  
            switch (name) {  
                case AttCO2EmissionFactor:  
                    this.setCO2EmissionFactor(Organisation.getValueDouble(att, 0.0));  
                    this.setCO2EmissionFactortype(fadt.getOutput());  
                    break;  
                default:  
                    break;  
            }  
        }  
    } catch (Exception ex) {  
        logger.catching(ex);  
    }  
}
```

Figure 35 EnergySource class, revised constructor

```
public List<EnergyConsumption> getEnergyConsumptions () {
    EnergyConsumptions.clear();
    try {
        for (JEVisObject obj : getObject().getChildren(getJc().getEnergyConsumption(), true)) {
            EnergyConsumption ec = new EnergyConsumption(obj);
            ec.setType(Name);

            EnergyConsumptions.add(ec);
        }
        if (EnergyBills.isEmpty()) {
            getEnergyBills();
        }
        for (EnergyConsumption ec : EnergyConsumptions) {
            for (EnergyBill eb : EnergyBills) {
                if (eb.getYear() == ec.getYear()) {
                    ec.setCosts(eb.getcosts());
                }
            }
        }
    } catch (JEVisException ex) {
        logger.catching(ex);
    }
    return EnergyConsumptions;
}
```

**Figure 36** Function for retrieving a List

The data input comes from the *JEVisObject* input and the *JEVisClasses* *jc* contains all ISO 50001 *JEVis* classes. The for loops get all children of the input object by the getter `getObject()`, which is the energy source passed from the energy planning class, that have the corresponding class defined in *jc*. This is the basic structure for nearly all directory classes which basically all classes in `org.jevis.rest.iso` adhere to. This *EnergySource* class is a hybrid class, as it is like a directory: it contains a list of energy consumption (Figure 37) and energy bills and there is one attribute for the CO<sub>2</sub> emission factor. All directory objects also have getter for the directory ID (Figure 38) and the directory name (Figure 39), usually one level higher in the class hierarchy. for example, a level above in the *Energy Planning Directory*.

```
public List<EnergySource> getEnergySources() {
    EnergySources.clear();

    try {
        for (JEVisObject obj : getObject().getChildren(getJc().getEnergySourcesDir(), false)) {
            for (JEVisObject m : obj.getChildren(getJc().getEnergySource(), true)) {
                EnergySource es = new EnergySource(m, jc);
                EnergySources.add(es);
            }
        }
    } catch (JEVisException ex) {
        logger.catching(ex);
    }
    return EnergySources;
}
```

**Figure 37** The getter for Energy Sources as a list

The object IDs and the object names are essential, because they are required for accessing the JEWebService with JavaScript on the later web pages. These are obtained by their respective getter (Figure 38 and Figure 39).

```
public long getEnergySourcesDirID() {
    try {
        for (JEVisObject obj : getObject().getChildren(getJc().getEnergySourcesDir(), true)) {
            EnergySourcesDirID = obj.getID();
        }
    } catch (JEVisException ex) {
        logger.catching(ex);
    }
    return EnergySourcesDirID;
}
```

**Figure 38** Getter for a directory ID

```
public String getEnergySourcesDirName() {
    try {
        for (JEVisObject obj : getObject().getChildren(getJc().getEnergySourcesDir(), true)) {
            EnergySourcesDirName = obj.getName();
        }
    } catch (JEVisException ex) {
        logger.catching(ex);
    }
    return EnergySourcesDirName;
}
```

**Figure 39** Getter for a directory name

The second type of classes mentioned above are classes, which are more object like: they represent a single object contrary to objects containing a directory. They have additional variables for the attributes. For example, the *Meeting* class (Figure 40).

```
public class Meeting {
    private static final org.apache.logging.log4j.Logger logger = LogManager.getLogger(Mee

    private final String AttContentOfMeetingAndResults = "Content of Meeting and Results";
    private final String AttMeetingDate = "Meeting Date";
    private final String AttMeetingParticipants = "Meeting Participants";
    private final String AttMeetingTime = "Meeting Time";
    private final String AttMinutesOfTheMeeting = "Minutes of the Meeting";

    private long ID = 0;
    private String name = new String();
    private String contentofmeetingandresults = new String();
    private FormAttributeType contentofmeetingandresultstype = Text;
    private String meetingdate = new String();
    private FormAttributeType meetingdatetype = Text;
    private String meetingparticipants = new String();
    private FormAttributeType meetingparticipantstype = Text;
    private String meetingtime = new String();
    private FormAttributeType meetingtimetype = DateTime;
    private File minutesofthemeeting;
    private FormAttributeType minutesofthemeetingtype = File;

    private Form form = new Form();
```

**Figure 40** First lines of the Meeting class

These classes list the relevant attribute names for obtaining data from the JEVIS System. The variables itself are of normal types like *String*, *Double*, etc. The *FormAttributeType* is explained later.

There are two sub-folders in the `org.jevis.rest.iso`, which are `add` and `rest`. In the `add` folder (Figure 41) there are a few specialized classes for data manipulation. At the moment this includes classes for charts, forms and tables as well as the basic template control. The template control is handled with the *TemplateChooser* and *TemplateManager* class. The *JEVISClasses* class contains the names of all ISO 50001 relevant classes, so changes made to the JEVIS class structure can easily be transferred.

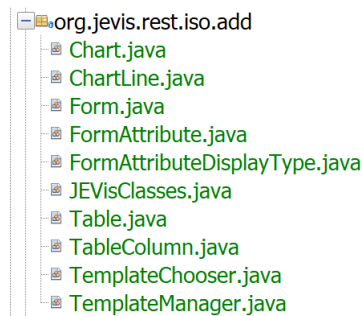


Figure 41 org.jevis.rest.iso.add classes

The *Chart* class handles a separate visualization approach for the REST service. It allows the visualization of data objects between two specified dates. This class shows what is possible with the template engine and appropriate chart libraries. The *Form* and *Table* classes and their subsidiaries are mainly for changing the data into a form that can be handled by the template engine.

As an example, the following figure shows the form creation process from the *InitialContact* class (Figure 42). The for loop iterates through all attributes of the *InitialContact* class object. As each attribute has its own unique format and type its

```
for (JEVisAttribute att : input.getAttributes(ID)) {  
  
    JEVisType jt = att.getType();  
    int primitiveType = jt.getPrimitiveType();  
    String GUIDisplayType = jt.getGUIDisplayType();  
    FormAttributeDisplayType fadt = new FormAttributeDisplayType(primitiveType, GUIDisplayType);  
  
    switch (att.getName()) {  
        case AttComment:  
            this.setComment(Organisation.getValueString(att, ""));  
            this.setCommenttype(fadt.getOutput());  
            break;  
        case AttDate:  
            String s = Organisation.getValueString(att, "");  
            this.setDateOfContact(s);  
            this.setDateOfContacttype(fadt.getOutput());  
            break;  
        case AttFile:  
            this.setContactFiletype(fadt.getOutput());  
            break;  
        default:  
            break;  
    }  
}
```

Figure 42 InitialContact class Form example

needs to be identified. The *FormAttributeDisplayType* can process the *primitiveType* and the *GUIDisplayType* from the *JEVisAttribute* into an understandable expression for the template engine. As there are some problems with the file input and output, it is not implemented yet.

Later a *Form* object is filled with the relevant data, which contains the attribute name, its type of value and its value (Figure 43).

```
List<FormAttribute> listFormAttribute = new ArrayList<>();
FormAttribute faComment = new FormAttribute(AttComment, Commenttype, Comment);
listFormAttribute.add(faComment);
FormAttribute faDate = new FormAttribute(AttDate, DateOfContacttype, DateOfContact);
listFormAttribute.add(faDate);
FormAttribute faFile = new FormAttribute(AttFile, ContactFiletype, ContactFile);
listFormAttribute.add(faFile);

this.form.setAttributes(listFormAttribute);
this.form.setID(this.getID());
this.form.setName(this.getName());
```

**Figure 43** InitialContact class Form example part 2

The *Form* class contains the `getOutput()` function, which uses the template engine to fill a specified preformatted HTML document, which is the standard input and output for all data (Figure 44).

```
public String getOutput() {
    String output = "";
    if (ID != 0 && attributes != null) {
        HashMap<String, Object> map = new HashMap<>();
        map.put("FormID", getID());
        map.put("name", getName());
        map.put("attributes", getAttributes());
        TemplateChooser tc = new TemplateChooser(map, "form");

        output = tc.getOutput();
    } else {
        output = "not enough data!";
    }
    return output;
}
```

**Figure 44** Form.getOutput() for further processing

## 6.2 Template files

The HTML documents for the form looks complex, because of the merging several programming languages, but its underlying principles are simple.

```
<#list attributes as att>
  <#if att.type == "Text">
    <tr>
      <td><p>${att.name}</p></td>
      <td><p><input type="text" value="${att.value}" id="id-${att.name?replace(' ', '')}" :
    </tr>
  <#elseif att.type == "TextArea">
    <tr>
      <td><p>${att.name}</p></td>
      <td><p> <textarea id="id-${att.name?replace(' ', '')}" rows="10" style="width: 100%;
    </tr>
  <#elseif att.type == "Date">
    <tr>
      <td><p>${att.name}</p></td>
      <td><p><input type="date" id="id-${att.name?replace(' ', '')}" value="${att.value}">
    </tr>
  <#elseif att.type == "Long">
```

**Figure 45** Excerpt of the Form Template for data input and output

`<#list attributes as att>` starts a list in the html document. All lines between `<#list ...>` and the closing `</#list>` is repeated for each object `att`, if the preselected `<#if att.type == "XXX">` condition is satisfied. This checks the attribute `type` to use the appropriate corresponding input type. After the FreeMarker template engine fills the form, the code of one filled row is shown in Figure 46.

```
<tr>
  <td><p>Meeting Date</p></td>
  <td><p><input type="date" id="id-MeetingDate" value="2016-02-11"></p></td>
</tr>
```

**Figure 46** Template row filled with data

Lastly the `<input type="button">` starts the function that sends the newly entered values to the JEVIS database. Future endeavors could integrate an `onchange()` event for dismissing the button. The next step is the execution of the function `sendToJEWebService()`, which is started by clicking the submit button.



```

function sendToJEWebService() {
  var bauth = "Basic FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF";
  var now = new Date();
  var nowFormatted = now.toISOString();

  <#list attributes as att>
  <#if att.type == "Text" || att.type == "Boolean" || att.type == "Double" || att.type == "Long" || att.type :
  var value${att.name?replace(' ', '')} = document.getElementById("id-${att.name?replace(' ', '')}").value;
  var xhr${att.name?replace(' ', '')} = new XMLHttpRequest();
  xhr${att.name?replace(' ', '')}.open("POST", encodeURI("http://localhost:6735/JEWebService/v1/objects/${FormID?c}"));
  xhr${att.name?replace(' ', '')}.setRequestHeader("Authorization", bauth);
  xhr${att.name?replace(' ', '')}.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
  json${att.name?replace(' ', '')} = [ { "ts": nowFormatted, "value": value${att.name?replace(' ', '')} } ];
  x${att.name?replace(' ', '')} = JSON.stringify(json${att.name?replace(' ', '')});
  xhr${att.name?replace(' ', '')}.send(x${att.name?replace(' ', '')});
  <#elseif att.type == "Date">

```

**Figure 47 JavaScript function send to JEWebService, one XMLHttpRequest**

This JavaScript function also uses the `<#list attributes as att>` from the template engine. It creates a basic XMLHttpRequest (xhr) for each attribute. Because attribute names can contain spaces, those need to be removed from the variable names. This is done with the FreeMarker built-in for Strings `?replace(search value, new value)`. This includes the URI for the upload, the RequestHeader and ContentType. As the JEWebService needs a JSON object for upload, the data needs to be converted first. Likewise, the date needs to be preformatted for the upload. With another function `deleteID` (Figure 48) an object can be deleted, with a different XMLHttpRequest.

```

function deleteID() {
  var bauth = "Basic FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF";
  var url = "http://localhost:6735/JEWebService/v1/objects/${FormID?c}";
  var xhr = new XMLHttpRequest();
  xhr.open("DELETE", url);
  xhr.setRequestHeader("Authorization", bauth);
  xhr.setRequestHeader("Content-Type", "application/json; charset=UTF-8");
  xhr.setRequestHeader("Accept", "application/json");
  xhr.send();
}

```

**Figure 48 JavaScript function deleteID() for object deletion**

## 6.3 Generation of browser pages on the server

The `org.jevis.rest.iso.rest` contains the classes which create the pages for the web server, which can be displayed by a regular web browser (Figure 49). These are evidently representing the directories from the standard class structure. These classes all share their similarities. What is different is mostly the query parameters and the template file which is used. These classes do HTML output formatting with FreeMarker template files.



Figure 49 `org.jevis.rest.iso.rest` classes

The main.java file behaves like the index.html in the traditional website construction (Figure 50). A web browser retrieves the list of *Sites* for the logged in user and sends them into the template.

```

@Path("/JEWebService/v1/main")
public class Main {

    private static final org.apache.logging.log4j.Logger logger = LogManag

    /**
     * @param httpHeaders
     * @return
     */
    @GET
    @Produces(MediaType.TEXT_HTML)
    public Response getObject(
        @Context HttpHeaders httpHeaders
    ) {
        JEVisDataSource ds = null;

        try {
            Login l = new Login();
            ds = new JEVisDataSourceSQL(l.getServer(), l.getPort(), l.getSe
            ds.connect(l.getDsUserName(), l.getDsPassword());

            ISO50001 iso = new ISO50001(ds);

            List<String> sites = new ArrayList<>();
            String firstSite = new String();
            firstSite = iso.getOrganization().getSite(0).getName();
            for (Site s : iso.getOrganization().getSites()) {
                sites.add(s.getName());
            }

            Map<String, Object> root = new HashMap<>();

            root.put("sites", sites);
            root.put("firstSite", firstSite);
            root.put("ISODirectoryID", iso.getOrganization().getID());

            TemplateChooser tc = new TemplateChooser(root, "main");

            return Response.ok(tc.getOutput()).build();
        } catch (Exception ex) {
            logger.catching(ex);
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).e
        } finally {
            Config.CloseDS(ds);
        } Figure 50 Class for the generation of the start page
    }
}

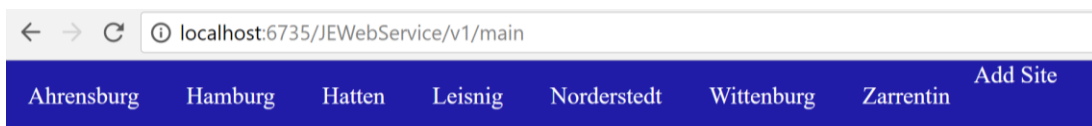
```

A special role is played by the Dashboard class. There is some data processing to get the data from the last three years, to get the costs and the CO<sub>2</sub> emissions and to sum up some data for comparisons. As by the ISO 50001 required, the current energy consumption is compared to an energy base line. Usually this energy baseline is valid for some time and three years validity is a common approach in various companies.

## 7 Demonstration

This section presents some application of the created software. The examples show the diversity of the possibilities. As each form is dynamically generated, the attributes can vary in their type.

First step is retrieving of the start page (Figure 51), where a company site can be selected.



**Figure 51** Site selection for from main.java

The underlying function retrieves all object of the *Site* class, which can be accessed by the logged in user.

If a site is selected in this interface, the software automatically loads the dashboard (Figure 52). The dashboard summarizes the energy consumption, the energy sources and the basic EnPIs.

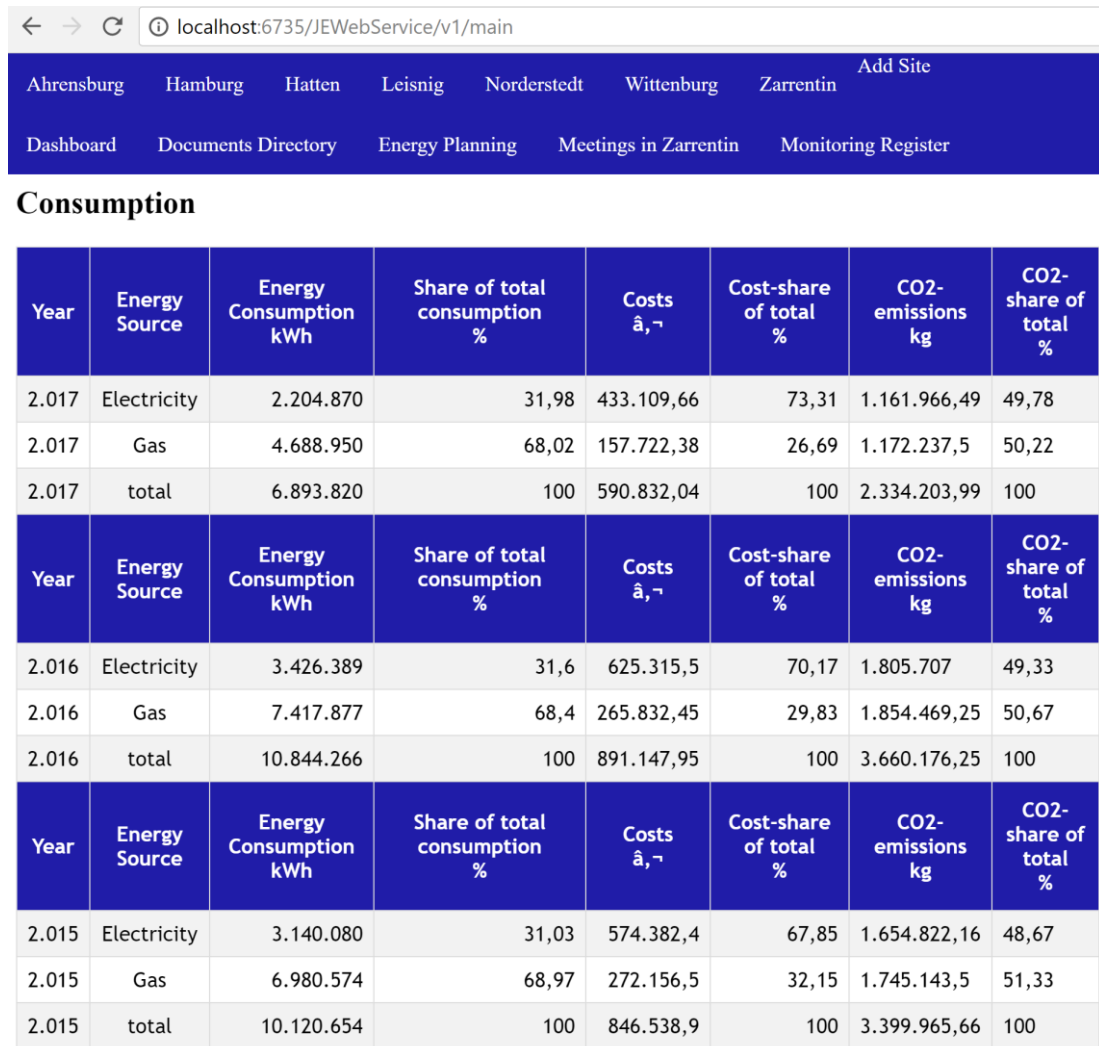


Figure 52 Dashboard with overview over the consumption

Figure 53 shows an excerpt from a conventional spreadsheet file from the traditional system of energy management. The values in the table are based on the values entered into the fields for energy consumption and energy bills in the energy planning section of the interface.

2015	Strom	3.175.485 kWh	31,14%	582.527,24 €	67,98%
2015	Gas	7.022.491 kWh	68,86%	274.383,18 €	32,02%
2015	Gesamt	10.197.976 kWh	100,00%	856.910,42 €	100,00%
2016	Strom	3.477.860 kWh	31,78%	637.126,32 €	70,40%
2016	Gas	7.465.491 kWh	68,22%	267.832,09 €	29,60%
2016	Gesamt	10.943.351 kWh	100,00%	904.958,41 €	100,00%
2017	Strom	2.204.870 kWh	31,98%	433.109,66 €	73,31%
2017	Gas	4.688.950 kWh	68,02%	157.722,38 €	26,69%
2017	Gesamt	6.893.820 kWh	100,00%	590.832,04 €	100,00%

**Figure 53 Comparison to conventional excel file**

The differences in 2015 and 2016 occur due to outdoor temperature correction on the values. In 2017 there are missing values for the last four months of the year. There is no temperature correction, so the values match.

Figure 54 shows the basic total energy performance indicators from the dashboard, while Table 4 shows the values from the conventional spreadsheet file, which are without temperature correction.

**Total EnPIs**

Year	January	February	March	April	May	June	July	August	September	October	November	December
2017	1,69	1,50	1,29	1,23	1,29	1,29	1,35	1,46	0,00	0,00	0,00	0,00
2016	1,67	1,64	1,42	1,36	1,43	1,42	1,63	1,48	1,44	1,58	1,50	1,43
2015	1,49	1,62	1,31	1,33	1,29	1,26	1,36	1,52	1,61	1,51	1,53	1,37

Figure 54 Basic EnPIs which are automatically calculated from consumption and production

Table 4 EnPI comparison for 2017 from conventional structure

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
<b>2017</b>	1,69	1,50	1,29	1,23	1,29	1,29	1,35	1,46				
<b>2016</b>	1,67	1,64	1,42	1,36	1,43	1,42	1,63	1,48	1,44	1,58	1,50	1,43
<b>2015</b>	1,49	1,62	1,31	1,34	1,30	1,26	1,36	1,52	1,61	1,51	1,53	1,37



Figure 55 shows a basic line chart, which is generated from the data values entered into the energy consumptions of the corresponding energy source.

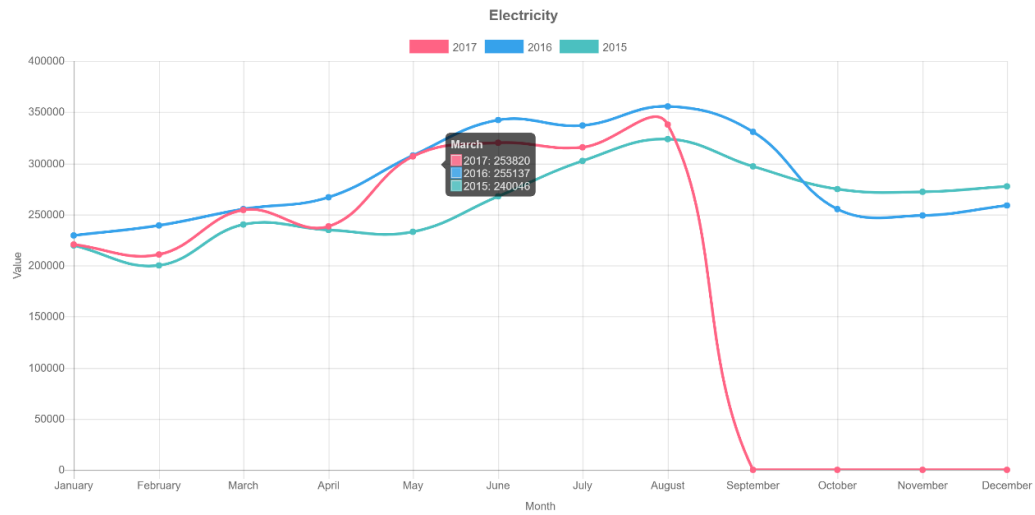


Figure 55 Chart generated for respective energy source

Figure 56 visualizes the basic total EnPIs calculated by the software solution in a line chart created with the help of chart.js library.

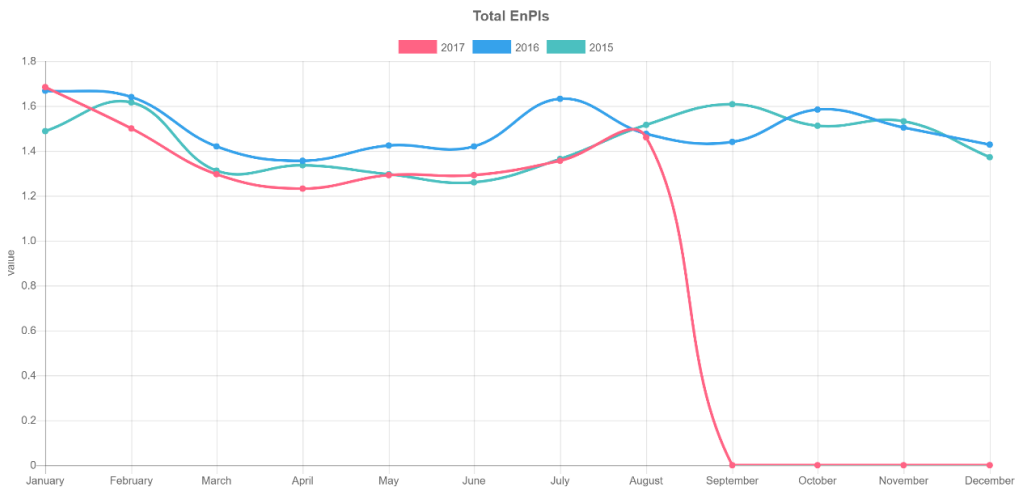


Figure 56 Chart visualizing the basic EnPIs

Opening a meeting allows the editing the previously defined object by the *JEVisClass Meeting* (Figure 57). The attributes Content of Meeting and Results, Meeting Date, Meeting Participants and Minutes of the Meeting speak for themselves.

<span style="float: right;">Add Meeting</span> 1. Treffen vor Ort	
Form ID:	7.961
Form Name:	1. Treffen vor Ort
Content of Meeting and Results	<div style="border: 1px solid #ccc; padding: 5px; min-height: 100px;"> <u>Inspection with project manager</u> </div>
Meeting Date	<div style="border: 1px solid #ccc; padding: 2px;">11.02.2016</div>
Meeting Participants	Mr. <u>X</u> Mr. <u>Y</u> Mr. <u>Z</u>
Minutes of the Meeting	<div style="border: 1px solid #ccc; padding: 2px;">                         Datei auswählen Keine ausgewählt                     </div>
<div style="display: flex; gap: 10px;"> <span style="border: 1px solid #ccc; padding: 2px 10px;">Submit</span> <span style="border: 1px solid #ccc; padding: 2px 10px;">Delete Form</span> </div>	

Figure 57 Form for editing a meeting in the meetings directory at the site

In the equipment register are the directories, which are distinguished by the usual technological distinctions. Figure 58 show the form for editing a cooler in the *Cooling Equipment Directory*. This object has attributes for technical aspects, and most importantly, a link to the corresponding *Measuring Point* in the *Measuring Point Directory* in the *Monitoring Register*.

Air Condition Equipment Directory	Compressor Equipment Directory	Cooling Equipment Directory	Hea
Pantry Equipment Directory	Production Equipment Directory		
Verdichter 1	Verdichter 2	Verdichter 3	Verdichter 5
Verdichter 6	Verdichter 7	Verdichter 8	

Form ID: 7.894

Form Name: Verdichter 1

Chiller Control:

Composite or single Plant:

Cooling Capacity:

Daily Operating Hours:

Energy Source:

Heat Recovery:

Manufacturer:

Measuring Point: 7836

Nominal Power:

Number of Compressors:

Performance Number Compressor:

Power:

Refrigerant:

Remark:

Figure 58 Form for editing a cooling equipment in the equipment register

As mentioned in chapter 5.1 mentioned, the ISO 50001 requires a regular evaluation of the energy management system. Figure 59 illustrates the possibilities of data a *Management Review* object in the *Management Review Directory* provides.

<a href="#">Audit Directory</a> <a href="#">Action Plan Directory</a> <a href="#">Announcement Directory</a> <a href="#">Energy Team Directory</a> <a href="#">Legal</a>	
<a href="#">Add Management Review</a>	
Management Review 2016	
Form ID:	7.940
Form Name:	Management Review 2016
Content	1. results of the internal audit of 28.09.2016 <u>The internal audit was passed in full, there are 3 recommendations and no deviations.</u> General recommendations: Registry must be updated (done 28.09.2016) List unallocated energy consumers Energy performance index â€ "elaborate details
Date	28.09.2016
Management Review File	<input type="button" value="Datei auswählen"/> Keine ausgewählt
Management Review PDF	<input type="button" value="Datei auswählen"/> Keine ausgewählt
Participants	<u>managing director</u> <u>energy manager</u>
<input type="button" value="Submit"/> <input type="button" value="Delete Form"/>	

Figure 59 Form for editing a management review

The legal regulations in the *Legal Regulation Directory* are periodically checked for changes in their texts and their significance for the company. The object in Figure 60 helps with the management of this process.

<a href="#">Audit Directory</a> <a href="#">Action Plan Directory</a> <a href="#">Announcement Directory</a> <a href="#">Energy Team Directory</a> <a href="#">Legal D</a>	
<a href="#">DIN EN ISO 50001:2011</a> <a href="#">EDL-G</a> <a href="#">EnEG</a> <a href="#">Add Legal Regulation</a>	
Form ID:	7.912
Form Name:	DIN EN ISO 50001:2011
Content Summary	<p><u>Requirements for an energy management system</u></p>
Date of Review	<input type="text" value="22.09.2017"/>
Issue Date	<input type="text" value="01.12.2012"/>
Last Amended	<input type="text" value="01.12.2012"/>
Regulation Designation	<input type="text" value="energy management system"/>
Relevance for ISO 50001	<input type="checkbox"/>
Significance for the Company	<p>Establishment of systems and processes that contribute to improving energy-related performance, including energy efficiency, energy use and energy consumption. It also aims to reduce greenhouse gas emissions, environmental impact and costs.</p>
<input type="button" value="Submit"/> <input type="button" value="Delete Form"/>	

Figure 60 Form for editing a legal regulation

A *Measuring Point* in the *Measuring Point Directory* contains useful data of its installation location (Figure 61), like the installed meter and to which station the meter is connected. This information is saved in the attributes *Meter* and *Station*, which contains the ID of the corresponding object. Another link to an object in the *Monitored Object Directory*, which is a directory from an established energy monitoring system, is possible.

Measuring Point Directory   Meter Directory   Station Directory	
Verdichter 1 <b>Add Measuring Point</b>	
Form ID:	7.950
Form Name:	Verdichter 1
Comment	<input type="text"/>
Data Point Assignment	Adresse 1 Register 01C1
Installation Location	Schaltraum Kälteanlage
Meter	7949
Monitoring ID	7836
Name	Verdichter 1
Photo	<input type="button" value="Datei auswählen"/> Keine ausgewählt
Physical Property	Energie
Station	7948
Unit	kWh
<input type="button" value="Submit"/> <input type="button" value="Delete Form"/>	

Figure 61 Form for editing a measuring point in the monitoring register

*Energy Sources* always have an *Energy Consumption* and an *Energy Bills* object which belong together. Figure 62 shows this form for simultaneously inputting data.

Electricity		Gas		Add Energy Source	
0	2015	2016	2017	Add Energy Consumption / Bills	
				Remove Energy Source	
Form ID:	8.049				
Form Name:	Energy Consumption 2017				
01 January	220786				
02 February	210880				
03 March	253820				
04 April	238347				
05 May	307027				
06 June	320268				
07 July	315987				
08 August	337755				
09 September	0				
10 October	0				
11 November	0				
12 December	0				
Energy Supplier					
Year	2017				
Submit		Delete Form			
Form ID:	8.052				
Form Name:	Energy Bills 2017				
01 January	43497,33				
02 February	41078,17				
03 March	49718,82				
04 April	46887,46				
05 May	63073,94				
06 June	59637,43				
07 July	60995,19				
08 August	68221,32				
09 September	0				
10 October	0				
11 November	0				
12 December	0				
Energy Supplier					
Year	2017				
Submit		Delete Form			

Figure 62 Form for editing the energy consumption and energy bills of an energy source

## 8 Conclusion and Outlook

The implementation of the objectives has been largely successful. There are some small cosmetic and functional parts, that should be improved or optimized. For example, the menu guidance via navbars and iframes and the file upload and download integration. A general uplifting of the layout with cascading style sheets (CSS) or for example bootstrap [13] could bring the interface to life.

There are still a lot of other topics, which were not part of the objectives but need to be improved or even implemented. For instance, the login process is at the moment hardcoded, which has to be changed to a defined system. This can stay as a basic authentication, but in this case, there needs to be a login page before the now existing components, which could be realized by a small webserver, e.g. Jetty or a similar small, lightweight server application for the authentication and get and post requests of the webservice.

Another possibility to improve security would be by implementing an OAuth2 authentication, which would increase the security level, but would require some more changes to the JEWebService. Because of the low priority and the few occasions that occur, the file input and output, from the user to the database and back, is not implemented, which needs to be addressed in the future. The cosmetic improvements that can be addressed are for example, the input and output form with `onchange()` events, which automatically update the data so the need for the submit button can be removed. Another one is the name of an object cannot be changed at the moment. This would drastically improve the usability. Further a preloaded list for the object target input boxes, so the user does not have to remember the object ID of the object he or she wants to link.



The energy management manual is not implemented yet and there are some small issues with the attributes of mixed object, like the *Energy Source* class. As they are handled like directories, there is no change to the attributes possible. The opposite occurs in the action plans, where the attributes are there but the sub-directories with the implemented and planned actions are missing.

*Energy Consumption* and *Energy Bills* could be merged on the form side. The objects need to be separate, but for ease of use, the form should only show one column for the attribute names and the second and third column *Energy Consumption* and *Energy Bills* respectively.

To improve the added value, unique characteristics and performance of the software, developing additional modules, like temperature adjustment for outdoor temperatures, etc. is important.

In light of the increasing complexity of the involved processes the created system should be moved out of the JEWService. The JEWService as a core component of the JEVIS System should be subjected to as few as possible changes. Creating a separate entity containing the new classes and functions will allow a much more flexible update and modification policy.

# 9 Appendices

## 9.1 Table of JEVIs Class Structure for ISO 50001

Description:

**Green:** classes

**Orange:** existing, modified, classes

**Red:** new classes

**Blue:** Attributes

Directories: in nearly all cases unique, with no Attributes

[objects\\_table\\_v01\\_00\\_20180206.xlsx](#)

## 9.2 Bibliography

- [1] Envidatec GmbH, company profile, <http://www.envidatec.com/kontakte/ueber-uns/>, retrieved December 18<sup>th</sup> 2017
- [2] DIN Deutsches Institut für Normung e.V, 2017 - E DIN EN ISO 50001:2017-09 ISO/DIS 50001:2017(E), Beuth Verlag (Hrsg.)
- [3] Frank T., Heinrich N., „OpenJEVis – Open-Source-Energiemonitoring“ in „Praxis Energiemanagement“, Ludger Pautmeier (Hrsg.), Köln, 2013, ISBN: 978-3-8249-1695-5
- [4] Tüzer Murat, August 2015, Planung und Implementierung einer
- [5] NetBeans IDE, official Java 8 IDE, <https://netbeans.org/>
- [6] Heinrich Nils, Envidatec GmbH, December 1<sup>st</sup> 2017, Meeting
- [7] Heinrich Nils, Simon Florian, Envidatec GmbH, December 3<sup>rd</sup> 2017, Meeting

- 
- [8] Heinrich Nils, Envidatec GmbH, December 4<sup>th</sup> 2017, Meeting
  - [9] Mertha Alexandra, October 2016, Analyse und Optimierung der zeitlichen Aufwände zur Einführung eines Energiemanagementsystems nach ISO 50001:2011
  - [10] Kortylak Tom, June 2017, Konzeption und Implementierung einer softwaregestützten Aufwandsoptimierung bei der Durchführung von Energieaudits nach DIN EN 16247-1 Energiemanagement-Software
  - [11] Heinrich Nils, Envidatec GmbH, December 15<sup>th</sup> 2018, Meeting
  - [12] Simon Florian, Envidatec GmbH, October 2014, JEWebService Version 3 - Specification
  - [13] Bootstrap, open source CSS framework, <https://getbootstrap.com/>

### 9.3 List of Tables

Table 1 Advantages and disadvantages of open source and proprietary software .	22
Table 2 ISO 50001 Steps from previous study [9] .....	23
Table 3 Currently existing file types in use-cases, file-based structure .....	25
Table 4 EnPI comparison for 2017 from conventional structure.....	56

### 9.4 List of Figures

Figure 1 Plan - Do - Check - Act Cycle illustration taken from the ISO 50001 [2].....	9
Figure 2 JEVIS System Architecture, Envidatec 2014 – The Open JEVIS System...	11
Figure 3 JEVIS database scheme .....	13

---

Figure 4 JEVisClass properties .....	14
Figure 5 JEVis classrelationship properties .....	14
Figure 6 JEVisObject properties .....	15
Figure 7 JEVis relationship properties .....	15
Figure 8 JEVisAttribute properties .....	16
Figure 9 JEVis type properties .....	17
Figure 10 JEVisSample properties .....	17
Figure 11 First Deliberations User Interface .....	20
Figure 12 Revised Concept of request chain .....	21
Figure 13 Organization with top-level objects .....	29
Figure 14 Site Object with its children .....	29
Figure 15 Documents Directory with its children .....	30
Figure 16 Action Plan Directory with examples for implemented and planned actions .....	30
Figure 17 Announcement Directory .....	31
Figure 18 Audit Directory .....	31
Figure 19 Energy Team Directory and Responsibilities .....	32
Figure 20 Legal Regulations Directory .....	32
Figure 21 Management Manual Directory .....	33
Figure 22 Management Review Directory .....	34
Figure 23 Procedural Document Directory .....	34
Figure 24 Training and Training Course Directories .....	34

---

Figure 25 Energy Planning Directory .....	35
Figure 26 Energy Sources Directory.....	35
Figure 27 Energy Flow Chart Directory.....	36
Figure 28 Equipment Register .....	36
Figure 29 Performance Directory.....	37
Figure 30 Meetings Directory .....	37
Figure 31 Monitoring Register .....	38
Figure 32 First Design tests with basic HTML and JavaScript .....	39
Figure 33 List of org.jevis.rest.iso classes .....	40
Figure 34 First EnergySource class constructor .....	41
Figure 35 EnergySource class, revised constructor.....	42
Figure 36 Function for retrieving a List .....	43
Figure 37 The getter for Energy Sources as a list.....	44
Figure 38 Getter for a directory ID .....	44
Figure 39 Getter for a directory name.....	44
Figure 40 First lines of the Meeting class .....	45
Figure 41 org.jevis.rest.iso.add classes.....	46
Figure 42 InitialContact class Form example .....	46
Figure 43 InitialContact class Form example part 2 .....	47
Figure 44 Form.getOutput() for further processing.....	47
Figure 45 Excerpt of the Form Template for data input and output.....	48
Figure 46 Template row filled with data .....	48

---

Figure 47 JavaScript function send to JEWebService, one XMLHttpRequest.....	49
Figure 48 JavaScript function deleteID() for object deletion.....	49
Figure 49 org.jevis.rest.iso.rest classes.....	50
Figure 50 Class for the generation of the start page.....	51
Figure 51 Site selection for from main.java.....	53
Figure 52 Dashboard with overview over the consumption.....	54
Figure 53 Comparison to conventional excel file.....	55
Figure 54 Basic EnPIs which are automatically calculated from consumption and production.....	56
Figure 55 Chart generated for respective energy source.....	57
Figure 56 Chart visualizing the basic EnPIs.....	57
Figure 57 Form for editing a meeting in the meetings directory at the site.....	58
Figure 58 Form for editing a cooling equipment in the equipment register.....	59
Figure 59 Form for editing a management review.....	60
Figure 60 Form for editing a legal regulation.....	61
Figure 61 Form for editing a measuring point in the monitoring register.....	62
Figure 62 Form for editing the energy consumption and energy bills of an energy source.....	63

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den \_\_\_\_\_