Development of a User Interface for Visualization of the Potential of Building Energy Management Algorithms

Bachelor Thesis

by

Lukas Landwich

Degree Course: Computer Science B.Sc.
Matriculation Number: 1974749

Institute of Applied Informatics and Formal Description Methods (AIFB)

KIT Department of Economics and Management

Advisor:	Prof. Dr. Hartmut Schmeck
Second Advisor:	Prof. Dr. Andreas Oberweis
Supervisor:	M.Sc. David Wölfle
Submitted:	February 16, 2021

Abstract

Most of the countries worldwide have agreed on a climate protection law. In Germany, the Bundes-Klimaschutzgesetz states that the "greenhouse gas emissions will be gradually reduced compared to 1990. By the target year 2030, a reduction rate of at least 55 percent is required". To achieve this goal, energy production is shifted to renewable energy sources and the total energy consumption is lowered. Buildings are responsible for 36 % of the global final energy use. Therefore, it is reasonable to utilize modern technologies to address this challenge in buildings. Building energy management systems are modern computer systems that manage energy usage, production and purchase and communicate with the building's devices and the energy market. These highly complex systems require configuration by the building's residents. Therefore, the understanding about these systems has to be raised so that the residents are capable of using these systems in the most effective way. This thesis will canvas the challenges of building energy management systems and, thereof, develop a user interface for visualization of the potential of building energy management algorithms. To do so, scenario buildings and use cases will be created and modern user interface design practices will be investigated. With their help, existing building energy management interfaces will be evaluated. The results are used to build this thesis' user interface for visualization of the potential of building energy management algorithms.

Contents

1 Introduction

1.1 Motivation

Germany, as well as many other countries worldwide, have agreed on a climate protection law [52]. In Germany the Bundes-Klimaschutzgesetz (KSG) states in § 3 Abs. 1 KSG [49] that the "greenhouse gas emissions will be gradually reduced compared to 1990. By the target year 2030, a reduction rate of at least 55 percent is required". This on one hand will be approached with the shift from fossil fuels to renewable energy sources [6], on the other hand with reducing the energy consumption itself [6]. The shift from fossil energy sources to renewable energy sources also represents a shift to distributed energy generation systems [43], like photovoltaic systems, wind turbines and biomass plants. These systems, especially photovoltaic systems and small combined heat and power plants, can also be used in buildings [23]. Buildings which are responsible for 39 % of the energy-related carbon dioxide emissions and 36 % of the global final energy use [1], become so-called *prosumers* [29], producing their own energy. Consequently, that shift raises a complete new set of challenges, like volatile energy supply because of uncertain power generation, depending on the weather or daytime, and, as a result, varying energy costs, volatile voltage issues and uncertain energy use and production forecasts [55] [54]. One way to approach these challenges are Building Energy Management Systems that manage energy usage, production and purchase, communicating with the building's devices and the energy market to achieve this goal [36]. It is in the nature of things that residential comfort such as thermal comfort comes with a high energy consumption which alone accounts for 77 % of the total energy consumption in buildings [12]. Hence, (thermal) residential comfort and lowering energy usage conflict each other. Additionally, this leads to a multi-objective optimization problem [47]. Optimizing such a problem results into a pareto optimum [53], a set of optimized data where no data point can be more optimized without deteriorating other data points. Algorithms like the analytic hierarchy process [44] can be used to solve these problems, but often require that the user has a precise idea of criteria weightings and the impact of his decisions in this context. Therefore, the development of a user friendly dashboard would be beneficial to raise the users' awareness about Building Energy Management Systems and their potential. Hence, this thesis shall be devoted to the development of such a user interface for visualization of the potential of these building energy management algorithms.

1.2 Objectives

The process of analyzing the impact and potential of Building Energy Management Systems is a complex objective. There are large sets of data to analyze, multiple algorithms

and settings to simulate, and the data monitoring metrics may be very complex. A simple user interface and data visualization should help users to achieve this goal. Therefore, a user interface for visualization of the potential of building energy management algorithms will be developed in this thesis. Thus, the challenges of energy management systems and the systems' user interfaces need to be analyzed. Based on the analysis' results a web application will be designed according to modern usability standards and design best practices. This application will be part of the Energy Management Panel (EMP) of the Forschungszentrum Informatik (FZI) in Karlsruhe, Germany. The designed application has to be implemented with the help of the Django web framework, since the application will be part of FZI's EMP. Afterwards, the implemented application needs to be analyzed with the findings of this thesis' research.

1.3 Thesis Structure

The following chapter 2 will introduce important concepts of building energy informatics and the challenges of building energy management. In the second half of chapter 2, building scenarios will be introduced. These shall serve as illustration of the introduced concepts. In chapter 3, user interface design best practices will be established. With these guidelines and the scenarios, state of the art building energy management applications will be evaluated in chapter 4. The evaluation results shall point out where energy management applications may be improved and which features are mandatory for a well-designed energy management user interface. In chapter 5, the findings of the previous chapters will be used to develop a user interface for visualization of the potential of building energy management algorithms. The resulting application will be evaluated in chapter 6. Finally a conclusion will be drawn in chapter 7, ending this thesis by giving an outlook to possible further work.

2 Concepts and Scenarios

This chapter provides basic concepts and definitions laying the theoretical groundwork for further development of a user interface for visualizing the potential of building energy management algorithms. Terms and basics of energy systems, (smart) buildings, their energy consumption and production, and their challenges will be defined.
In this chapter's second part scenarios and scenario use cases will be crafted out of these basic concepts. The use cases will serve as a starting point for the development of a user interface for visualizing the potential of building energy management algorithms and later on as help for evaluating the implementation.

2.1 Basic Concepts

While defining some basic concepts, an overview of the challenges that come with building energy management is to be created. This overview should clarify why it is helpful to use building energy management systems and which challenges these systems bring to inexperienced users and, therefore, why it is reasonable to create a user-friendly interface that visualizes the data and concepts listed here in a simple and easy-to-understand manner.

Building Situation in Germany

In Germany 21,7 million buildings exist [12]. The 19 million residential buildings[9] make up for 64% of the building sectors energy use, the other 36% belong to the non-residential buildings which only make up for 12% of all buildings in Germany [12]. Therefore, it is reasonable to take a closer look on both residential and non-residential buildings.
65% of the residential buildings are single family buildings and 6% are multi-family buildings with seven or more apartments [9]. But the 7+ multi-family buildings include nearly one third of all apartments, which makes them a not neglectable building type.
In matters of non-residential buildings office and administration buildings are the biggest share with about 27% of all non-residential buildings in German [31].

Energy Usage in Buildings

As buildings are responsible for 39% of the energy-related carbon dioxide emissions and 36% of the global final energy use [1] it is reasonable to take a closer look on where these numbers come from. Looking inside the building 39% of the energy consumption is *mechanical energy* [15], followed by 27% for *room heating*, and 22% for *process heating*.[12].

Because they make up for 88% of buildings energy consumption these categories will serve as basis for this thesis work, ignoring other energy consumption sources.

Devices

To review and process energy consumption precise data are required. With the technology of *smart meters* [4] it is possible to record the energy usage building wide and for each device permanently and query it in small time intervals like every five or fifteen minutes. In combination with basic computer systems and databases thus it is possible to create energy *consumption histories* for every device and the whole building[16].
In a smart home context not only the devices energy consumption is monitored, the devices are also able to connect to the home's network [45]. This enables device controlling like switching a device on or off and *device scheduling* like setting a time when the device will switched on or off or changes its state.
This information enables computer systems to *forecast* energy consumption[16]. A simple example my be to rely on human habits and forecast that at a given day the same amount of energy is needed as the day before.

Energy Production, Energy Storage, and the Energy Market

As buildings consume energy their owners need to buy energy or produce it themselves. As *energy market* we understand the suppliers the consumer buys energy from. The consumers are able to receive an energy price whereby the suppliers are able to increase or decrease the prices to give an incentive to use more cheap energy [36] [48]. This gets even more interesting when the building itself produces energy via a photovoltaic plant or a co-generation plant. The owner then has to decide if he wants to use his produced energy or buy cheap energy of the market and otherwise, if he wants to use his energy or sell it, when the energy price is at a high level.

Buildings as Energy Producers

As mentioned in the section before buildings more and more often produce their own energy with the help of modern technologies. With the use of *photovoltaic systems*[42] on the top of buildings or *combined heat and power plants*[41] [15] building owners cover parts of their energy consumption. In Germany the share of photovoltaic systems increases slowly but steadily every year [19].
Giving flexibility to the energy distribution and utilization, *energy storages* are installed in buildings [16]. These storages provide the freedom of storing or selling produced energy, but also add another complex parameter to the challenge of managing building energy

usage, since owners now need to decide if to use, sell, buy, or store produced energy.
In addition it is possible to have electric vehicles as kind of dynamic energy storages that may be plugged into the buildings energy grid [27]. They can be charged with abundant energy or even decharged when energy is needed, adding another decision factor to the building's energy system.

(Building-) Energy Management System

With the building energy grid environment getting more and more complex so-called *(Building-) Energy Management Systems* are used. These are computer based systems that coordinate the procurement, distribution, and utilization of energy in order to cover requirements taking ecological and economical objectives into consideration [16]. They have access to buildings energy data and are able to control electric devices. Out of collected data and user set parameters an energy management system computes *schedules* when to run devices, sell, store, use, or buy energy, and keeping track of energy consumption. This is accomplished with the help of optimization algorithms and energy management simulations [25].

Optimization Categories

The energy management systems need the users' input to determine what needs to be optimized[3]. Therefore, the two categories *economical optimization* and *ecological optimization* are intruoduced.
In terms of economical optimization the energy management system optimizes energy cost with the help of device scheduling in energy cost lows and by reducing the energy consumption as a whole. In terms of ecological optimization lowering energy consumption and using as much regenerative energy as possible is the way to go.
Beside the basic ecological and economical objectives of an energy management system this thesis also discusses *residential comfort optimization* as third objective of energy management systems. As residential comfort we understand three subcategories: thermal comfort, illumination, oxygen saturation. Especially, in commercial buildings comfort is worth optimizing since it is directly linked with the employee satisfaction and thus the working moral [22].

2.2 Scenarios

Based on these concepts scenarios now will be created and use cases will be build assisting the development of a user interface for visualizing the potential of building energy management algorithms. These use cases will describe different data evaluation processes

which need to be supported in the application. These scenarios and their use cases will be used in chapter 4 to analyze some of the state of the art implementations introduced there. Also the use cases are used to implement the data visualization elements of this thesis. This implementation from chapter 5 will be evaluated with the help of the scenarios in chapter 6.
To select appropriate scenarios their buildings will represent statistically average buildings in Germany.

2.2.1 Scenario Buildings

The data of section 2.1 would seem to suggest to build scenarios around three types of buildings. Single family buildings because they make up the biggest share of all residential buildings. Multi-family buildings with more than seven apartments, because they include a big share of all apartments in multi-family buildings. Also they represent a whole different setting than a single family building or a two apartment building, for example. As last scenario building type we choose office buildings for the same statistical reasons, they make up the biggest share in non-residential buildings.
In the following, out of these building types three scenario buildings will be constructed using average building key figures. Furthermore, each building will be equipped with a set of devices, which are equipped with a smart meter and are able to be controlled via the network. To each setting a optimization goal will be inferred. In general all buildings are equipped with smart meters measuring energy consumption and an energy management system with a user interface, enabling home owners to configure and monitor their system.

Single Family Building The average single family building in Germany has 127,2 square meters of living space distributed among 4,5 rooms [30]. About 79% of all single family building residents own a bicycle and a car [30].
To adapt this scenario to the subject of this thesis the house additionally is equipped with a photovoltaic system on the roof, a battery, a heat pump, and a thermal accumulator. This enables the residents, producing and consuming their own energy with the goal of energy autarky. Also it is assumed that the car is an electric vehicle that can be plugged into the building's energy grid. Thereby, it can be used as a dynamic battery. In this scenario the emphasis is on using the energy market price signals, deciding whether to sell or store generated energy resulting in optimized economic key figures [36].
Therefore, the energy management system needs to know the energy consumption of this building, as history, actual value and as forecast. The forecast will be calculated with the help of the consumption history and other data like the weather forecast or energy management device setpoints. Additionally, a price signal from the energy market is needed. As last important key figure the charge level of all available energy storages is

required.

Multi-Family Building The multi-family building is a modern smart home building with ten apartments which each have a statistically average living space of 68,9 square meters [9] distributed over three rooms. It is assumed that all devices owned by the residents are smart appliances, therefore, they can be controlled by the energy management system. Each household is equipped with a washing machine, a dishwasher, and a dryer. These devices are statistically owned by more than 65% of all German households [30]. Because it is not necessary to run these devices immediately, it lies in the nature of things to schedule these [10]. Hence scenario focus lies on optimizing device scheduling to shift energy consumption peaks into energy price lows decreasing residents' energy costs [34]. Therefore, energy consumption data are required as well as energy market data. In addition, setpoints for scheduling devices are needed to make the scheduling decisions (e.g., dishwasher has to be finished at 9:00 pm).

Office Building An average building following the model of the dena office real estate insight study [28] will serve as scenario office building. The building has 1200 square meters of office space. Because heating, ventilation, lighting, and water heating make more then 85% of office buildings' energy consumption [46] all other electric devices will be omitted. Therefore, it is assumed that the office building is equipped with a heating, ventilation, and air conditioning system (HVAC), electric controllable shutters, and system controlled lights. For thesis reasons it is also assumed that employees book a room via a booking system.
The main focus of this scenario lies on lowering energy consumption and with it the energy cost while keeping a perfectly comforting work environment as this has a strong impact on the productivity of workers[22]. In particular comfort in terms of thermal comfort, illumination comfort and oxygen saturation will be supported
Therefore, again energy consumption and market data are required. The energy management system requires data of light, oxygen, and temperature sensors as well as preferred setpoints for each key figure (e.g., 21°C in office #4 at 9:00 pm).

A listing of all data used in the scenarios can also be found in 1 where also a default measure and an example value is given. The data is split into each scenario. In the implementation process the listed data will be used to specify detailed use cases and to derive UI elements.

Scenario	Data Name	Formula	Unit	Example
Single Family Building				
	Energy Consumption	-	kW/h	42 kW/h
	Energy Price	-	ct/kW/h	30 ct/kW/h
	Heat Demand	-		
	Energy Generation	-	kW/h	24kW/h
	Dynamic/Static Battery Power	-	Ah	110Ah
	Temperature (outside)	-	°C	28°C
	Wind	-	km/h	10 km/h
	Solar Radiation	-	W/m^2	1 W/m^2
	Energy Cost	Consumption * Price	€	35.00 €
Multi-Family Buidling				
	Energy Consumption	-	kW/h	42 kW/h
	Energy Price	-	ct/kW/h	30 ct/kW/h
	Energy Cost	Consumption * Price	€	35.00 €
	Device Staus	-	Boolean Value	On/Off
Office Building				
	Energy Consumption	-	kW/h	42 kW/h
	Energy Price	-	ct/kW/h	30 ct/kW/h
	Energy Cost	Consumption * Price	€	35.00 €
	Illumination	-	$lx = lm/m^2$	500 lx [7]
	Temperature (inside)	-	°C	21°C
	Heat Demand	-	W	68.5 W
	Oxygen Saturation	-	%	97.00%
	Shutter State	-	Boolean Value	On/Off
	Thermal Comfort	[8] [35]	-3 to +3 or %	20%
	Light Demand	-	lx	500 lx
	Oxygen Saturation	-	%	20.95 %
	Carbon Dioxide Emission	401g CO2/kW/h [26]	g	1564kg/Jahr

Table 1: Scenario Data Overview

2.2.2 Use Cases

In the following the scenario buildings and associated data will be used to define energy management interface use cases. These will be used to evaluate the state of the art projects detecting their strength and weaknesses. Furthermore, the collected information is used to assist the development process of a user interface for visualizing the potential of building energy management algorithms.

UC01: The owner of the single family scenario building likes to use as much of his produced energy as possible. But also has in mind that his system may buy ore sell energy when energy prices are high or low. For these reasons he configured his own dashboard. With the help of energy management systems' admin page he created and configured database objects that represent the dashboard pages. He uses the so created user interface elements to monitor his energy consumption, the energy market price, and the composition of his energy consumption.
With the help of the admin panel he is also able to add a new device data source to the building's energy management system. He then has to create and configure a Datapoint object which he is able to include in his dashboard.

UC02: The residents of the multi-family scenario building have the possibility to configure an energy management system that manages their apartment for their own needs. One resident uses the energy management system to analyze his apartments data.
He uses the energy management system's dashboard pages to monitor his consumption with the help of simple card data visualizations and more complex charts. These UI elements are configurable according to his will. Therefore, he is able to choose the time intervals of the data sets shown in charts or the data sets used to create these charts. With the help of these elements he will compare his consumption and the energy market peaks. Thereby, he recognizes issues and configures the energy management system.
With the help of these user interface element features he is able to comprehend the system's decisions and adjust the configuration if he is unsatisfied with the outcome.

UC03 The energy management system manager of the company running the office building has the goal to higher the work environment comfort with the lowest possible energy cost. In a weekly cycle he changes optimization parameters and monitors the changes with the help of the comparison tool. Therefore, he configures what aspects are to be compared. Besides simple comparison of data the system offers him a special comparison charts that show up the differences between two configurations or algorithms.
Since he is about to analyze complex system circumstances he might need more

then just simple values presented as plain values or charts. Therefore, he is able to register metrics that which are applied to these values.
He is also able to create register new algorithms with the help of the admin panel.

3 Visualization Best Practices

To support the development of an user interface for visualization of the potential of building energy management algorithms, this chapter will introduce user interface design best practices. They will be used to evaluate state of the art applications in chapter 4, design this thesis' web application in chapter 5, and later on evaluate this implementation in chapter 6.
One of the most cited and referenced authors is Jacob Nielsen. Both older and newer studies like [24],[11], [50], and [2] use the groundbreaking work of him. Therefore, it is reasonable to use his best practices when designing a web application. Based on a study from Rolf Molich and Jakob Nielsen [38] and other design practices studies, Nielsen lists in a succession study [39] 101 design best practices. Despite this he also published 113 Design Guidelines for Homepage Usability [40] but there are also other guidelines for user interface development. Google's material design [21] or Facebook's brand resources [17] are some of them. But the of focus these guidelines mainly lies on user interface element design not on usability. Therefore this thesis will rely on Nielsen's heuristics for design and evaluation. In the following the ten most important of these heuristics will be listed as citation [39]:

DP01: Simple and Natural Dialogue "Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their visibility. All information should appear in a natural and logical order."

DP02: Speak the User's Language "The dialogue should be expressed clearly in words, phrases and concepts familiar to the user, rather than in system-oriented terms."

DP03: Minimize the User's Memory Load "The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate."

DP04: Consistency "Users should not have to wonder whether different words, situations, or actions mean the same thing."

DP05: Feedback "The system should always keep users informed about whats going on, through appropriate feedback within reasonable time."

DP06: Clearly Marked Exits "Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue."

DP07: Shortcuts "Accelerators - unseen by the novice user - may often speed up the interaction for expert user such that the system can cater to both inexperienced and experienced users."

DP08: Good Error Messages "They should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution."

DP09: Prevent Errors "Even better than good error messages is a careful design that prevents a problem form occurring in the first place."

DP10: Help and Documentation "Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, be focused on the user's task, list concrete steps to be carried out, and not be too large."

4 State Of The Art

This chapter introduces relevant related applications that provide user interfaces for building energy management systems. Since there are many different approaches in developing such an application, we will take an closer look on three applications which were often quoted, for example by [3]. The user interfaces of these applications will be analyzed using the scenarios and their use cases from chapter 2.2 and the visualization best practices from chapter 3. The best aspects and features will be used later on to develop the user interface for visualization of the potential of building energy management algorithms. The negative aspects shall show what should be avoided during implementation.
It is important to point out that the listed approaches never meet all requirements raised out of the scenarios and also implement features never used in this thesis' use cases. Therefore, only the important user interface features will be listed and analyzed.

4.1 Approaches

OpenHAB

The Open Home Automation Bus (openHAB) is a Java based software solution for home automation [33]. It is developed with the goal of creating a protocol and manufacturer independent framework. Therefore, it is open source and thus easily extendable. Since it is extendable there is no such thing as the one openHAB interface, therefore, the openHAB demo application [32] for web browsers will be evaluated.
OpenHABs demo page consists of four pages: a chart overview, a map with positions, building floor plans, and a temperature dashboard. Since this thesis is not about connecting with other buildings the map will be omitted as well as the floor plans since there are existing features in FZI's EMP.

Temperature Dashboard First screen to evaluate is a temperature dashboard as seen in figure 1. It is kept simple, consisting out of cards containing the temperature values of every room. Also headlines structure these cards assigning them to a floor. Located at the bottom of the page are some buttons containing logic to interact with the automation system. Nearly every card can be clicked to show more information or additional functionality. To pick up the visualization best practices from chapter 3: The screen is kept simple (DP01), is consistent in data visualization and in terms of getting additional information (DP04), and also gives feedback about the systems state (DP05) when no information is available, see corridor's temperature. But it is hard to read the temperature graphs in the card background because of the missing chart axis (DP02).

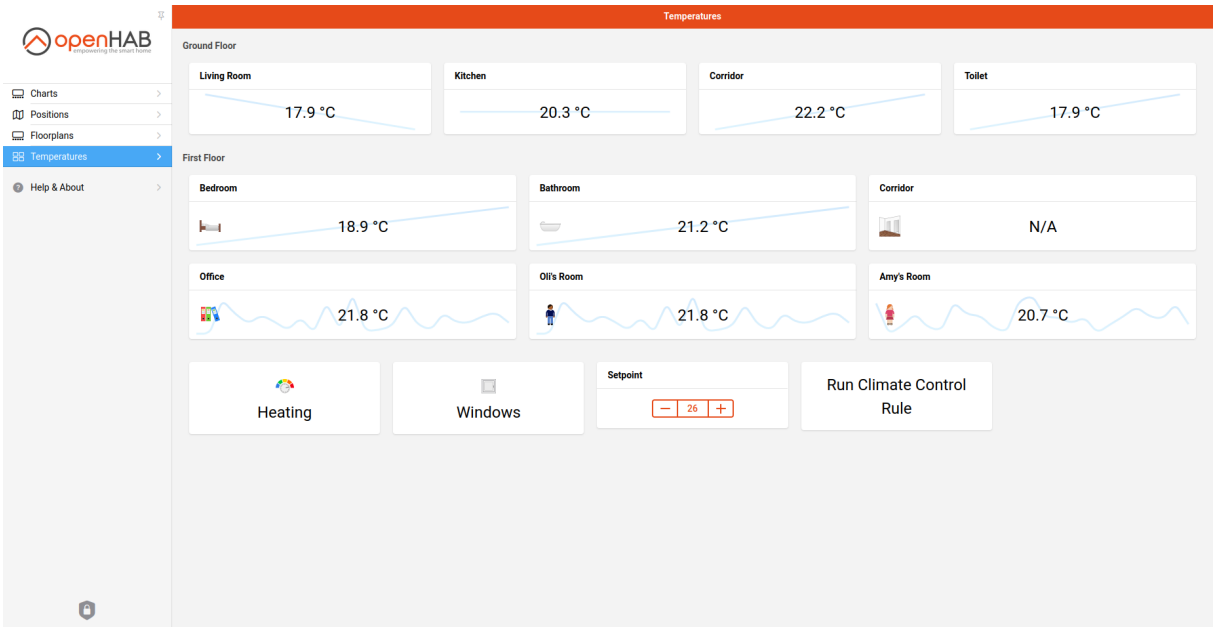

Figure 1: openHAB temperature dashboard

Graphs Looking at the graphs two of four were chosen: A temperature comparison graph and a temperature heat map. The comparison graph, as seen in figure 2, maps hourly values of a room and distinguishes between current value and yesterdays value and a third one that has no tooltip or description violating DP10. But the graph is once more kept simple (DP01) and speaks the users language (DP02) since it is a basic area graph that every user should be able to read. The simplicity disappears when taking a look at the temperature heat map at figure 3. This chart maps every hour of every weekday onto a color that represents the room temperature. On the first look this comes very confusing (DP01). Taking a closer look one can notice by hovering over these color coded temperatures that some color changes like from blue to orange represent a temperature change of less then 0.5°C. This is not very intuitive (DP02). Also one has to hover every color to see the temperature value, so a user has to remember its value while comparing temperatures (DP03), making this a very poorly designed user interface for beginners or first time users.

When these page are used in the scenario use cases context one has the possibilities to monitor his buildings data in various ways and with many details. But when it comes to configuration the user freedom ends. There are only preset data configurations in graphs, for example there is no possibility to change intervals. Speaking of configuration there is another big point: The addition of new data points. This is made possible by the framework but requires "some hacking skills" as the developers state [33]. This makes is harder for beginners or users with little knowledge of the context to add new data to their dashboards.

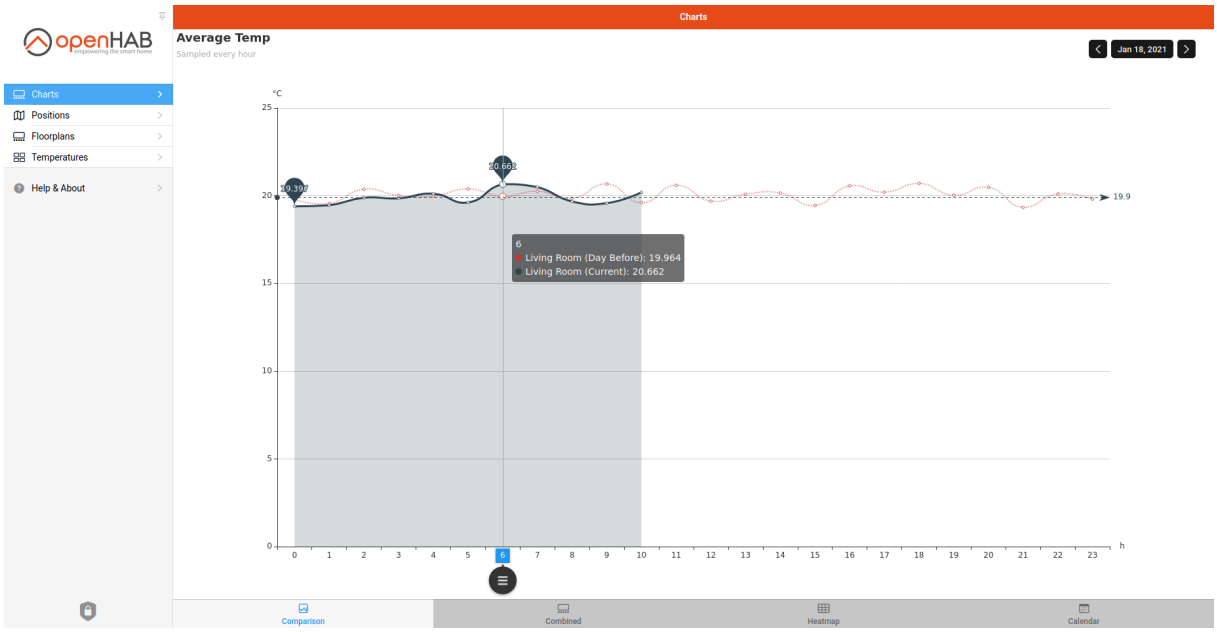

Figure 2: openHAB temperature chart

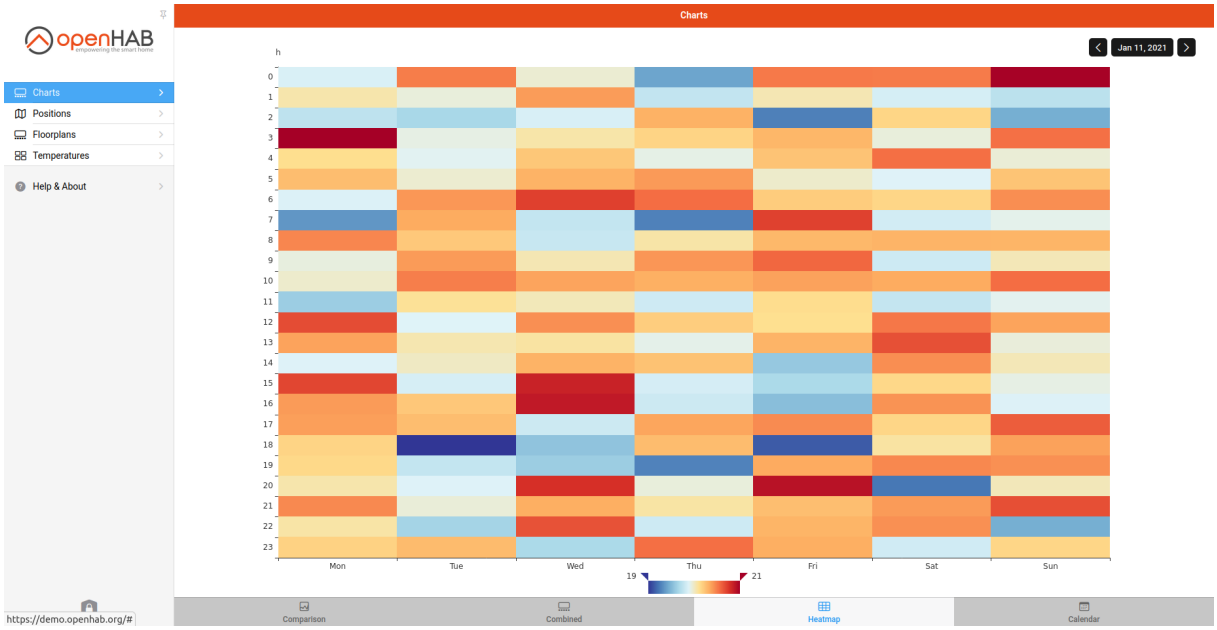

Figure 3: openHAB heatmap

Wiser Energy

Next the Wiser Energy mobile app [14] from Schneider Electric [13] was chosen and will be evaluated. The application has five pages, four of them will be evaluated, because the last one is about push notifications that will not be necessary in this thesis web application

development.

Dashboard At the dashboard, in figure 4, that functions as mobile app home page multiple data is presented. First of all weather data, then a system status, and a slider consisting of four items with different data like energy charge this month, energy flow or charging status of a electric vehicle. It may or may not be important to a user to see the weather first (DP01), but it is reasonable to show the systems status at the front page(DP05). To use a slider for important front page information whereas is not(DP03), because users are therefore forced to monitor only one piece of data at the time.

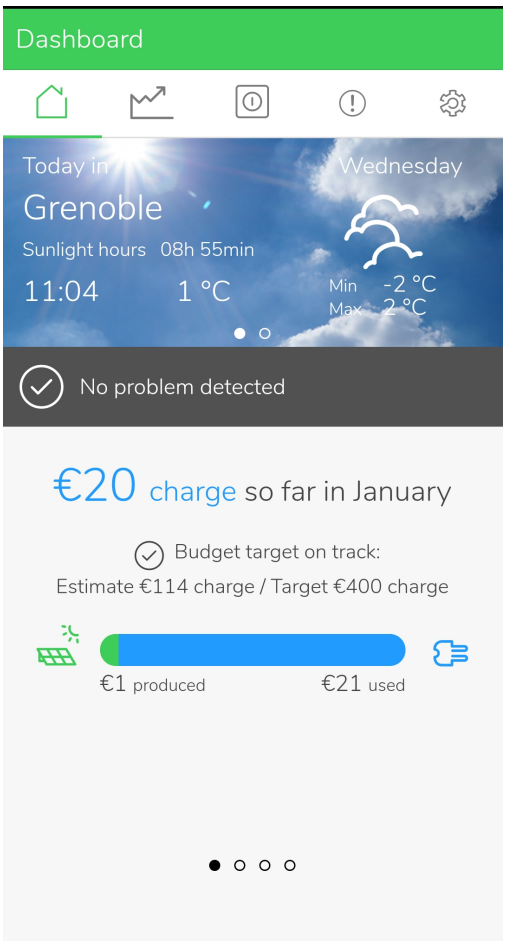

Figure 4: Wiser Energy Dashboard

Controls At the control page, in figure 5, there are simple (DP01) on/off buttons to enable or disable different home automations or optimizations. With a click on them one can get additional information (DP03) and a configuration form for these, as shown in figure 6. These configuration pages can easily be exited (DP06), but there is one problem: There is no information about what 'optimize energy' is(DP10), making this a well designed page with the negative point of missing information.

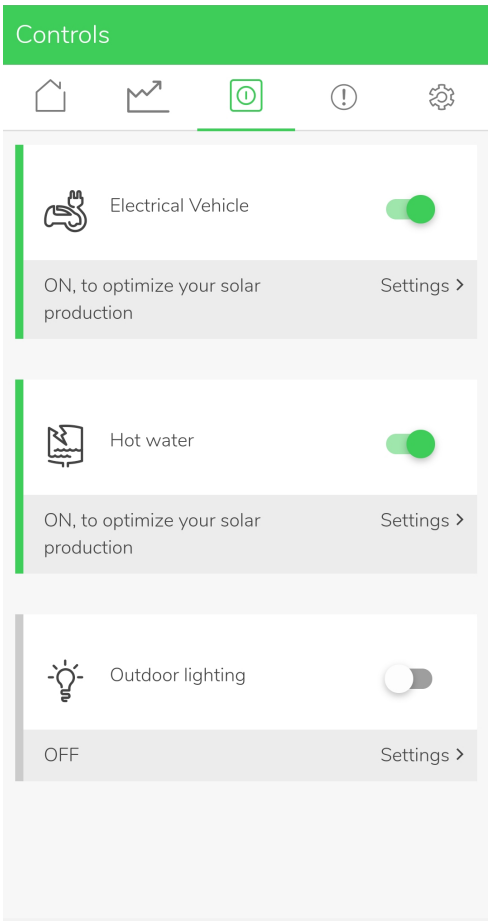

Figure 5: Wiser Energy Controls

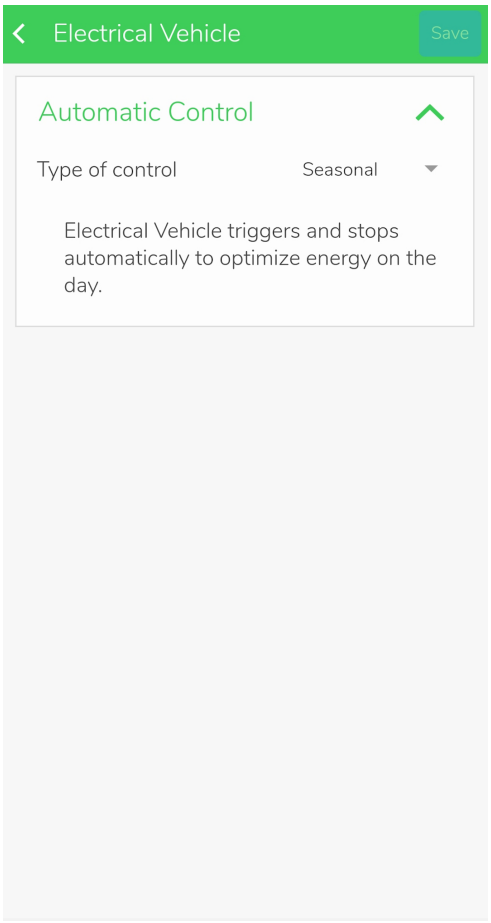

Figure 6: Wiser Energy Control Settings

Expenses The expenses page, shown in figure 7, is all about breaking down the buildings energy cost. Therefore, expenses and revenue are listed as simple value and in different charts. One can change between overall and more detailed views, also time intervals can be set. This offers the users perfect monitoring possibilities (DP01) in every situation. Beginners are able to analyze their consumption presented in basic and simple key figures. Experts might use the charts for a more detailed view.

Menu Figure 8 menu page is just a list of links to more detailed pages. The navigation is kept simple (DP01 and DP06) but one does not always know what to do on these details pages (DP10) making it hard to configure the system for new users.

In case of the scenario's use cases this mobile app provides good possibilities to monitor building data with simple user interface elements. Also there are multiple ways to configure the presented data, but no way to add new user configured user interface ele-

ments. The possibility to generate reports/documents is given in the menu. But there is no other way then comparing data values on different pages to compare non optimized and optimized building values.

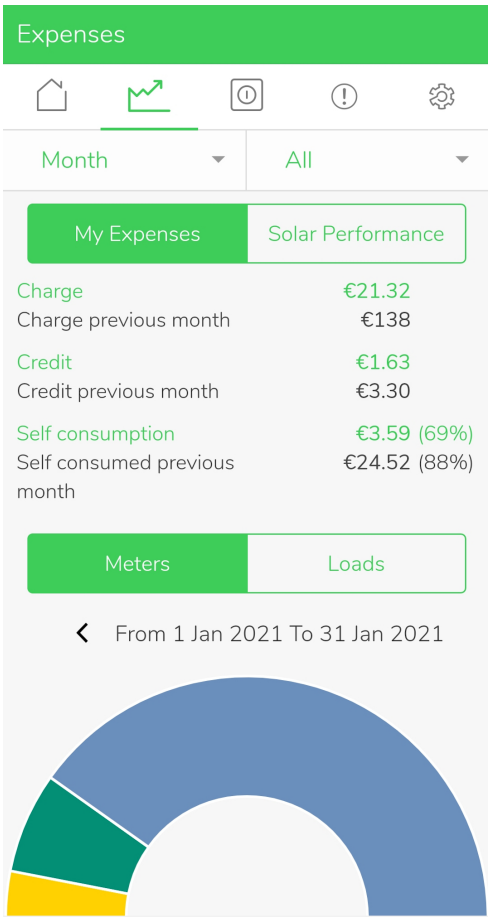

Figure 7: Wiser Energy Expenses

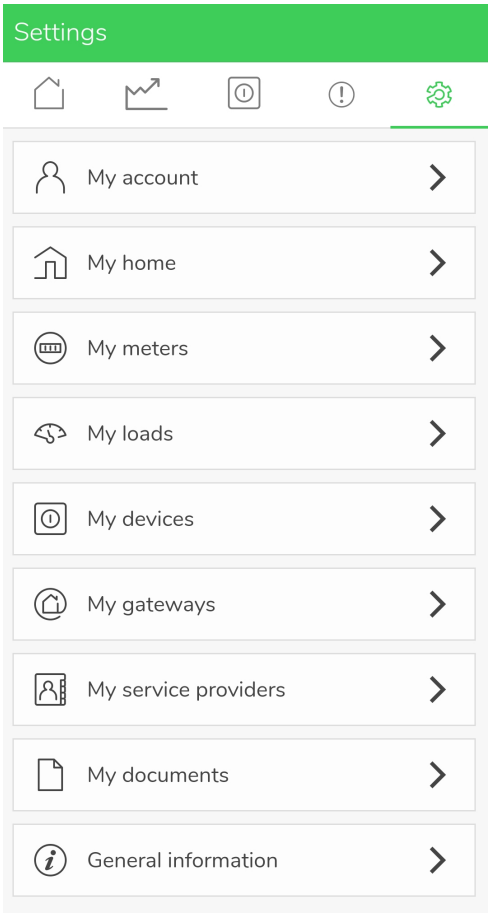

Figure 8: Wiser Energy Menu

HomeGenie

Next the open source software HomeGenie [20] will be evaluated. It is build as modular application where the user can plug in devices and sensors from a set of predefined manufactures. HomeGenie will then work as management system, controlling and automating plugged in modules.

Dashboard As homepage a modular dashboard, as seen in figure 9, is used. Users are able to configure it at their will by adding new modules and a drag and drop user interface element placement. Thereby the dashboard is as simple as the user configures it (DP01 and DP02). It is possible to use the module's controls shown in the user interface elements, but one is not able to receive more detailed information. Users are informed

about the system status(DP05) with status notifications but they often come very unclear (DP08) like 'Dashboard Porch Light Level 0.73'. For experienced users or the developers this might be intuitive that this message tells that the porch light's status is 'on' with an intensity of 0.73 of 1, but new or unexperienced users might not understand that. Besides printed values and controls, a graph is used on the dashboard page. This graph is kept simple (DP01), but also has not x axis labels. That is another point unexperienced users might have problems with.

Figure 9: HomeGenie Dashboard

Charts The next page to analyze is shown in figure 10 and is accessible on clicking the chart's button on the top right. It is the detail view of the consumption graph. This chart comes with axis labels, a legend, tooltips while hovering over the graphs, a data picker for choosing the data set interval, and the possibility to configure data set parameters. For monitoring energy data this chart is well suited and simple designed (DP01). The only negative aspect to point out is the different shapes of graph entries that might not be intuitive (DP02).

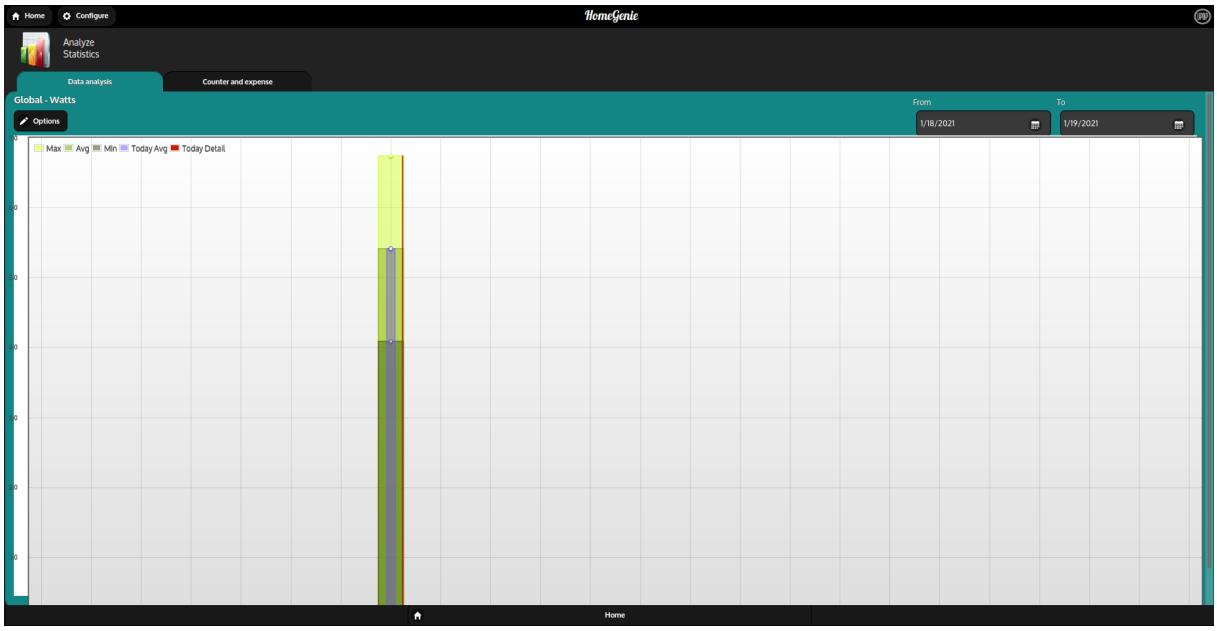

Figure 10: HomeGenie Chart

Schedules The last HomeGenie page to show is the feature to set schedules for home automation, shown in figure 11. It is possible to add new schedules including time period, a schedule behavior, and add device modules to the schedule. The dialogue is kept simple (DP01) but lacks of help and description texts (DP10). Once inside the dialogue there is no marked exit (DP06) back to the schedule overview. Also some coding skills may be required to customize schedule behavior, if desired.

Figure 11: HomeGenie Schedules

Overall this application is build simple but can also be used be experts with advanced knowledge. In terms of this thesis scenario use cases one can create and customize one (not multiple) dashboards. It is also possible to add new devices to the system, but only if the manufacturer is supported. Simple and intuitive user interface elements like charts and cards exist for precise energy data monitoring. The one requirement missing is that one can not compare the impact of optimization and automation algorithms.

4.2 Evaluation

In the following the state of the art user interface's analysis will be summarized. The learnings of this analysis shall be used in the development of an user interface for visualization of the potential of building energy management algorithms. An attempt is also made to formulate a learning from negative points, in order to avoid these errors in this thesis' development of an user interface.

Simple User Interface Once more it has to be pointed out, that a simple structured user interface is a key point. Following DP01 to DP04 of chapter 3 it is possible to create such a simple user interface, minimizing the users experience needed to understand it. As noticed in all of the approaches listed above, user interfaces might be structured simple but it is also the features and the small design decisions that help to minimize users cognitive load. This should be kept in mind while implementing this thesis' user interface elements.

User Interface Elements As seen in the openHAB or the HomeGenie application, it is helpful to categorize and group data and user interface elements. This helps the user to understand the interface faster and might assist in focusing on the important things. Therefore, it might be helpful to introduce something like user interface containers for categorizing and labeling user interface elements. Also it should be possible to have multiple pages and, thereby, less user interface elements on one page, to avoid dashboards like the HomeGenie Dashboard, that includes a lot of different data points and controls. Speaking of the interface elements itself, two kinds of data visualizations were used in all introduced approaches: Plain data values and charts. This should also be the way to go for this thesis development of an user interface. Plain values should be presented simple like in openHAB's dashboard and give users the option to receive more detailed data, for example when clicking on them. The charts on the other hand should be designed simple and lucid, but users should have the possibility to configure them at their will. In terms of chart types, area and bar charts where used in the evaluated approaches. This covers a wide range of presentation possibilities, therefore, these types will be used in the 5th section's implementation as well. Also charts should have a description, a legend, and tooltips as seen in figure 10. This supports new users and also lowers the cognitive load of every type of user.

Configurability In terms of configurability the introduced application showed many different approaches. Therefore, a distinction should be made between user interface element, page and system configurability.

User Interface Element Configurability: In terms of user interface element configurability the applications never made possible to configure the elements itself, only the element's content like data sets or intervals in graphs as in HomeGenie's charts (see figure 10). This was made possible with control forms at the top of the user interface elements or hidden behind buttons that pop up configuration dialogues. This will be adopted in this thesis' implementation. But also the configuration of the user elements itself should be supported.

Page Configurablitity: Only HomeGenie made it possible to configure the page layout by the users needs. This was achieved with a drag and drop feature in a configuration mode. This feature has its boundaries, like limited positions. To configure a whole page according to users imagination was only possible when the user had some basic coding skills. This is appropriate when experienced users interact with the system. But for everyday use in different home settings there should be found a other solution.

System Configurability: Each system was configurable on its own way. Some had only

preset actions available, some were build to enable all kinds of energy and house management data. Often coding skills were required. In this thesis' implementation this should be avoided. It should be possible to add new data points or devices easily and intuitive.

Help and Documentation All the introduced applications were lacking in terms of help texts or documentation. Not in regard of code documentation but in regard of explanations for new and inexperienced users. Therefore, the best features of every evaluated application should adopted in this thesis' application. System status messages, graph legends, describing tooltips, and explaining texts should be a part of the application.
It seems like this may be the hardest part of implementing a good user interface. Therefore, a well designed plan should be made before implementing a new complex user interface.

Algorithm and Parameter Comparison In none of the evaluated applications a tool for optimization comparison was included. This needs to be implemented from scratch in this thesis. As seen in some of the listed approaches, it is hard to compare data if you have to switch pages or configure features while comparing. Therefore, in the development of a user interface for visualization of the potential of building energy management algorithms this error should be avoided. That is why a single comparison page should be introduced.

5 Development of the User Interface

After the basic context in chapter 2 and use case scenarios in chapter 2.2 have been set and related work has been evaluated in chapter 4, this thesis' user interface for visualization of the potential of building energy management algorithms will be developed in this chapter. With the help of previous chapters' work a modern, responsive and user friendly web application will be designed and implemented. The design process will start with a summary of all requirements and a description of all used technologies. Already available software parts will be introduced as well. Then the application's structure and architecture will be developed, a way to build the application as generic as possible will be found, and the web application will be implemented. Afterwards, this implementation will be evaluated in chapter 6.

5.1 Requirements

First of all, it is important to fix the requirements and constraints of the system. These will be derived form the use cases of chapter 2.2. In the figures 12, 13, and 14, use case diagrams are shown that were developed from these use cases. Requirements and constraints will be set with their help. In the following, these system requirements and constraints will be listed.

Requirements

R01: Configurablility To configure the dashboard according to the users' wishes, the web pages has to be modular configurable. The best way to achieve this is to implement an admin page. This page has to provide the following features:

- **R0101: Create Pages** Dashboard pages can be created at the admin panel.
- **R0102: Configure Pages** The created pages may be configured by including and editing user interface elements.
- **R0103: Create UI Elements** The user interface elements that will be included in the user configured pages may be created at the admin page as well.
- **R0104: Configure UI Elements** It is possible to configure already created user interface elements.
- **R0105: Create Datapoints** Since the user interface elements are there to visualize data, data points may be registered and edited at the admin panel.
- **R0106: Create Metrics** Simple values may not be enough for monitoring complex systems such as buildings. Therefore, metrics that represent formulas over data values may be created.

R02: User Interface Elements In chapter 4 plain value cards, area charts and bar charts were chosen as the user elements. Grouping user interface elements was positively pointed out, as well.

R0201: Cards Cards will represent basic plain values. They consist of one value, the value's unit, an icon and a color scheme for visual support, and some decorative borders. A card may also have a tooltip providing more information or may function as a button with some additional logic.

R0202: Charts All charts consist of an descriptive headline, one or many data sets, tooltips while hovering above chart values, and an x and an y axis labeled with its values. Also charts might be configured via dropdown options at the chart's top.

R0203: Containers It is possible to create user interface element containers. These group all kinds of user interface elements both at the admin dialogues and at the web page.

R03: Comparison Tool Since it is the thesis' goal to visualize the potential of building energy management algorithms, there has to be a tool to compare different algorithms or algorithm configurations.

R0301: Plain Value And Chart Comparison Since all of the data will be presented in cards or charts, it is reasonable to use these elements again for comparison. Listing these UI elements in two columns one for each algorithm will create a simple comparison tool.

R0302: Comparison Graph A comparison graph will be introduced as a special user interface element. Its purpose is to point out differences in the building's key figures between two algorithms or configurations. Therefore, bar or area charts will be reconfigured showing the differences in chosen key figures.

R0303: Comparison Selection Users are able to select different algorithms or algorithm configurations to compare. This will be done with the help of a selection form.

R0304: Comparison Configuration Just like normal dashboard pages the comparison page should also be configurable via the admin page.

R0305: Comparison Algorithm Registration New algorithms to compare might be registered at the admin panel. Algorithms hold an identifier that maps them on an algorithm at the backend.

R04: Generics The application has to be built as generic and modular as possible. The user is able to extend the functionality or the frontend by only using the application's features.

R05: Screen size optimization This thesis' web page will be designed responsively. Since the web application will mostly be used on PCs or tablets, the page will be optimized for these devices.

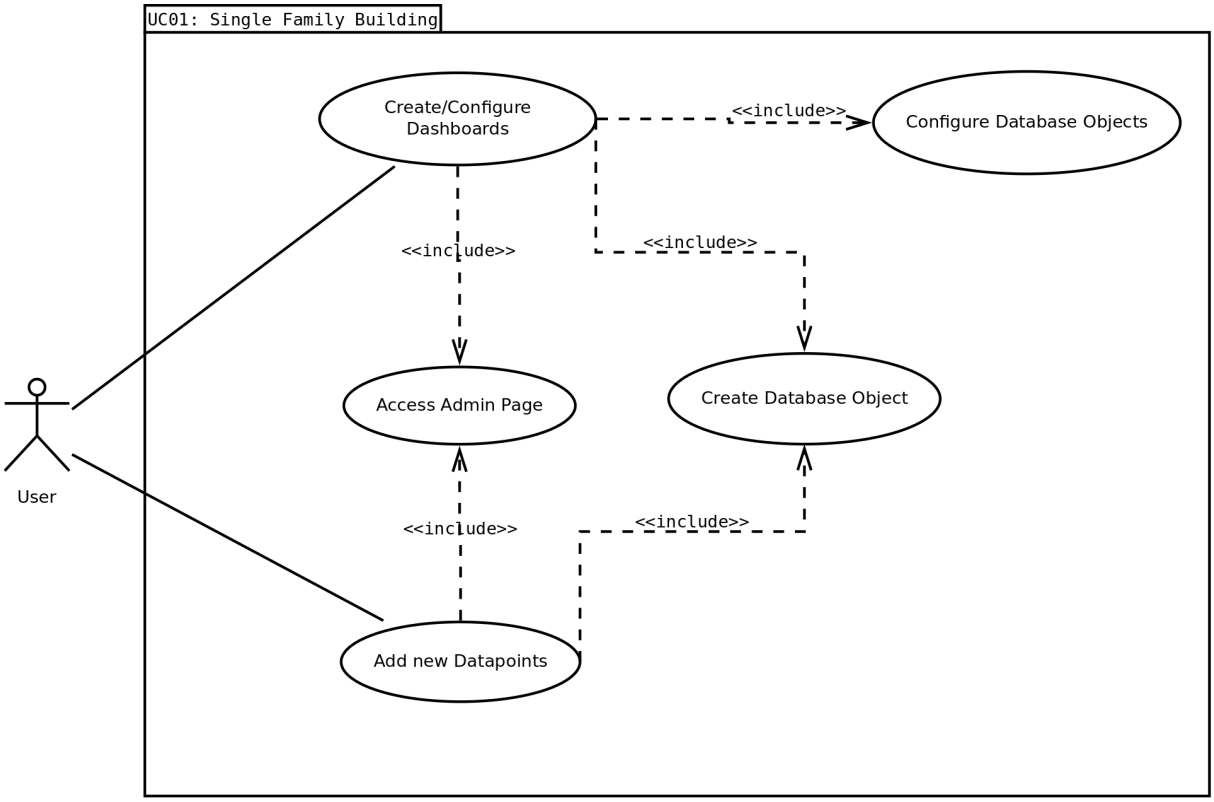

Figure 12: Use Case 1: Use Case Diagram

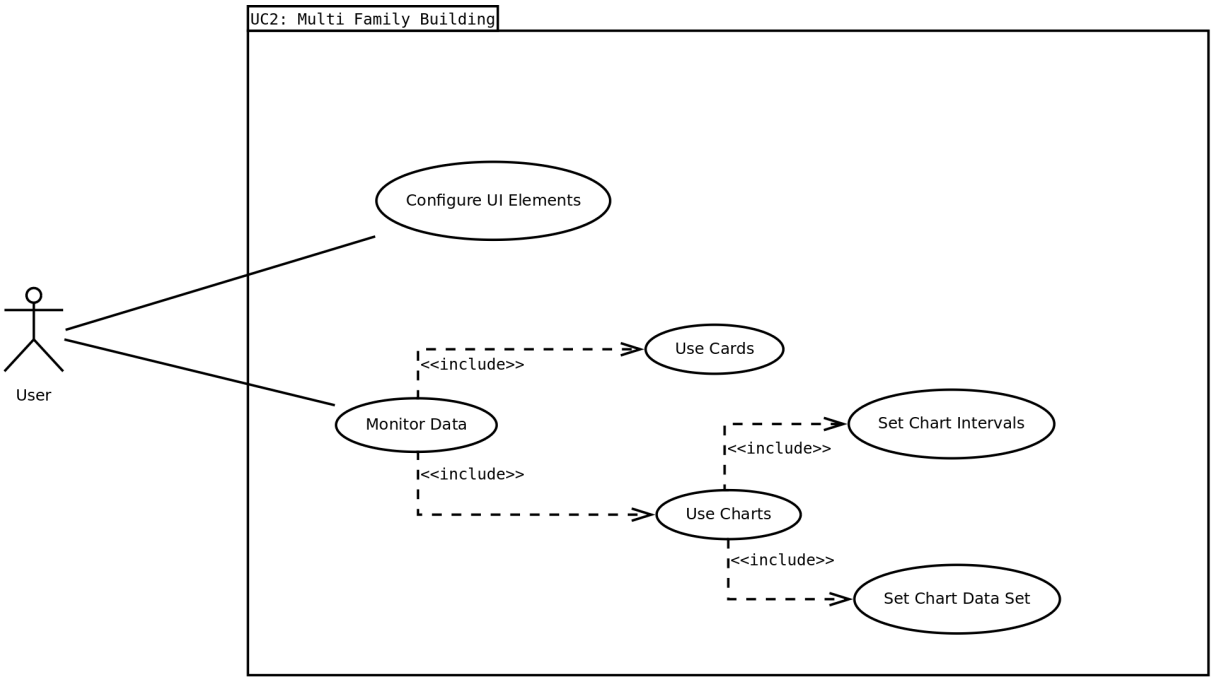

Figure 13: Use Case 2: Use Case Diagram

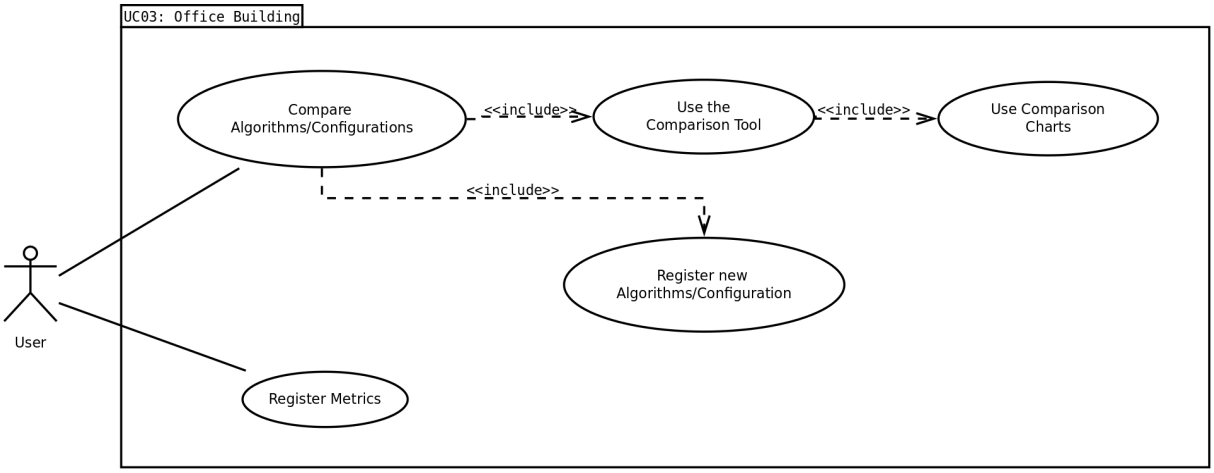

Figure 14: Use Case 3: Use Case Diagram

Out of these requirements the application's features will be designed and implemented in the following steps. To limit the implementation effort in terms of non-required features, a list of constraints will now be introduced.

Constraints

C01: Application as Module This application will not be a standalone web page. The application will be part of FZI's EMP and reachable through EMP's navigation.

C02: No Energy Management Settings Unlike some of the evaluated applications in chapter 4, there will be no energy management system settings, like configuring the algorithms, controlling devices, or scheduling devices. Other parts of FZI's EMP will take care of this.

C03: No User Written Code It should not be possible to configure the system with user written code. The system will be completely configurable via the admin panel.

5.2 Generics

The requirements state that the application pages should be generic and should also be created and configured by users with the help of an admin panel. To achieve this goal an approach inspired by content management systems was chosen. Since the Django [18] framework is used, an admin page is available. With the help of this feature database objects can be created and configured using HTML forms, as seen in figure 15. Therefore, it is reasonable to use this admin page to configure the thesis' frontend pages. On a page request the configured page data will be loaded. This data will be processed and an HTML page will be built.
This workflow allows the user to create and configure a page only using admin dialogues. No coding skills are required by the user. Thereby, this approach is not only generic but also user friendly.

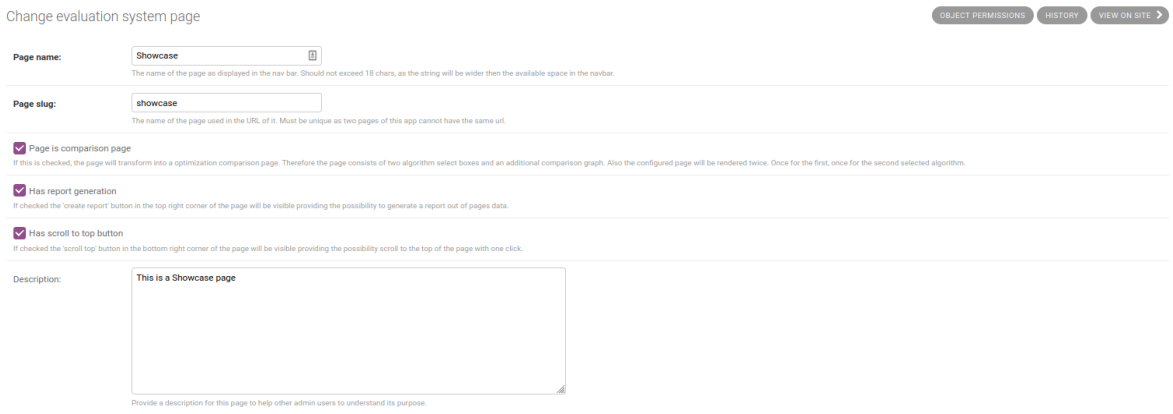

Figure 15: Example: Admin Page Forms - Text and Boolean Fields

To follow the generic approach, it has to be possible to include an arbitrary amount of user interface elements to created pages. These user interface elements need to be configurable as well. Therefore, a admin dialogue has to be defined for each possible user interface element type. The objects created of with these dialogues will be stored in the

database and a user configured page will consist of one to many such objects.
These objects usually have to be configured each at their own Django admin page. This is not very user friendly, since users therefore need to switch admin dialogues for every single user interface element they want to create or edit. For example, a user creates a dashboard page and wants to add a user interface element. Therefore, he saves the newly created page object, leaves the dialogue and creates the required user interface element in another admin dialogue. Afterwards, he has to add the newly create element to his page and, therefore, opens the first dialogue to add the element. This workflow is very inconvenient and comes with a high user memory load. Therefore, a solution has to be found for this challenge.
To avoid such a configuration workflow the referenced user interface elements and their HTML configuration forms will be included as nested admin forms into the containing page's admin form. This results in one single page configuration dialogue. New user interface elements may be create, configured, included, and deleted at the same admin page. A page element object serves as head and entry point of this recursive page structure. This tree structure can be traversed and, thereof, a web page can be build. An example page structure is shown in figure 16. This structure would have been configured in one page element creation dialogue. The container and the elements would have been added and configured in this dialogue as well. Thereof, on request, a page would be build that contains a user interface element container consisting of three user interface elements. Below this container, one single user interface element would be rendered.

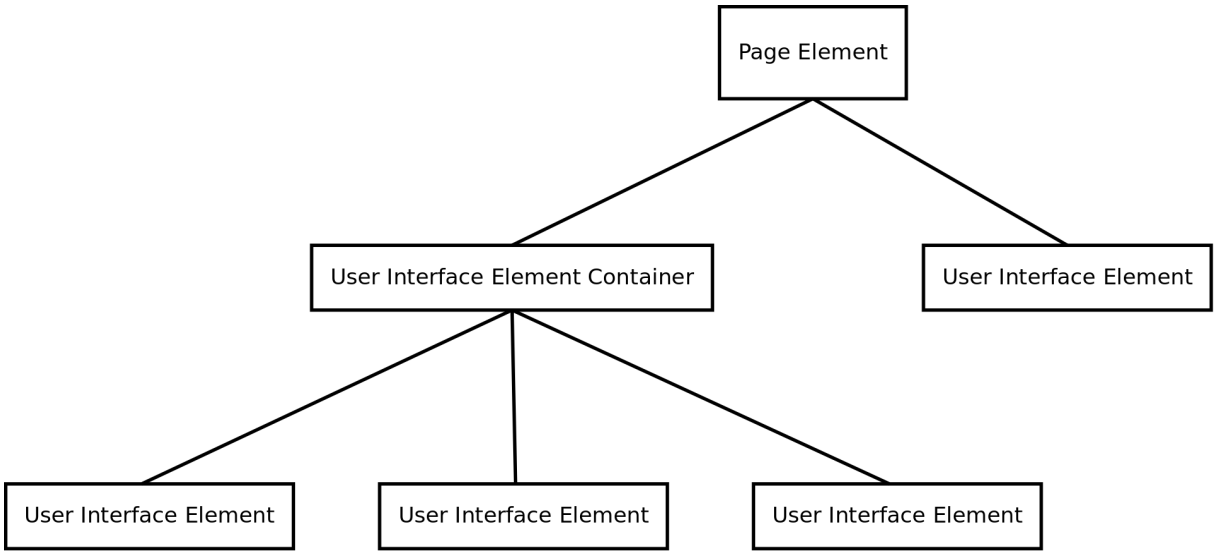

Figure 16: Example: Recursive Page Structure

5.3 Architecture

In this section the systems architecture will be designed and described. First the application's presets will be listed. Since the thesis' application will be embedded in FZI's energy management panel, these architectural and logical presets are given.

Presets

The FZI's energy management is written in Python and with the help of the Django framework [18]. Therefore, it is mandatory to use this framework as well. Due to the use of this framework the application follows the model view presenter design pattern. The details of this pattern will be described later on.
Also Bootstrap [51] is used as Frontend CSS framework. Therefore, a basic web page structure was implemented in FZI's EMP main module, that will be adopted and extended. The base page, as seen in figure 17 consists of a top bar and a navigation bar. The top bar includes a logo, a title, and the actual time. The navigation bar contains a user salutation, a button linked to the login page, and menu items for each in the main module registered module . In figure 17 one module is registered and shown in the navigation bar, here marked yellow. This menu item holds all submenu items for navigating the module's pages. This basic page structure will be inherited by every of the application's frontend pages. The module page's components will be loaded in the white center. This results in high usability since users are able to use this navigation bar at every time at every page.
With the help of Django's build-in django.contrib.auth package a user management was build. This allows system administrators to create different users and user groups and, therefore, manage the application's pages' access. User rights may be set for frontend pages and admin dialogues.
Besides these presets, the main module of the energy management panel introduces a database storing all important application data, especially the objects created of Django models. This database will be used in this thesis' application's context as well.

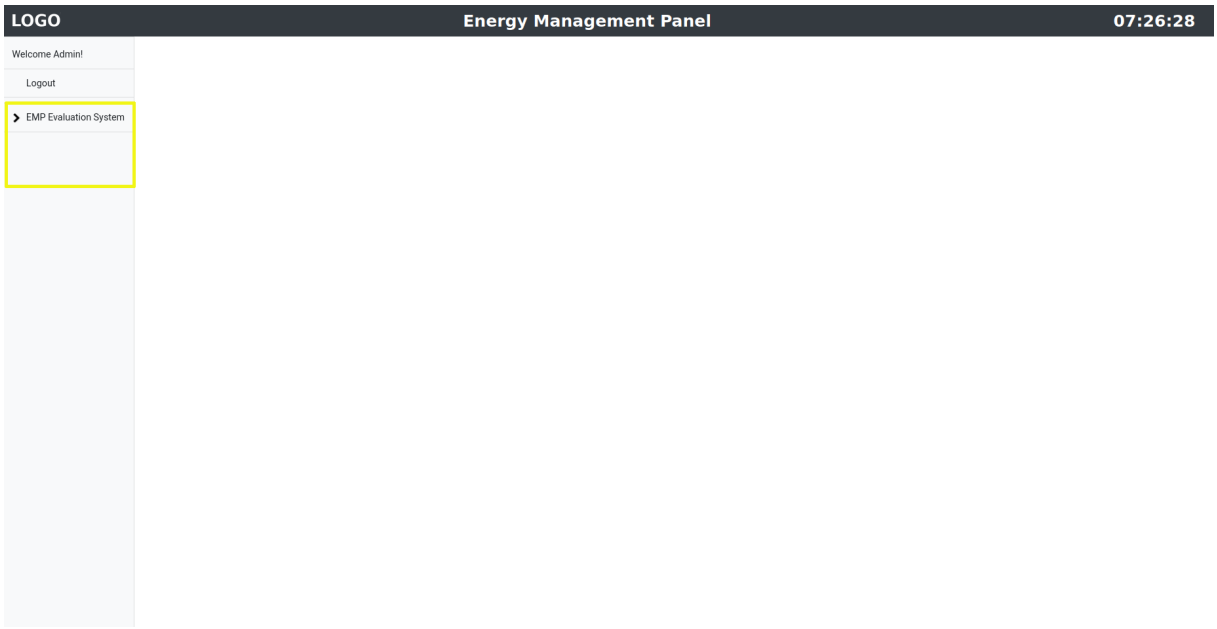

Figure 17: Empty EMP Web Page

Besides these architectural and structural presets there are two application parts that will be used to implement the user interface for visualization of the potential of building energy management algorithms. These parts will be introduced in the following.

Datapoint API Since a building or an apartment may be equipped with any number of diverse devices, these devices and especially their measured values need to be registered, managed, stored, and prepared for further processing. Therefore, the Building Energy Management Communication Framework (BEMCom) has been developed. This framework is designed to minimize the necessary effort to establish communication with diverse entities. These device entities are represented by a Datapoint objects as seen in figure 18. A Datapoint is defined as a Django model that holds all important information to describe a device entity. The name, the id, and the descriptions are provided for further use of the object. The Datapoint type indicates which kind of device is represented. Also there are fields that describe possible values, important last values and the values unit. This Datapoint model is used to communicate device information in FZI's energy management panel. Besides the Datapoint model there are three other models that are directly related to the Datapoints and each other. DatapointValues describe measured values at one given timestamp. With the help of this model's objects a history of Datapoint values is created. DatapointSetpoints represent a user configuration of one Datapoint. For example, the setting that the room temperature may not be lower then 19°C at nine o'clock. The third model, the DatapointSchedule, represents a schedule for a given Datapoint. For example, from 9 a.m to 11 a.m the dishwasher will be active and running. These schedules are

create by the building energy management system.
In this application's context all of these models are used, especially the Datapoint and the DatapointValue model. To access these model's objects a API was build providing endpoints for receiving, updating, and deleting objects that are stored in the database. It is possible to receive all Datapoint objects, all Datapoint value, schedule, or setpoint objects of a given Datapoint, or a value, a schedule, or a setpoint of a given object to a given timestamp. Therefore, for all conceivable uses a endpoint is provided.

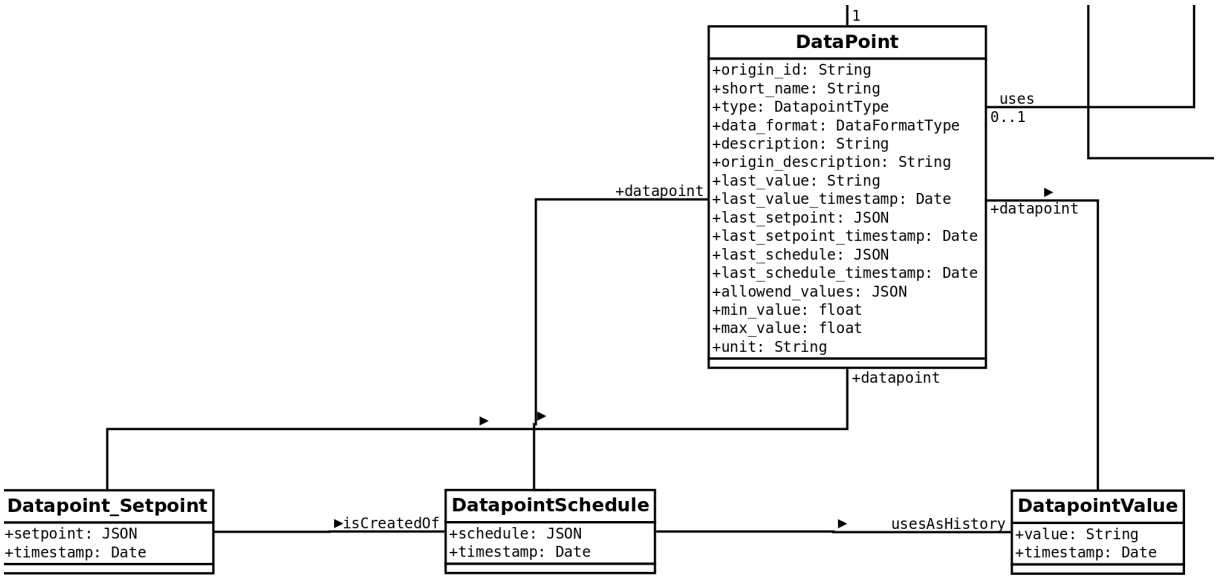

Figure 18: UML Class Diagram: Data Point Section

Simulation API Implementing an application for visualization of the potential of building energy management algorithms, there has to be a feature to receive large amounts of data produced by building. Since a user should not have to wait 30 days to review the buildings data in this time interval and wait another 30 days to run an other algorithm to compare, a simulation application was build. This module of FZI's EMP simulates building energy management decisions and, thereof, the resulting data. A API was build to communicate with this simulation module. The API consists of three important endpoints. One endpoint to start a simulation, given an algorithm identifier, a start and a end date. Since such a simulation may take a long time, the second endpoint returns a simulation status and a estimated waiting time. If the status tells that the requested simulation is finished, the third endpoint may be used. On request it returns the simulated data consisting of values, setpoints and schedules for all registered datapoints.
This API will be used to simulate building energy management algorithms and receive their simulated data. The data will be presented to the user to compare these algorithms and visualize their potential.

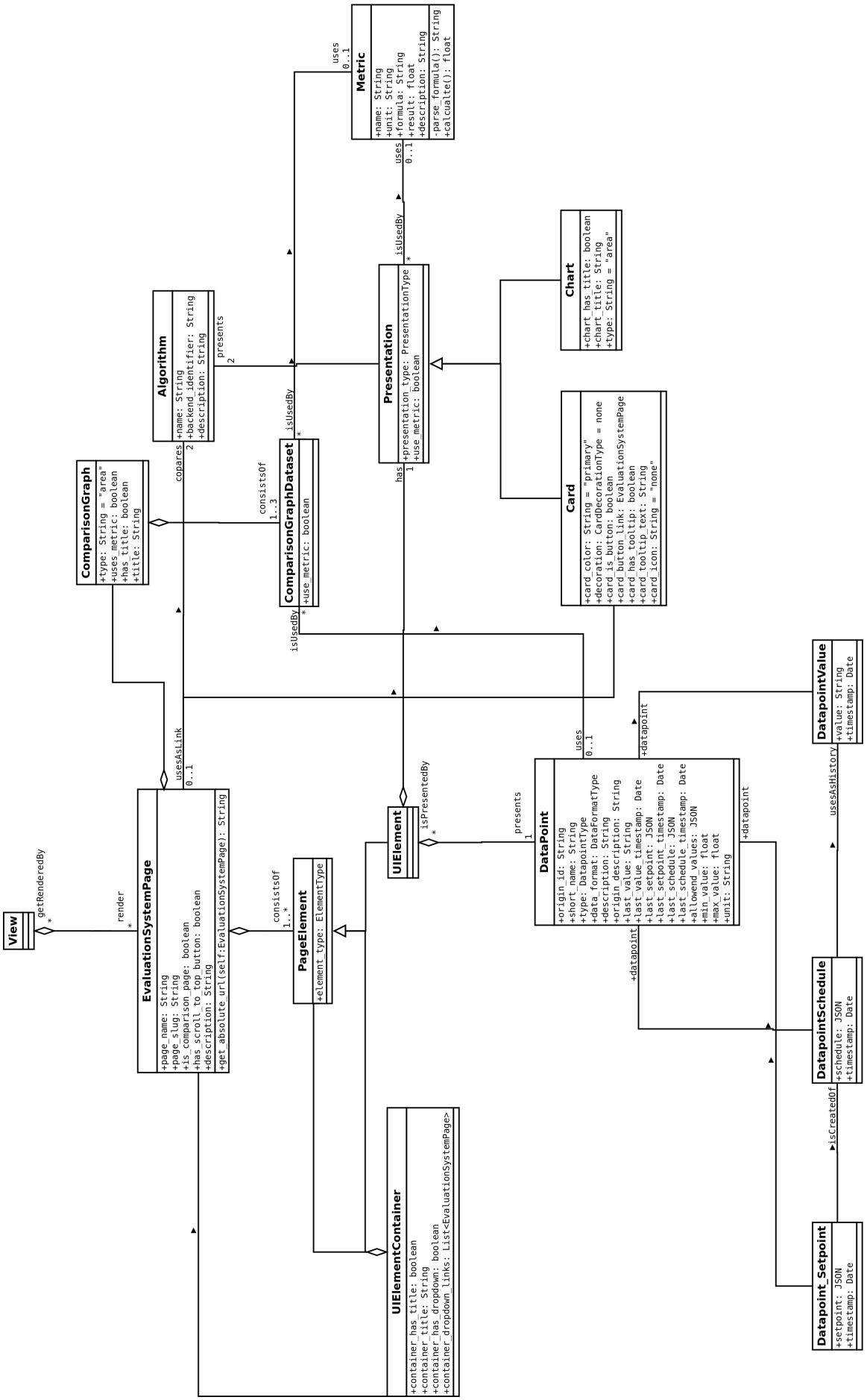

Figure 19: UML Class Diagram of Model Architecture

System Architecture

Due to the use of the Django framework the application follows the model view presenter design pattern. The object logic of this thesis application is defined in Djagno models. These are defined in python code and will be created and configured with the help of the admin panel. Crated model objects are saved in the application's database, prepared for presentation in the presenter, and presented as HTML templates.
To design and document the model architecture a UML class diagram as seen in figure 19 was created . The shown class structure follows the generics as described previously. In the following we will take a closer look at the most important design decisions and how they enable a generic web page to be build. Also in each of these steps the most important fields of every model will be introduced.

Evaluation System Pages Figure 20 shows the top of the recursive page architecture. To make generics possible the EvaluationSystemPage model was designed. Every user configured page is based on this model. A EvaluationSystemPage consists of a name, a slug, and a description. These are used for navigation, URL creation and user information. Additionally there are two boolean fields. One that implicates if the EvaluationSystem-Page is used as comparison page or as dashboard page. The second field implicates if a scroll top button is included if the pages is getting longer then the users screen height.

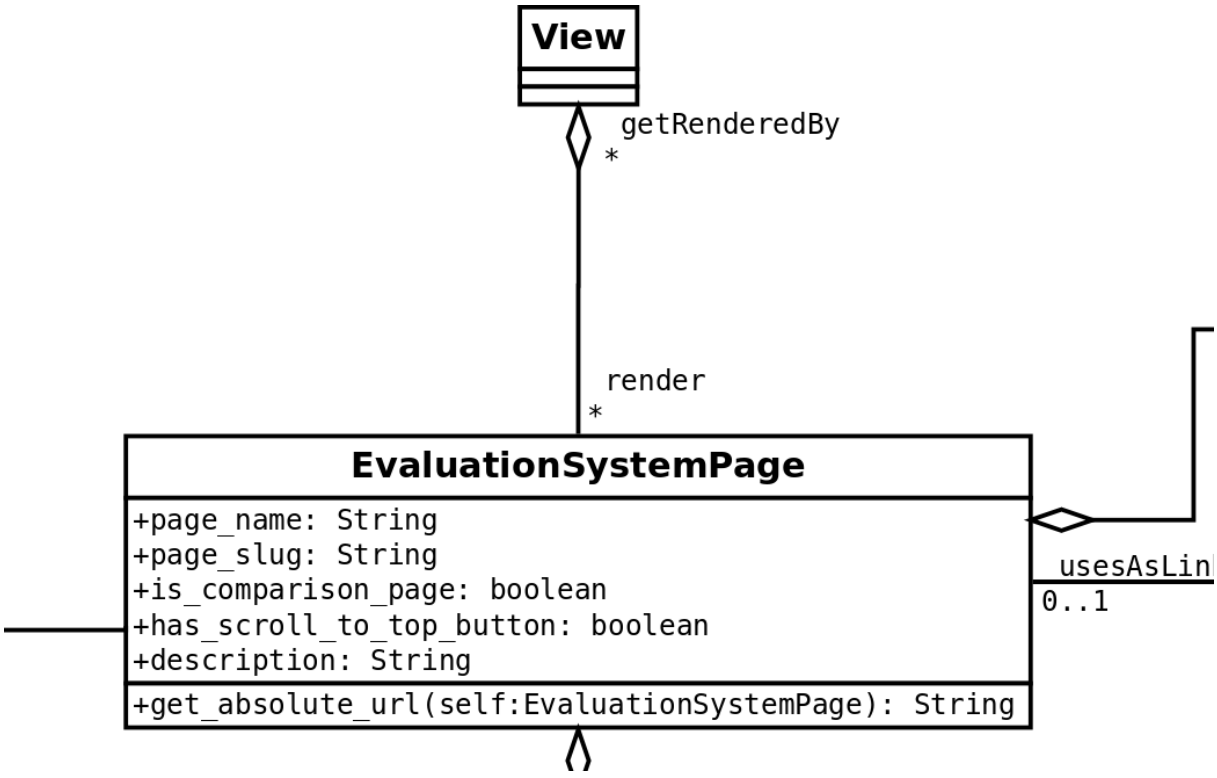

Figure 20: UML Class Diagram: Evaluation System Page

EvaluationSystemPages are assigned to a hard coded Django HTML template. This template dynamically creates a web page out of the EvaluationSystemPage's objects and their linked PageElements.

Page Elements EvaluationSystemPages consist of one to many custom user interface elements. Figure 21 shows the core models for this generic page structure. The composite pattern was used to create the recursive page setup possibilities. One EvaluationSystemPage object consists of one to many PageElement objects. A PageElements element is a abstract class and is either a UIElementContainer or a UIelement. UIElementContainers may consist of one to many PageElements and are used to group PageElements, most likely UIElements. UIElements on the other hand serve as recursion end points as already shown in figure 16.

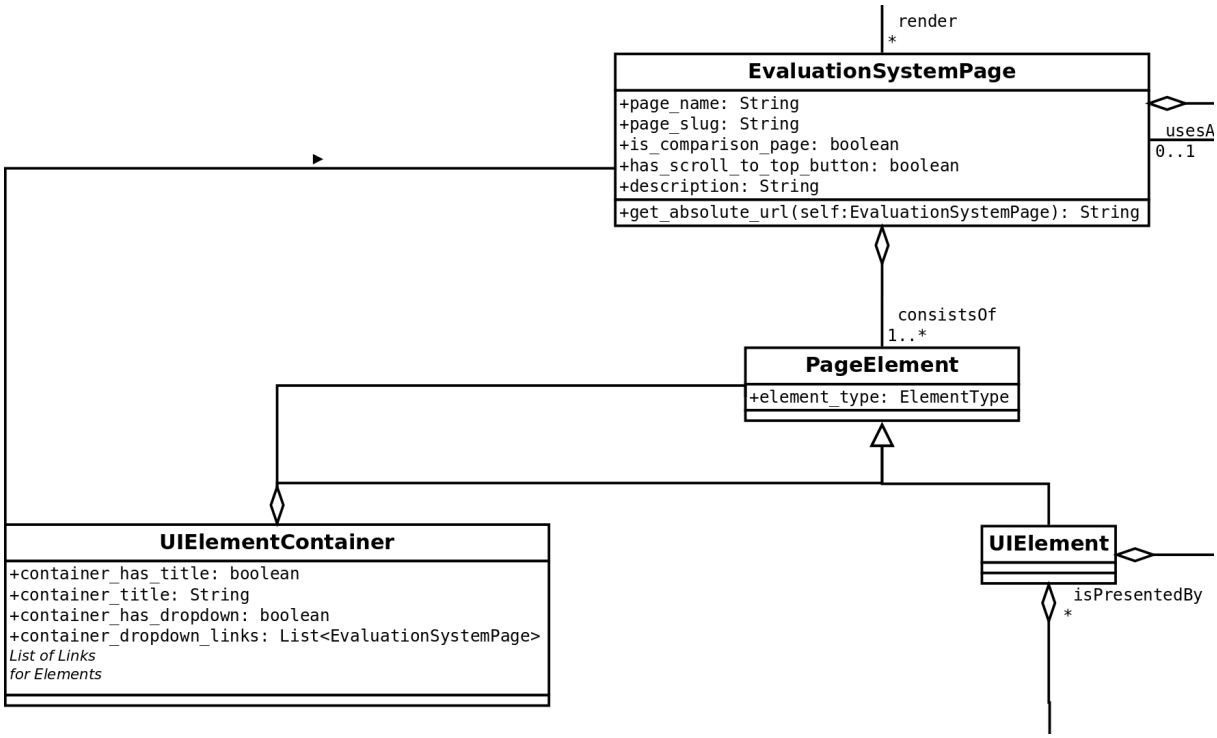

Figure 21: UML Class Diagram: Page Element

UI Elements The UIElement model object holds no further information. It's only purpose is to function as map, linking a Datapoint to a Presentation as seen in figure 22. The actual user interface customization happens in the Presentation model.

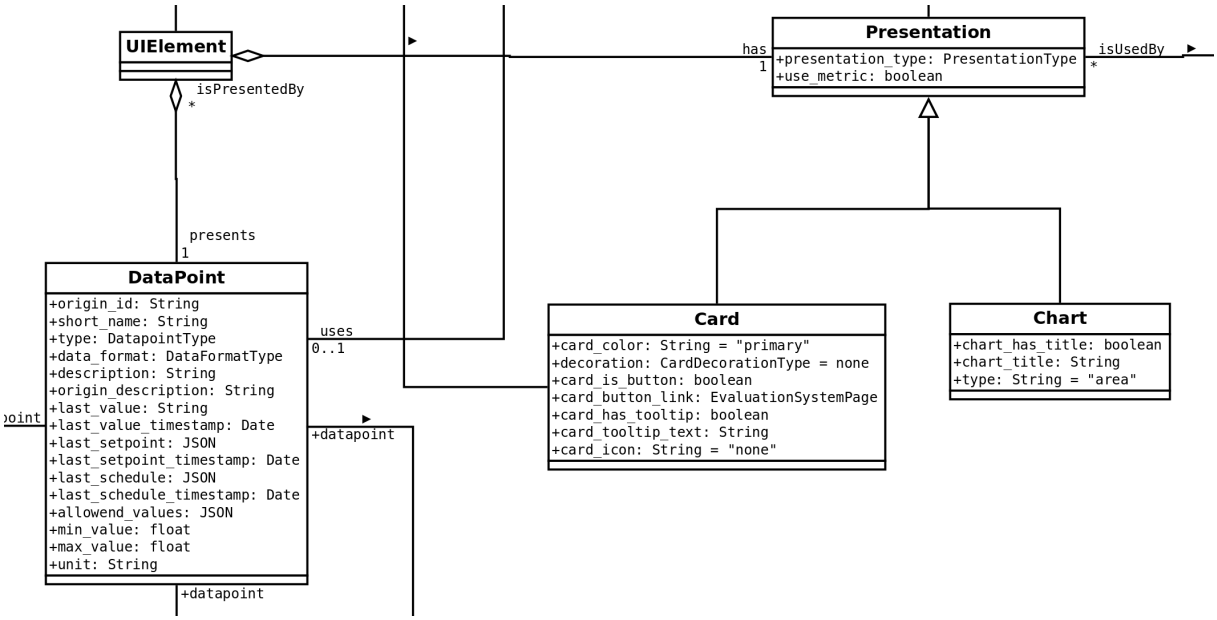

Figure 22: UML Class Diagram: UI Elements

Presentation Figure 23 shows the Presentation model. This model is defined as an abstract class having two possible types. A Presentation object may either be a Card or a Chart. These are the two user interface elements set in the requirements. Their models hold all important information for user configuration. Besides these types a presentation may be linked to a Metric object. Metric objects represent a formula calculating values over Datapoints.

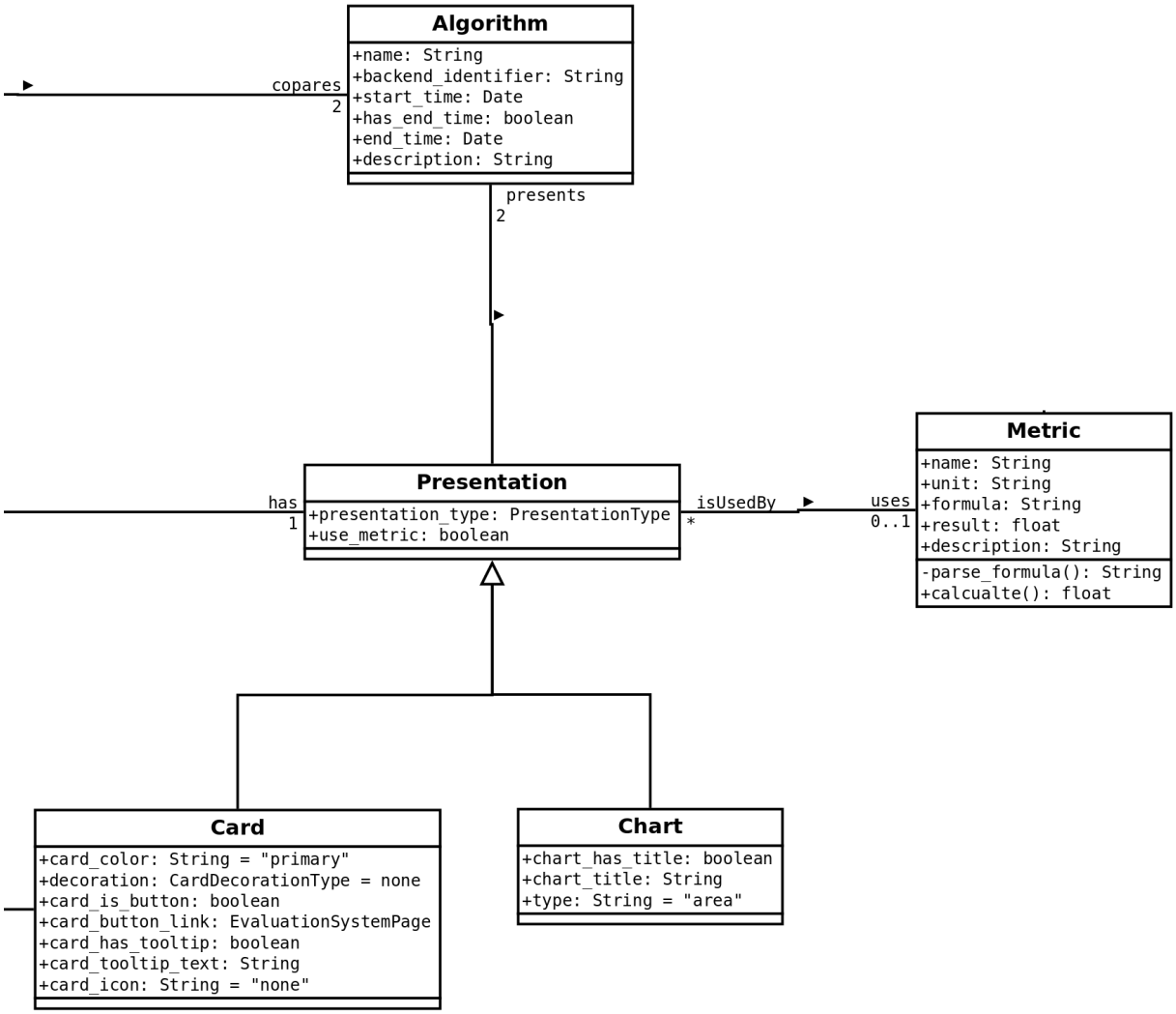

Figure 23: UML Class Diagram: Presentation

Comparison Pages The requirements state that apart from building dashboard pages there has to be the possibility to create algorithm comparison pages. Therefore, the EvaluationSystemPage model has a boolean field that, if true, converts a dashboard page into a comparison page. For this purpose ComparisonGraphs may be linked to a EvaluationSystem page as shown in figure 24. Each ComparisonGraph consists of one to three ComparisonGraphDataset that link either a Datapoint or a Metric to the ComparisonGraph. In this way the data presented by ComparisonGraphs may be chosen.

Since the application has to communicate with the simulation API to gather data for comparison, the system has to know what simulation algorithms are available. Therefore, algorithms may be registered with the help of the Algorithm model. Algorithm objects hold a name and a description, but most important a backend identifier. This identifier

is used to call the simulation API and has to be the same as the simulation algorithm identifiers at the simulation module.

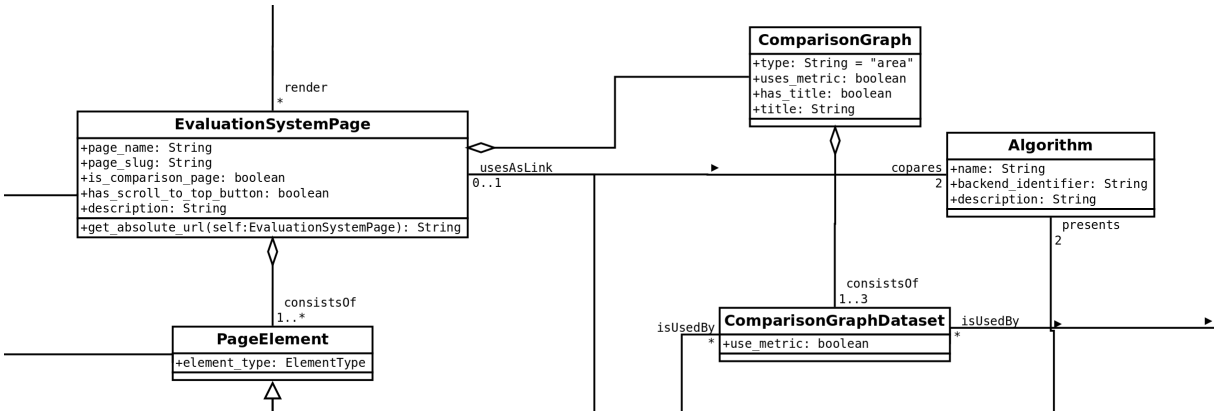

Figure 24: UML Class Diagram: Comparison Page

5.4 Implementation

In the following section a closer look is taken on how this system architecture, the generics and the requirements from the previous sections are set in code.

Backend

Since the Django framework takes over most of the backend tasks like managing database communication, the interaction between the database and the application, chaching, user authentication and admin page core functions like data validation, there is only little backend work left to do. The system model structure was translated one to one into python code. Thereof, Django builds the database schema. Relations between database objects, like a EvaluationSystemPage consisting of PageElements, are implemented as foreign keys.
Special setting variables and non-native Django modules were registered in the module's setting file.

Admin Page

Before introducing the web app's page implementation, the admin pages' implementation needs to be introduced for a deeper understanding of how database objects will be created by the users. Using Django's basic admin page the users lands at a admin overview page, as seen in figure 25, once they are logged in. This and deeper level admin pages

are configured with the help of Django's admin settings and some additional JavaScript methods for advanced configuration.

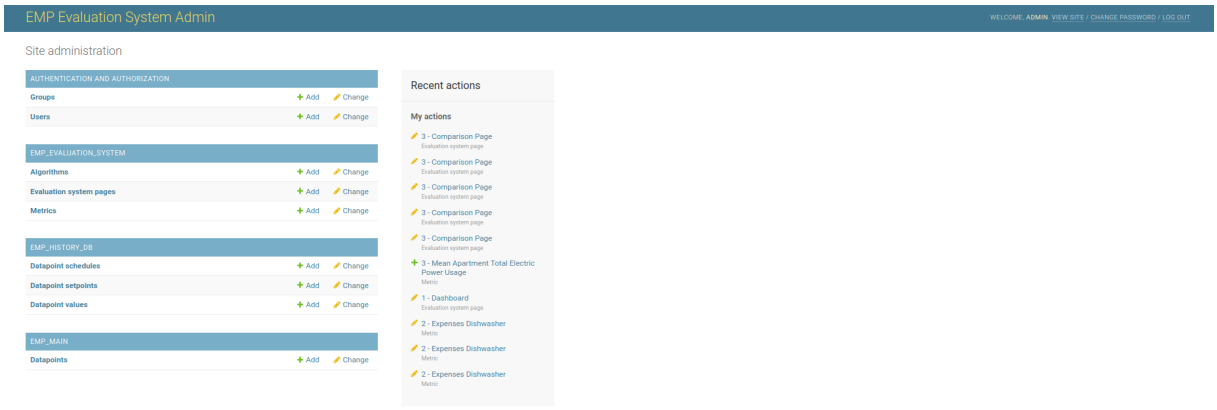

Figure 25: Admin Starting Page

Admin Configuration All basic admin configuration takes place in the admin.py file. The first step of configuring the admin page is to register all models that shall have an own admin creation and configuration form. In this application only EvaluationSystemPages, Metrics, and Algorithms have their own admin pages. All other defined models can not exist on their own and, therefore, do not need their own configuration forms.
Since these non registered model objects still need to be configured, they are included as nested admin forms in an EvaluationSystemPage form. This is accomplished with the help of the nested_admin Django module. In figure 26 one can see the resulting recursive form structure of an admin dialogue. Users are able to create a EvaluationSystemPages, add an UIElementContainer or UIElement to the page, then add a Presentation and, therefore, a Presentation type to it. In each step it is possible to configure these elements. If an other element is needed another recursive branch can be added and configured.

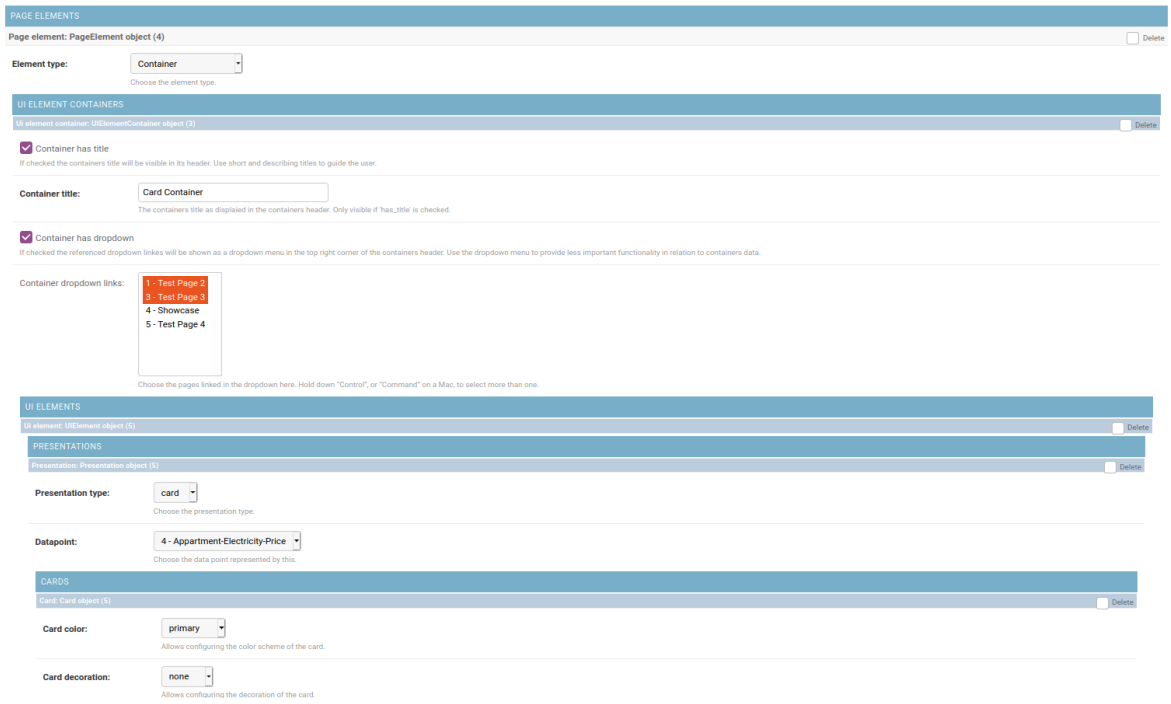

Figure 26: Admin Nested Forms

Admin JavaScript As Django's admin forms do not come with unlimited configurability, some form behavior needs to be handwritten. Therefore, a custom JavaScript file is registered. These scripts handle custom form behavior like hide and show HTML forms when they are not needed. This clears the screen from not needed forms and, therefore, increases the usability by hiding unimportant information.

Webpage

After creating web page objects with the help of the admin panel, these pages are now available to users at the application's web frontend. In the following it will be explained how these pages are build, filled with data and how the user interface elements are designed and implemented.

Building Pages On page request the view preparer prepares and loads the requested EvaluationSystemPage object from the database and sets this page object into the request's context. Therefore, the object's fields are available to use in Django's HTML templates. As recursion starting point serves one HTML template for the EvaluationSystemPage. This template loads another template for every user interface element linked to this page and so on. Nearly every defined models has it's own HTML template adding it's custom information to the web page. This modular structure enables generic pages to

be build. The HTML templates of cards and charts are the leafs and thus the endpoints of the recursion.

Receiving Data Once the page structure is set in HTML, the cards and charts need to be filled with the data that was linked in their database objects. Therefore, the Datapoint API endpoints, or in case of a comparison page the simulation API endpoints, are used to request the linked data. This data will be further processed at the client side with the help of JavaScript.

Process Data The via API calls collected data needs to be processed before being presented at the web page. Hereby the data is divided in four categories: real time building data, metrics on real time building data, simulated data, and metrics on simulated data. In terms of real time data the linked Datapoint object is requested and the last data value will be used to present as value. If the data is simulated the whole building's data is return by the simulation API. This data set is searched for the Datapoints last value, so no additional API call is needed.
When it comes to metrics this process becomes more complex. Every Metric provides a formula as String. This in the actual implementation might contain basic mathematical operators for addition, subtraction, multiplication and division. Additionally opening and closing parentheses are allowed. Besides that a 'sum' and a 'mean' function over arrays are provided. As last feature the user may link datapoints using 'dp_{Datapoint Id}' inside the formula. These formula strings are parsed for these Datapoint links. Found links will be replaced with their Datapoints last value after requesting the value from the API or searching it inside a simulated data set. The resulting string then is evaluated using JavaScript's build in 'eval' function. The computed result is then used as the Metric's value.
Since charts present Datapoint values over a given time interval this process is extended for their use. The Datapoint or Metric values are computed as before. Each value is collected for each timestamp of the chart's time interval. Afterwards, the collected dataset can be used to print a chart.
After processing the data is ready to be presented by the application's user interface elements.

User Interface Elements In the following all user interface elements will be introduced and their implementation described. For user interface design Bootstrap CSS classes were used. Also elements of the sb-admin-2 Bootstrap template [37] were adopted, especially for designing the charts.
In figure 27 the frontend presentations of UIElementContainers and Cards are shown.

The UIElementContainers are implemented with the help of bootstrap cards, consisting of a card header and a card body. The card header consists of a container title and a dropdown linking further pages. The body holds the contained user elements. The Card model's objects are also created with Bootstrap cards. The cards values are requested, and in terms of Metrics calculated, on page load and set as text. The card's color, icon, and borders are taken over from the model's customization, as well as possible tooltips or button links. As card headline the linked Datapoint's short name is used.

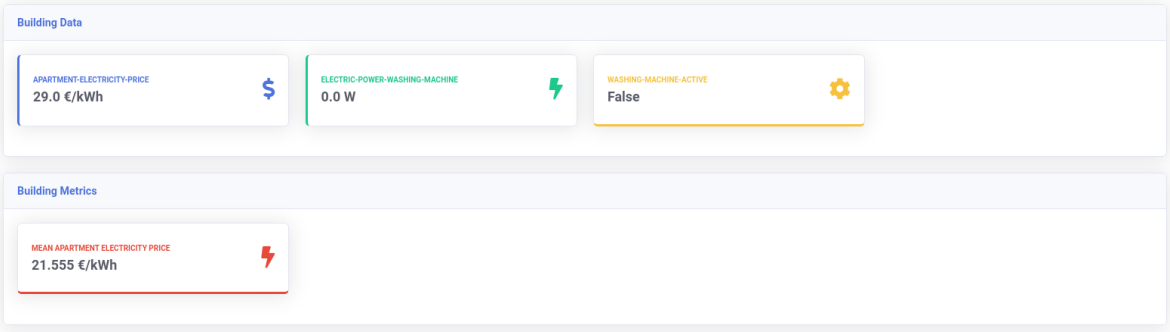

Figure 27: User Interface Elements: Cards

Figure 28 shows the two possible chart types. First an area chart, second a bar chart. On page load all existing chart elements will be setup. Therefore, the chart's linked Datapoint is requested and his last update timestamp is read. Outgoing from this timestamp and the chart's interval settings the last n timestamps will be calculated. For example, the chart is configured to show data in a hourly interval. Therefore, the timestamps of the last 24 hours will be calculated. These timestamps are used to request the required Datapoint values. The values are collected and used to setup the chart. Besides the datasets, the x-axis' labels will be created of the timestamps, the y-axis' labels will be calculated of the highest presented values, the chart's legend is set as the Datapoint's short name, and the chart's colors are predefined.
These information are given to the JavaScript chart library Chart.js that is also used in the sb-admin-2 Bootstrap template. The library then handles printing the chart onto a HTML canvas object. Via the dropdown link at the top right of the chart the dataset intervals may be changed. On change, the process of calculating the requested timestamps and collecting the data is started for the selected time interval.

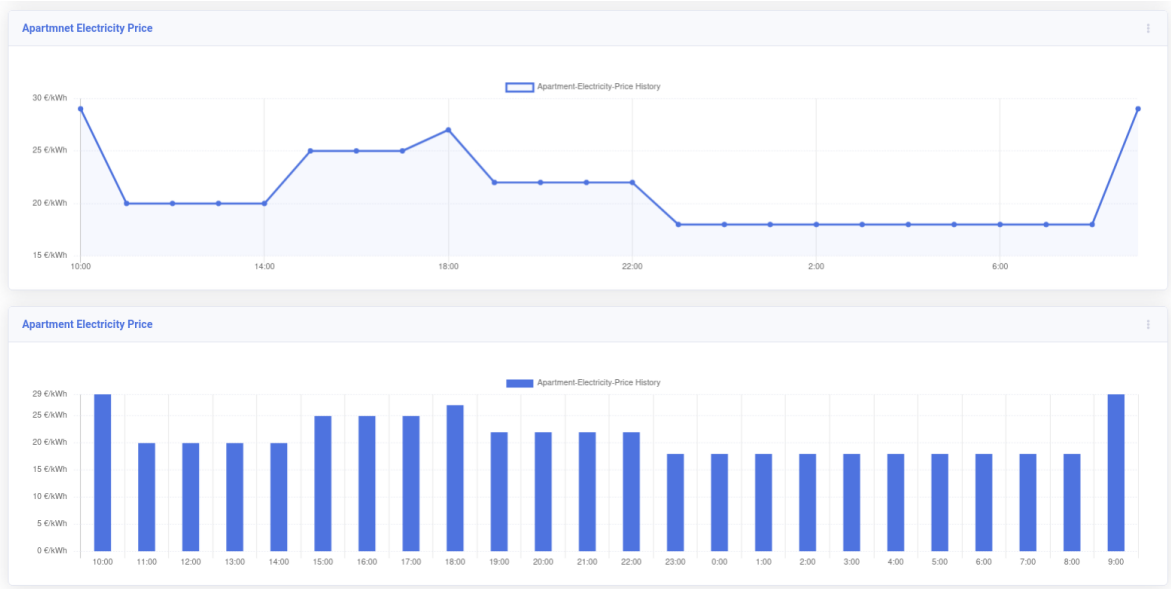

Figure 28: User Interface Elements: Charts

When it comes to EvaluationSystemPages used as comparison page the same user interface elements and implementations are used. For a better comparison the page will be split into half. Each page side for one of the two algorithms to compare. The left and the right side will be filled with the same user interface elements that are configured at the EvaluationSystemPage. Figure 29 shows this double column setup.

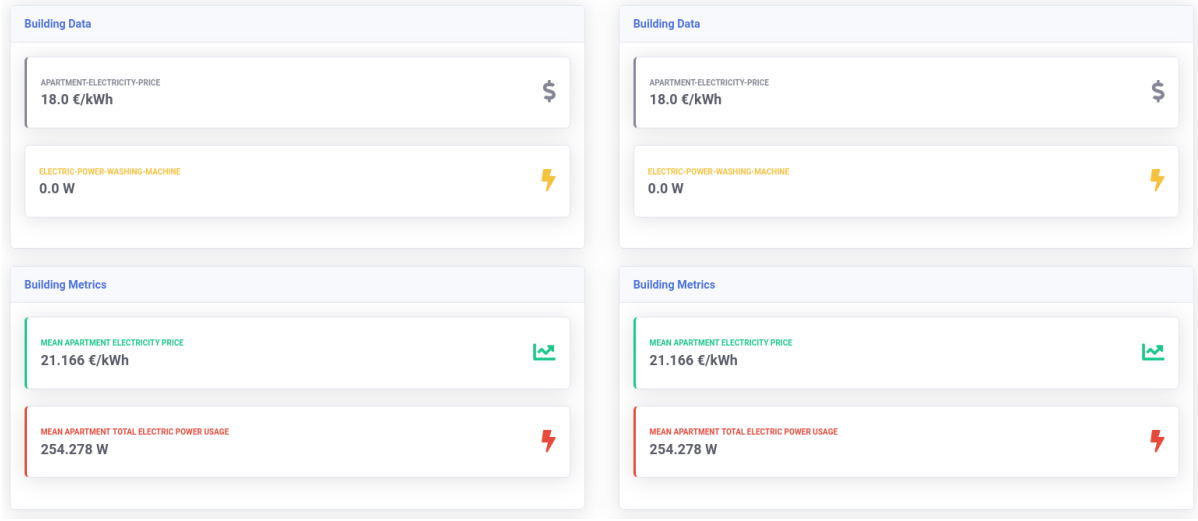

Figure 29: Comparison Page: Cards

Unlike dashboard pages, comparison pages can not be filled with data on page load. The data first needs to be simulated and requested form the simulation API. Therefore, a

simulation setting form was included at the top of each comparison page. This form, as seen in figure 30, consists of two selection forms to select the algorithms to compare and two date picker forms to chose the simulation's start and end date. Additionally, a button which starts the simulation was added.

Figure 30: Comparison Page: Settings

Before starting the simulation the comparison columns are hidden. On simulation start a text field is shown. This text field shows the estimated waiting time for the left and the right algorithm simulation. If one of the simulations is finished the results will be requested and processed in the same way as at a dashboard page. Since one simulation result data set is only represented by one of the two data columns, users have to wait for both simulations to finish. Besides the plain values for comparison, there are charts, as seen in figure 31, as well.

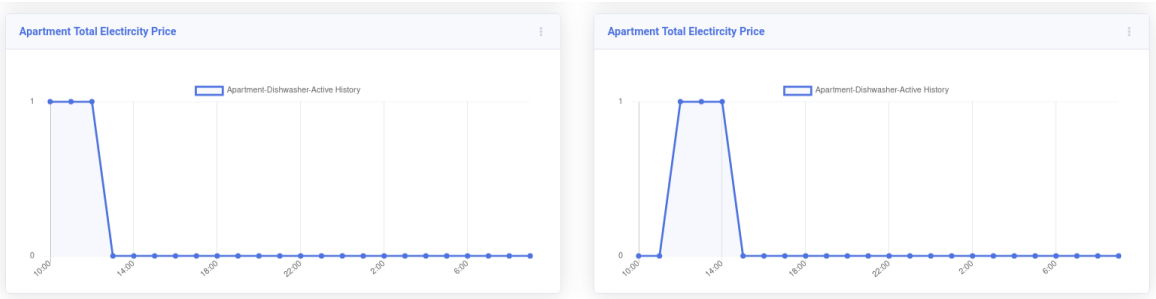

Figure 31: Comparison Page: Charts

In addition to this charts the comparison pages may include special comparison graphs. In figure 32 one can see one of these charts. They are located at the bottom of each comparison page and are only shown and filled with data when both simulations are finished.

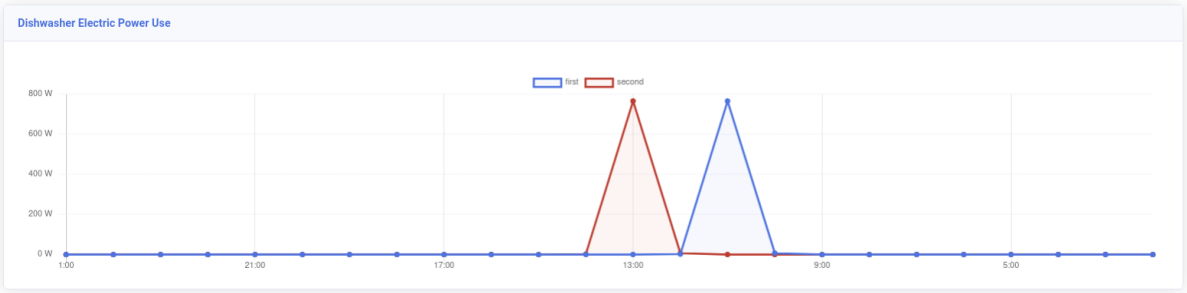

Figure 32: Comparison Page: Comparison Chart

Both at dashboard and comparison pages error messages were implemented. These may be raised on client or server side errors. To raise them a simple JavaScript is used that builds the error message object and shows it at the front end. Error messages always are included at the page's top. In figure 33 a error message is shown that is raised on wrong user input. Wrong user input and an negative API return statuses are the main use case for these error messages.

Figure 33: Error Message

6 Evaluation

In this chapter the user interface for visualization of the potential of building energy management algorithms, implemented in chapter 5, will be evaluated. Therefore, the requirements from chapter 5, the design best practices from chapter 3, the scenarios and their use cases form chapter 2.2, and the learings of the state of the art evaluation in chapter 4 will be used.

6.1 Evaluation of the Requirements

The application requirements were build of the scenario use cases and were used as guideline of the design and implementation process. In order to evaluate the required features the scenarios of section 2.2 will be gone through step by step, inspired by the *cognitive web walkthrough* method [5]. As user goal we uses the goals of each use case scenario. In the walkthrough it will be described how a user could achieve this goal using the developed application. This on the one hand reviews the requirements and on the other hand evaluates a typical user work flow.

Use Case One: Single Family Building

As the owner of the single family building is about to configure his own dashboard, he visits the admin panel of his energy management system, logs in, creates a new EvaluationSystemPage object, or chooses an existing one for configuration. Since his goal is to monitor his energy consumption, the energy market price, and the composition of his energy consumption, he might use Card objects as presentation objects for the simple values like his building's actual energy consumption and the actual energy market price. For more complex data sets like an energy consumption history he may set up an area graph for the appropriate DataPoint and configure the graph with the intended data point's history using for example, a hourly or daily interval. To monitor his energy consumption's composition he sets up a bar chart, which compares the percentages of his consumed energy types.
As he is also about to register a new device data source, he stays at the admin panel's web page, creating a new DataPoint object, configuring it for his needs. This object in turn may be presented by another Presentation object at his dashboard pages.
After this setup process he is able to visit his new dashboard at the frontend and monitor the requested data.

Use Case Two: Multi-Family Building

The multi-family building resident wants to analyze his apartment's data first of all has to log in. As the building management system runs for the whole building, every resident has it's own login credentials. With these he is able to login at the frontend and at the admin panel viewing and configuring pages for which he has the appropriate rights.
As the resident used Card and Chart objects to present and monitor his data, he configures them at the admin panel at his will. To support his visual senses he uses the card colors, icons and decorative possibilities. Therefore, he designs his own unique dashboard, perfectly structured and designed for his needs.

Use Case Three: Office Building

As the office building management system analyst's work is all about comparing different optimization algorithms and settings, he is mainly using the application's comparison tool. Therefore, he configures a EvaluationSystemPage object as a comparison page. He adds Presentation objects for all required DataPoints and configures them after his wishes. Also he creates Algorithm objects for each backend optimization simulation algorithm he wants to compare. Each of these objects holds the right backend identifier so the analyst is able to request simulation results at the frontend. For more detailed monitoring he also creates some useful Metrics. For example, a Metric object that multiplies the energy consumption with a constant and, therefore, results in a carbon dioxide measure. Additionally, he includes and configures some ComparisonGraphs to his comparison page. Afterwards, he visits the frontend and chooses two algorithms to compare. He then uses the two comparison columns and the comparisons graph to find the optimal settings for his building.

6.2 Evaluation of the Design Best Practices

In the following it is reviewed if and how the design best practices from chapter 3 are used in the development of the user interface for visualization of the potential of building management algorithms. The ten best practices will be evaluated step by step.

DP01: Simple and Natural Dialogue The application's user interface is designed simple and without showing any irrelevant or rarely needed information. The concept of tooltips used in cards or charts allows to hide additional information that can be received when hovering over certain user interface elements. The frontend is designed in a simple color scheme and uses less complex user interface elements. Therefore, the interface appears clearly and simple designed.

As well as the frontend, the admin panel is designed as simple as possible. The custom admin scripts hiding unnecessary information contribute to this as well as the simple structure.

DP02: Speak the User's Language As the frontend is designed with the Bootstrap framework and is also using Fontawesome icons, many well known interface elements, like a top and a navigation bar or well known icons, are present. Also the page structure similar to many other webpages. Therefore, users should understand simple interface elements right away. Furthermore, both the admin pages and frontend pages use simple and well know HTML form elements. Elements, that every user should have used already. This should ease the use the application.
Besides common web page concepts, simple language is used to describe admin panel functionality in help texts, improving the usability even more.

DP03: Minimize the User's Memory Load As already evaluated for design best practice number one and two: Only important information are presented to the users. Therefore, users memory load is minimized. Also the comparison page structure is made from minimizing user's memory load, as data can be compared side by side without switching pages or even scrolling.

DP04: Consistency As Bootstrap is used as CSS framework there are no inconsistencies in structural design. Also with only cards and charts as possible user interface elements and their consistent design, users do not have to learn unfamiliar user interface elements.
In terms of presented data, there are also no inconsistencies possible, as the data always is updated at the same time intervals and, therefore, there is no possibility of presenting two different data values for the same data point and, thereby, an inconsistent data state.

DP05: Feedback In terms of system feedback there are multiple features informing users about missing data, system errors, or exceptions. If card data is missing there is an extra card caption implying missing data. If anything goes wrong in terms of user input or data computation there may be error messages at the top of user interface elements. Beside these user interface feedback features there are also detailed error messages for debugging in the browser's console log.

DP06: Clearly Marked Exits Since the pages are completely built without popups, modals, or more than one link deep page structures, there simply are no exits to be marked. Users navigate with the easiest way possible: The navigation bar.

DP07: Shortcuts As already evaluated, there are no complex page structures that need user shortcuts. But the EvaluationSystemPages may be designed as complex as users

wish. The Metric objects may be used to computed complex key figures that might save the user a lot of comparison work. These may be considered as shortcuts for experienced users.

DP08: Good Error Messages Every possible error message of this system is written simple and describes the error and it's cause.

DP10: Help and Documentation The application provides detailed help and documentation. During the development the systems code was commented very extensively. The admin pages provide help texts for every user input form, informing novice users about their possibilities. Also system administrators are able to provide help texts as tooltips or descriptions to their users with the help of the admin panel's configuration possibilities.

This evaluation shows that every of the ten listed design best practices were implemented very conscientiously, always having the usability of the application in mind. Therefore, the application is well designed for new and inexperienced users. But also for the use by professional users and for complex data analysis.

6.3 Evaluation of the State of the Art Learnings

As last step of the evaluation it is to be compared whether the learnings from chapter 4 were taken over into this thesis' application.

Simple User Interface The user interface is kept as simple as possible, as already pointed out in the previous evaluation steps. There are no additional unnecessary features and no visual decorations when they are not needed. Therefore, the learnings about simple user interfaces were implemented very well in this thesis' application.

User Interface Elements During the state of the art evaluation it was determined that it is helpful to categorize and group data and user interface elements. Also the possibility of having multiple dashboard pages was denoted positively. Both of these aspects were well implemented in the user interface. UIElementContainers enable grouping data and user interface elements. This allows page admins to already group data visually for users. Also due to the generic system structure it is easily possible to create multiple pages, even with the same content. Therefore, clean, structured, and clear pages are made possible. In terms of data presentation elements the suggestions of the state of the art analysis were adopted, including all positively mentioned user interface element features, like tooltips, descriptive titles, chart legends, and chart configuration possibilities.

Configurability The generic structure of this thesis' application paired with the use of Django's admin page and a content management like setup, results in full configurable web pages. User interface element configurability, page configurability, and system configurability were all made possible with the help of this system architecture. Therefore, the application implements all configurability features pointed out in the state of the art analysis.

Help and Documentation Unlike the introduced applications in chapter 4, this thesis application does not lack help texts or documentation. The features both at the admin pages and at the frontend are well described for new and inexperienced users. Help texts describe functionality for admin user input forms and it is possible to provided tooltips for every user interface element.

Algorithm and Parameter Comparison Also the comparison page did not adopt erros made in other applications. Data values may be compared side by side, reducing users memory load and providing easy comparison tools from both simple and complex data analysis.

Overall all errors listed in chapter 4 analysis were avoided and the learnings were implemented very well, resulting in a well designed application, even for complex data monitoring.

7 Conclusion and Outlook

In the final chapter this thesis will be summarized. Afterwards, a conclusion will be drawn about the findings of this thesis. Finally, there will be a short outlook on how this thesis could be continued through possible further work.

7.1 Conclusion

This thesis contributes to the fields of *energy informatics* by developing a user interface for visualization of the potential of building energy management algorithms. Therefore, the base concepts and challenges of the building energy management were summarized and scenarios were developed thereof. Afterwards, these scenarios and common web design best practices were used to evaluate three state of the art energy management applications. Then again, the evaluation results, the design best practices and the scenarios were used to develop a web application, which is designed for energy management algorithm comparison and monitoring. The application supports both simple and complex analysis processes and is, therefore, suitable for both inexperienced and professional users. In order to review the development objectives, the application was evaluated in the same way as the state of the art was and, thereby, the well executed design and implementation of the development of a user interface for visualization of the potential of building energy management algorithms was confirmed.

7.2 Outlook

There are several ways in which the work of this thesis may be continued. The developed application needs to be tested in a real building setting and may be evaluated by users in the context of a study. Since this thesis' focus was on residential and office buildings, it might be interesting to focus on the complex environment of other building types like industrial buildings. As these come in greater varieties, more detailed and complex monitoring possibilities are needed. Therefore, an evaluation of those settings might be reasonable.
Besides that, the application may be used to analyze and evaluate existing building energy optimization algorithms which may be further improved by the evaluation results.
Including more features to this thesis' developed application might also be conceivable. Additional chart types or more complex mathematical functions for data metrics may be such features which could help increase the application's utility.
The abovementioned concepts indicate that the work of this thesis holds many innovations in store for future research and development.

References

[1] *Towards a zero-emission, efficient, and resilient, buildings and construction sector. Global Status Report 2017.* UN Environment and International Energy Agency, 2017.

[2] Barnum, C. M. *Usability Testing Essentials - Ready, Set...Test.* Elsevier Inc., 60 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States, 2021.

[3] Becker, B. *Interaktives Gebäude-Energiemanagement.* PhD thesis, Karlsruher Institut für Technologie, 2014.

[4] Bendel, O. Smart metering, 2018.

[5] Blackmon, M. H., Polson, P. G., Kitajima, M., and Lewis, C. Cognitive walkthrough for the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2002), CHI '02, Association for Computing Machinery, p. 463–470.

[6] BMU, A. I. I. . Klimaschutzprogramm 2030 - maßnahmen zur erreichung der klimaschutzziele 2030. Tech. rep., Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit (BMU), Referat Öffentlichkeitsarbeit, Online-Kommunikation, Social Media · 11055 Berlin, October 2019.

[7] Bundesregierung, D. Arbeitsstättenverordnung vom 12. august 2004 (bgbl. i s. 2179), die zuletzt durch artikel 226 der verordnung vom 19. juni 2020 (bgbl. i s. 1328) geändert worden ist.

[8] Buratti, C., Vergoni, M., and Palladino, D. Thermal comfort evaluation within non-residential environments: Development of artificial neural network by using the adaptive approach data. *Energy Procedia 78* (11 2015), 2875–2880.

[9] Bögel, J., Mesecke, F., and Schult, C. *Gebäude- und Wohnungsbestand in Deutschland - Endgültige Ergebnisse.* Statistische Ämter des Bundes und der Lände, 2015.

[10] Carli, R., and Dotoli, M. Energy scheduling of a smart home under nonlinear pricing. In *53rd IEEE Conference on Decision and Control* (2014), pp. 5648–5653.

[11] Czaja, S. J., Boot, W. R., Charness, N., and Rogers, W. A. *Designing for Older Adults - Priciples and Creative Human Factors Approaches.* CRC Press - Taylor and Francis Group, 6000 Broken Sound Parkway NW, Suite 300´, 2019.

[12] Deutsche Energie-Agentur (dena, . dena-gebÄudereport kompakt 2016 - statistiken und analysen zur energieeffizienz im gebäudebestand. Tech. rep., Deutsche Energie-Agentur(dena), 2016.

[13] Electric, S. Schneider electric website, 2020.

[14] Electric, S. Wiser energy playstore download, 2020.

[15] e.V., V. V. D. I. Vdi guideline 4608 - energy systems - combined heat and power - terms, definitions, examples. Tech. rep., VDI Verein Deutscher Ingenieure e.V., 2005.

[16] e.V., V. V. D. I. Vdi guideline 4602 - energy management - terms, definitions. Tech. rep., VDI Verein Deutscher Ingenieure e.V., 2007.

[17] Facebook. Facebook brand resources, 2019.

[18] Foundation, D. S. Django project, 2005.

[19] für Wirtschaft und Energie, B. Zeitreihen zur entwicklung der erneuerbaren energien in deutschland. Tech. rep., Bundesministerium für Wirtschaft und Energie, 2020.

[20] glabs. Homegenie, 2012.

[21] Google. Material design guidelines, since 2015.

[22] Haynes, B. The impact of office comfort on productivity. *Journal of Facilities Management 6* (02 2008), 37–51.

[23] Hayter, S. J., and Kandt, A. 48th aicarr international conferencebaveno-lago maggiore, italy. In *Renewable Energy Applications for Existing Buildings* (August 2011).

[24] Hornbæk, K. Current practice in measuring usability: Challenges to usability studies and research. *International Journal of Human-Computer Studies 64*, 2 (2006), 79 – 102.

[25] Huang, W. Z., Zaheeruddin, M., and Cho, S. Dynamic simulation of energy management control functions for hvac systems in buildings. *Energy Conversion and Management 47*, 7 (2006), 926 – 943.

[26] Icha, P., and Kuhs, G. Entwicklung der spezifischen kohlendioxid-emissionen des deutschen strommix in den jahren 1990 - 2019. Tech. rep., Umweltbundesamt, April 2020.

[27] Jochem, P., Kaschub, T., and Fichtner, W. How to integrate electric vehicles in the future energy system?

[28] Koch, S., Krieger, O., and Müller, C. *dena-Analyse - Insight Büroimmobilien - Marktsituation und Ausblick für klimafreundliche Bürogebäude*. Deutsche Energie-Agentur GmbH (dena), 2018.

[29] Kotler, P. Prosumers: A new type of consumer. *The Futurist 20* (1986), 24–28.

[30] Kott, K., and Behrends, S. *Ausstattung mit Gebrauchs-gütern und Wohnsituationprivater Haushaltein Deutschland - Ergebnisse der Einkommens- und Verbrauchsstichprobe 2008.* Statistisches Bundesamt, Wiesbaden, 2009.

[31] Kretzschmar, D., Schiller., G., and Weitkamp, A. *Nichtwohngebäude in Deutschland – Typisierung eines dynamischen Marktes.* Wißner-Verlag, 2019.

[32] Kreuzer, K. openhab demo website, 2020.

[33] Kreuzer, K. openhab website, 2020.

[34] Lee, E., and Bahn, H. Electricity usage scheduling in smart building environments using smart devices. *TheScientificWorldJournal 2013* (12 2013), 468097.

[35] Markov, D. Practical evaluation of the thermal comfort parameters. *Annual International Course: Ventilation and Indoor climate, Avangard, Sofia, 2002, P. Stankov (Ed), pp. 158 – 170, ISBN 954-9782-27-1* (10 2002).

[36] Mauser, I. *Multi-modal Building Energy Management.* PhD thesis, Karlsruher Institut für Technologie, 2017.

[37] Miller, D. sb-admin-2, 2013.

[38] Molich, R., and Nielsen, J. Improving a human-computer dialogue. *Commun. ACM 33*, 3 (Mar. 1990), 338–348.

[39] Nielsen, J. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1994), CHI '94, Association for Computing Machinery, p. 152–158.

[40] Nielsen, J. 113 design guidelines for homepage usability, October 2001.

[41] Paschotta, R. Blockheizkraftwerk, 2020.

[42] Paschotta, R. Photovoltaik, 2020.

[43] Ruggiero, S., Varho, V., and Rikkonen, P. Transition to distributed energy generation in finland: Prospects and barriers. *Energy Policy 86* (08 2015).

[44] Saaty, R. The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling 9*, 3 (1987), 161 – 176.

[45] Schiefer, M. Smart home definition and security threats. In *2015 Ninth International Conference on IT Security Incident Management IT Forensics* (2015), pp. 114–118.

[46] Schlomann, B., Wohlfahrt, K., Kleeberger, H., Hardi, L., Geiger, B., Pich, A., Gruber, E., Gerspacher, A., Holländer, E., and Roser, A. *Energieverbrauch des Sektors Gewerbe, Handel, Dienstleistungen (GHD) in Deutschland für die Jahre 2011 bis 2013. Schlussbericht an das Bundesministerium für Wirtschaft und Energie (BMWi), Karlsruhe, München, Nürnberg.* 2015.

[47] Scholz, D. *Optimierung interaktiv.* Springer-Verlag GmbH Deutschland,, 2018, ch. 7, pp. 169–187.

[48] Schreiber, M., Wainstein, M. E., Hochloff, P., and Dargaville, R. Flexible electricity tariffs: Power and energy price signals designed for a smarter grid. *Energy 93* (2015), 2568 – 2581.

[49] Steinmeier, F. W., Merkel, A., and Schulze, S. Gesetz zur einführung eines bundes-klimaschutzgesetzesund zur Änderung weiterer vorschriften, December 2019.

[50] Tullis, T., and Albert, B. *Measuring the User Experience - Collecting, Analyzing and Presenting Usability Metrics.* Elsevier Inc., 225 Wyman Street, Waltham, MA, 02451, USA, 2013.

[51] Twitter. Bootstrap, 2010.

[52] United Nations, N. Y. Paris agreement as available on https://treaties.un.org/pages/ctcs.aspx, December 2015.

[53] Wirtschaftslexikon, G. Pareto-optimum, February 2018.

[54] Zeng, M., Huang, L., Yan, F., and Jiang, D. Notice of retraction: Research of the problems of renewable energy orderly combined to the grid in smart grid. In *2010 Asia-Pacific Power and Energy Engineering Conference* (2010), pp. 1–4.

[55] Zipf, M., and Most, D. Impacts of volatile and uncertain renewable energy sources on the german electricity system. *2013 10th International Conference on the European Energy Market (EEM)* (2013), 1–8.

Assertion

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, February 16, 2021 Lukas Landwich