

Fachrichtung Informatik

Schuljahr 2017/2018

# DIPLOMARBEIT

Gesamtprojekt

# AEMS – Advanced Energy Monitoring System

**Ausgeführt von:**

Niklas Graf, 5BHIF-2

Lukas Knoll, 5BHIF-8

Sebastian Mandl, 5BHIF-11

Grieskirchen, am 25.03.2018

**Betreuer:**

DI Josef Doppelbauer

---

**Abgabevermerk:**

Datum: 25.03.2018

Betreuer: DI Josef Doppelbauer

## **Erklärung gemäß Prüfungsordnung**

„Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.“


Grieskirchen, 01.04.2018

Verfasser/innen:

Niklas Graf

Lukas Knoll

Sebastian Mandl


	<b>HTBLA Grieskirchen</b>
	Fachrichtung: <b>Informatik</b>

## DIPLOMARBEIT DOKUMENTATION

<b>Namen der Verfasser/innen</b>	Niklas Graf Lukas Knoll Sebastian Mandl
<b>Jahrgang Schuljahr</b>	2017/18
<b>Thema der Diplomarbeit</b>	AEMS – Advanced Energy Monitoring System
<b>Kooperationspartner</b>	Energiegenossenschaft Region Eferding


<b>Aufgabenstellung</b>	<p>Entwicklung eines Bots, welcher die AMIS-Zählerdaten ausliest und diese in einer Datenbank speichert.</p> <p>Eigene Zähler für z.B. Strom und Wärmemenge auf Raspberry Pis aufsetzen.</p> <p>Alle gesammelten Daten werden in Form von Statistiken grafisch aufbereitet.</p> <p>Weiters sollen in festgelegten Zeitabständen Berichte über die Verbrauchswerte generiert werden.</p> <p>Downloadfunktion für Statistiken und Berichte.</p> <p>Entwicklung einer Anomalieerkennung bei Abweichung der Verbrauchswerte.</p> <p>Benachrichtigungssystem um bei Verbrauchsabweichungen benachrichtigt zu werden.</p> <p>Entwicklung einer Android-App, um sich am Smartphone Statistiken anzusehen, die Möglichkeit zu haben diese herunterzuladen und bei Verbrauchsabweichungen eine Benachrichtigung zu erhalten.</p> <p>Administrationstool für die Verwaltung der Zugriffsrechte.</p>
-------------------------	---

<b>Realisierung</b>	<p>Entwicklung der Android-App mithilfe von Androidstudio V2.3. Für die Statistiken wird die Library MPAndroidCharts verwendet. Zur Kommunikation mit dem Server werden die eigens entwickelte Library aems-api-lib und der Java REST Dienst verwendet. Alle weiteren Funktionen der Android-App wurden selbst entwickelt.</p> <p>Erstellung des Webinterfaces unter Zuhilfenahme der CSS-Library Bootstrap und Erstellung der Modals mithilfe von clean-modal-login-form. Für das Webinterface wurde die Sprache xhtml verwendet und in Kombination damit Java Server Faces. Für weitere Funktionalität wurden jQuery und Javascript verwendet. Für das Anzeigen der Statistiken wurde die Chart-Library Chart.js genutzt und der Download als PDF wurde mit der Library jsPDF ermöglicht. Das Webinterface bietet die Möglichkeit sich auf der Startseite seine gewünschten Statistiken anzeigen zu lassen, sich Schnellauswertungen anlegen und herunterladen zu können, Statistiken anzulegen, Statistiken für die Startseite und Android-App zu konfigurieren, Berichte für gewünschte Perioden einzurichten und sich Richtlinien für Benachrichtigungen zu erstellen.</p>
---------------------	---

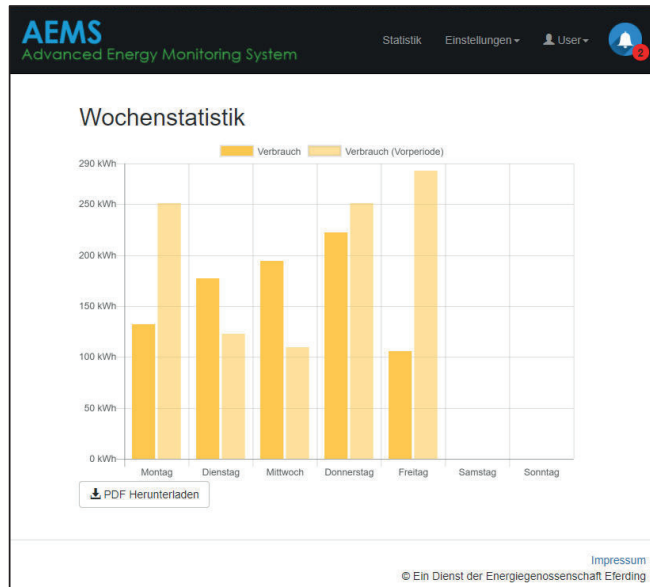
	<p style="text-align: center;"><b>HTBLA Grieskirchen</b></p>
	<p>Fachrichtung: <b>Informatik</b></p>

	<p>Das Administrationstool wurde auch als xhtml-Webinterface mit Hilfe von Bootstrap, clean-modal-login-form, jQuery, Javascript und JSF erstellt. Hier gibt es die Möglichkeit die Freigaben für das System zu betreuen, Administratoren zu verwalten und sich seine Zuständigkeitsbereiche als Administrator zu definieren.</p> <p>PostgreSQL-Datenbank für die Speicherung der Daten.</p> <p>Der Report-Bot, welche die Daten aus dem netzonline Portal herunterlädt und in die Datenbank speichert, wurde mit Java realisiert. Dazu wurde die Klassenbibliothek „HTMLUnit“ verwendet, um die Interaktion mit dem netzonline Portal zu ermöglichen, da es dafür keine API oder ähnliches gibt. Des Weiteren wurde die Bibliothek „Apache POI“ verwendet, um die benötigten Daten aus den heruntergeladenen MS-Excel Dateien zu extrahieren.</p> <p>Über eine Java REST Schnittstelle wird die Kommunikation zwischen der Datenbank und den einzelnen Clients realisiert. Diese Schnittstelle bildet eine Zwischenschicht, welche ein immer gleichbleibendes Kommunikationsprotokoll garantiert.</p> <p>Mit Hilfe eines Raspberry Pis werden verschiedene Typen von Zählern ausgelesen, diese Daten werden dann über eine REST-Schnittstelle an die Datenbank gesendet und dort gespeichert. Das Erfassen der Zählerdaten wird über ein Plugin-System bewältigt, welches dynamisch Plugins in den Verarbeitungsprozess miteinbindet.</p> <p>In Verbindung mit Home-Automation wurde eine Schnittstelle zum Open-HAB errichtet. Über diese Schnittstelle ist es möglich die Daten der Zähler, also z.B.: den aktuellen Verbrauch, einzusehen.</p>
--	--

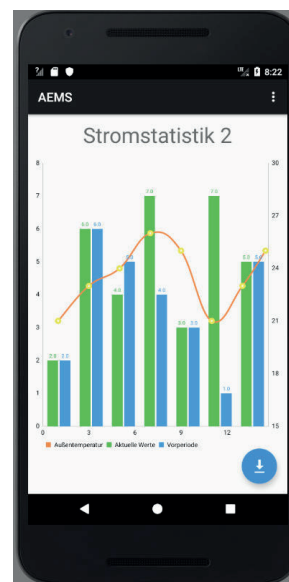
<p><b>Ergebnisse</b></p>	<p>Funktionierender Bot, welcher die Daten ausliest und in die Datenbank einspeist.</p> <p>Funktionierende Client/Server Infrastruktur zwischen den verwendeten Diensten.</p> <p>Funktionierende Dienste Webinterface, Administration-Webinterface, Android-App, Raspberry PI Zähler, aems-apilib, Datenbank und Java REST Service.</p> <p>Jegliche Anforderungen des Auftraggebers wurden erfüllt.</p>
--------------------------	---

	<b>HTBLA Grieskirchen</b>
	Fachrichtung: <b>Informatik</b>


Typische Grafik, Foto etc.  
(mit Erläuterung)

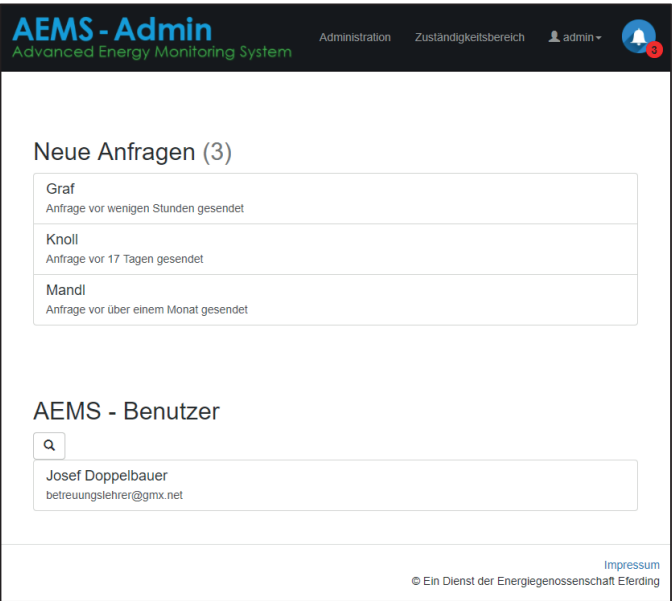


Übersicht der Startseite. Oben befindet sich die Navigation-Bar mit den Menüs, dem Benutzer und dem Benachrichtigungssystem. Der Seiteninhalt selbst besteht aus ausgewählten Statistiken, welche für den Download bereitstehen.



Ansicht einer Statistik in der Android-App. Oben befindet sich die Menübar mit den Funktionen zur Anzeige aller Benachrichtigungen und dem Logout. Der Inhalt zeigt eine Statistik mit aktuellen Verbrauchswerten, Verbrauchswerten der Vorperiode und einer Anomalie. Weiters gibt es die Möglichkeit zum Download der Statistik.


	<b>HTBLA Grieskirchen</b>
	Fachrichtung: <b>Informatik</b>

	 <p>Übersicht der Administrationswebsite. In der Navigation-Bar sind die verfügbaren Menüs eingeblendet, der verwendete Nutzer und das Benachrichtigungssystem. Im Seiteninhalt werden die neuen Anfragen und die bereits von einem Administrator verwalteten Benutzer angezeigt. Diese können durch eine Suchfunktion einfacher gefunden werden.</p>
--	---

<b>Teilnahme an Wettbewerben, Auszeichnungen</b>	Bosch – Technik fürs Leben Preis 2018 Weitblick – Champions Jugend Innovativ – Sonderpreis Sustainability  Da die Wettbewerbe erst nach Beendigung der Diplomarbeit ausgewertet werden liegen noch keine Informationen zu Auszeichnungen bereit.
--	--

<b>Möglichkeiten der Einsichtnahme in die Arbeit</b>	Der Quellcode dieser Diplomarbeit unterliegt auf Wunsch des Projektteams der Geheimhaltung und steht nur der Prüfungskommission zur Einsicht zur Verfügung. Eine Version der Diplomarbeit wird im Archiv verwahrt und ist für die öffentliche Einsichtnahme gesperrt.
--	---

<b>Approbation (Datum/Unterschrift)</b>	Prüfer/Prüferin	Direktor/Direktorin
---	-----------------	---------------------

	<b>HTBLA Grieskirchen</b>
	Department: <b>Informatik</b>


## DIPLOMA THESIS

### Documentation

<b>Author(s)</b>	Niklas Graf Lukas Knoll Sebastian Mandl
<b>Form Academic year</b>	2017/18
<b>Topic</b>	AEMS – Advanced Energy Monitoring System
<b>Co-operation partners</b>	Energiegenossenschaft Region Eferding


<b>Assignment of tasks</b>	<p>Development of a bot which extracts data from AMIS-meters and stores it in the database of the AEMS-System.</p> <p>Define customized meters for instance electricity-meters, heat-meters and many more and set them up on your personal Raspberry-PI.</p> <p>The entirety of data will be processed and displayed as statistics. Furthermore the generation of reports which specify time constraints are possible as well. The aforementioned functionalities also provide download capability.</p> <p>Additionally the detection of anomalies is provided by an integrated scripting language called AEMS-scripting-language (AEMSSL). The main aim of the scripting language is the detection of deviation of consumption values either caused by meters themselves or sensors affecting the meters values (day-time, light-intensity).</p> <p>Moreover an integrated notification system supports easy recognition of abnormal meter data.</p> <p>Development of an android-app to display priorly defined and generated statistics. The android-app also comes along with the download capability of statistics and notifies the user if abnormal meter data is noticed by the system.</p> <p>Administrative tool for the management of access rights to the system.</p>
----------------------------	--

<b>Realisation</b>	<p>Android app development was supported by the IDE Android Studio V.2.3. For the generation of statistics the MPAndroidCharts library was consulted. The communication between the server and the client is managed by a REST-API as well as by a self-created library AEMS-API-LIB. Every further functionality found in the android-app was entirely self-created.</p> <p>Creation of the web-interface was supported by several libraries namely CSS, Bootstrap. Additionally modals were created by the help of clean-modal-</p>
--------------------	---

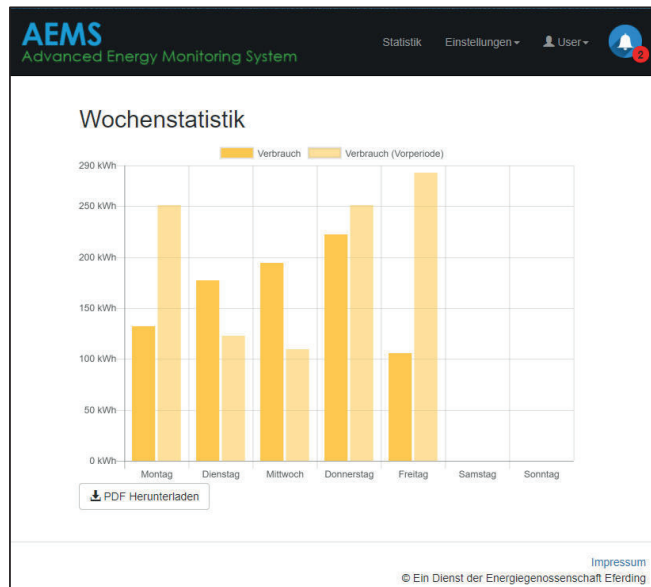
	<b>HTBLA Grieskirchen</b>
	Department: <b>Informatik</b>

	<p>login-form.</p> <p>The language in which the web-interface was designed was chosen to be XHTML and was utilized in conjunction with Java-Server-Faces (JSF). Moreover jQuery and JavaScript were used to implement further functionality.</p> <p>The library which grants the display of statistics is named Chart.js And the one which provides the capabilities for downloading PDFs is named jsPDF.</p> <p>The web-interfaces supplies the user with the possibility to display the desired statistics which the user wants to have immediate access to as well as to define quick statistics. These statistics can furthermore be subject to download or configuration. Possible configuration would be to only let the statistic appear on the smartphone app or to declare time constraints on reports or statistics to only receive reviews of periods that the user demands. In Addition guidelines for the detection of anomalies can be specified which will be taken under advisement by the unit which is accountable for processing the data received from the meters and sensors (AEMSSL).</p> <p>The administrative tool was also chosen to be designed with XHTML, jQuery, JavaScript, JSF and Bootstrap. This tool permits the administration of access rights hence the declaration of new administrators and moreover to specify the exact geographical realms where the administrators operate in.</p> <p>PostgreSQL-Database as a means of data storage.</p> <p>The report-bot is in charge of downloading the data from the NETZONLINE-page which then gets stowed in the database of the system. This program was entirely implemented with the JAVA programming language. In order to had implemented the communication with the NETZONLINE-page a special library named HTMLUnit was consulted. For the extraction of the data from the excel files which were downloaded by the report-bot another library named Apache POI was utilized.</p> <p>A Java REST-API enables the communication between the database and the several clients. The aforementioned REST-API provides a logical layer for all database requests performed via GraphQL to guarantee a never changing communication protocol.</p> <p>With the utilization of a Raspberry PI the data of various different meter types is able to be read from. After the read the data is transmitted to the database via the REST-API which then is responsible for creating the proper SQL statements to store the data properly in the database. The acquisition of meter data is performed by a Plugin-System which is capable of dynamically including user-defined Plugins into the processing environment.</p> <p>In conjunction with home automation an interface to Open-HAB was established. Via this interface it is possible for the user to retrieve the current meter value from the server and to display it.</p>
<b>Results</b>	<p>Functional report-bot, which retrieves data and stores it in the database.</p> <p>Functional client/server infrastructure between the utilized services.</p> <p>Functional services are web-interface, administrative tool, android-app, Raspberry PI meter data extraction, aems-apilib, database and Java REST-API.</p> <p>Every functionality demanded by the customer was satisfied.</p>

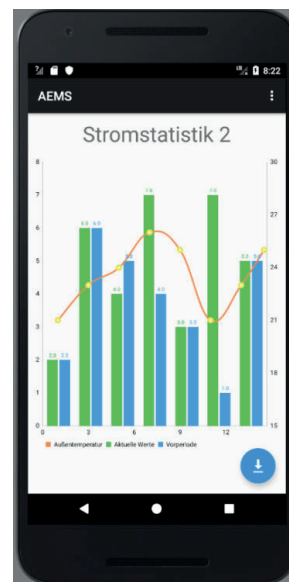


	<h2 style="text-align: center;">HTBLA Grieskirchen</h2> <tr> <td style="text-align: right;">Department:</td> <td style="text-align: center;">Informatik</td> </tr>	Department:	Informatik
Department:	Informatik		


Illustrative graph, photo  
(incl. explanation)

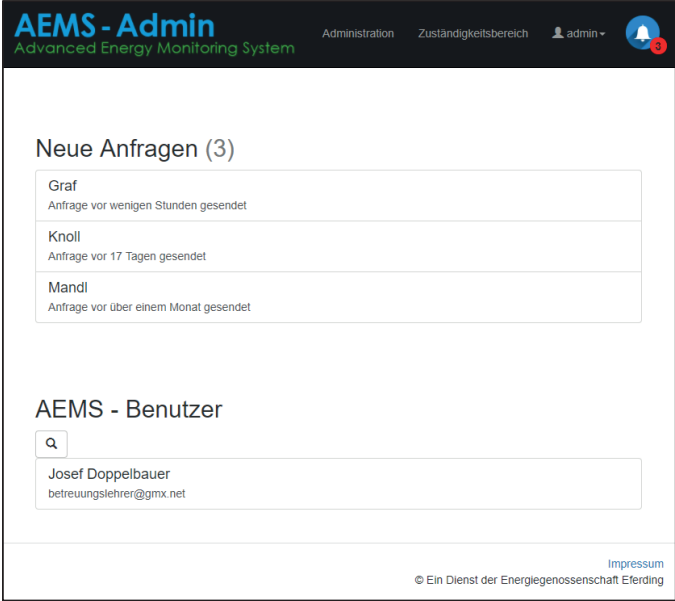


Dashboard. On the top one can find the navigation bar with the menus, the user and the notification system. The content of the page itself is comprised of user selected statistics which are available for download.



View of a statistic in the android-app. On the top one can find the navigation bar which provides the functionalities of displaying the current notifications accessible by the user as well as layout changes. The view depicts a statistic with current consumption values and consumption values of the pre-period and the corresponding anomaly. Furthermore there is the possibility to download the statistics.

	<b>HTBLA Grieskirchen</b>	
	Department:	<b>Informatik</b>

		<p>The view of the administrative tool web-page. The navigation bar displays the menus accessible by the user as well as the currently logged-in user and the notification system. The content of this picture illustrates incoming request for usership. These have to be accepted in order to be viewed as a user of the system. Previously added users can be easily found by utilizing the provided search functionality.</p>

<b>Participation in competitions Awards</b>	<p>Bosch – Technik fürs Leben Preis 2018          Weitblick – Champions          Jugend Innovativ – Sonderpreis Sustainability</p> <p>Since the evaluation of the diploma is performed after the conclusion and submission of the project work to competitions no information regarding awards is present.</p>
---	--

<b>Accessibility of diploma thesis</b>	<p>The source code of this diploma is subject to strict secrecy, at the wish of the project team, and is merely accessible by the commission responsible for the grading of this diploma. One version of this diploma will be stored in the archive and is subject to prohibition for public insight.</p>
--	---

<b>Approval (date/signature)</b>	Examiner	Head of College/Department
----------------------------------	----------	----------------------------

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung (Aufgabenstellung des Gesamtprojektes und Überblick über die Funktionen).....</b>	<b>8</b>
1.1	Aufgabenstellung .....	8
1.1.1	Überblick Webinterface.....	8
1.1.2	Überblick Admin-Website .....	8
1.1.3	Überblick Webinterface.....	8
1.1.4	Weitere Aufgaben.....	8
<b>2</b>	<b>Individuelle Zielsetzungen des Projektteams .....</b>	<b>8</b>
2.1	Zielsetzung Niklas Graf.....	8
2.2	Zielsetzung Lukas Knoll .....	8
2.3	Zielsetzung Sebastian Mandl .....	8
<b>3</b>	<b>Umsetzung .....</b>	<b>8</b>
3.1	Umsetzung Web-Application .....	8
3.2	Umsetzung Web-Admin-Application.....	8
3.3	Umsetzung Androidapplication.....	8
3.4	Umsetzung Datenbank.....	8
3.5	Umsetzung Server.....	8
3.6	Weitere Dienste .....	8
3.7.1	aems-apilib .....	8
3.7.2	Diffie-Hellman Library.....	8
<b>4</b>	<b>Verwendete Libraries.....</b>	<b>8</b>
4.1	Web-Applications.....	8
4.2	Android-App.....	8

4.3	Servertechnologien.....	8
<b>5</b>	<b>Technologien.....</b>	<b>8</b>
5.1	Java REST.....	8
5.2	Postgre SQL.....	8
<b>6</b>	<b>Ergebnisse.....</b>	<b>8</b>
6.1	Erreichte Ziele .....	8
6.2	Auswirkung .....	8
6.3	Abschluss .....	8
6.4	Statement Auftraggeber .....	8
<b>7</b>	<b>Danksagung .....</b>	<b>8</b>
<b>8</b>	<b>Quellen-/Literaturverzeichnis.....</b>	<b>8</b>
<b>9</b>	<b>Verzeichnis der Abbildungen, Tabellen und Abkürzungen .....</b>	<b>8</b>
<b>8</b>	<b>Anhang .....</b>	<b>8</b>
8.1	Projektdokumentation (Soll-Ist-Vergleich, Kostendarstellung) .....	8
8.2	Arbeitsbericht Niklas Graf.....	8
8.3	Arbeitsbericht Lukas Knoll.....	8
8.4	Arbeitsbericht Sebastian Mandl.....	8
8.5	Begleitprotokoll.....	8

# **1 Einleitung (Aufgabenstellung des Gesamtprojekts und Überblick über die Funktionen)**

## **1.1 Aufgabenstellung**

Die Aufgabe ist es in öffentlichen Gebäuden einen einfacheren Überblick über die Verbrauchswerte zu liefern. Hierzu sollen die Verbrauchswerte der AMIS-Stromzähler ausgewertet werden. Für eine gute Übersichtlichkeit sollen diese Verbrauchswerte als Statistiken und Berichte aufbereitet werden.

Um den Energieverbrauch besser steuern zu können soll ein Benachrichtigungssystem entwickelt werden, über welches man bei abweichendem Verbrauch im Vergleich zu den Vorperioden benachrichtigt wird.

Eine weitere Aufgabe ist die Erstellung einer mobilen Application für Android Smartphones. Hier soll es die Möglichkeit geben sich vordefinierte Statistiken in der App anzeigen zu lassen und diese herunterladen zu können. Des Weiteren soll man auch am Smartphone bei abweichendem Verbrauch benachrichtigt werden.

Um nicht nur von den AMIS-Zählern abhängig zu sein, soll ein modulares System entwickelt werden, mit welchem man sich auf Raspberry Pis selbst eigene Zähler anlegen und konfigurieren kann. Als Beispiel Strom-, Wärmemengen-, Gaszähler.

Als Erweiterung wird eine Anomalieerkennung umgesetzt. Diese Anomalieerkennung können sich AEMS-Nutzer/innen selbst konfigurieren um sich eigene Abhängigkeitsfaktoren wie z.B. die Außentemperatur einzustellen. Diese Anomalien können dazu verwendet werden das Benachrichtigungssystem intelligent auf Abhängigkeitsfaktoren reagieren zu lassen.

## **1.2 Überblick über die Funktionen**

AEMS bietet die Möglichkeit sich selbst Statistiken über selbst gewählte Zähler und Perioden erstellen zu lassen. Diese Statistiken werden in der Datenbank gespeichert und stehen jederzeit auf der Webpage zur Verfügung.

Es gibt die Möglichkeit zur Erstellung von Berichten. Diese Berichte können durch eigene Wünsche frei konfiguriert werden. Man kann sich die zu verwendenden Statistiken auswählen und die Berichtsperiode. Weiters gibt es die Möglichkeit sich diese Berichte in periodischen Abständen automatisch generieren zu lassen. Zum Beispiel 1 Bericht pro Monat.

Erstellung von Benachrichtigungen und Anomalien. Man kann sich selbst Benachrichtigungen konfigurieren, bei welchen man die zu kontrollierenden Zähler und die höchst zulässige Abweichung angibt. Bei Größerer Abweichung wird man benachrichtigt. Um das Benachrichtigungssystem dynamischer zu gestalten gibt es die Möglichkeit sich selbst Anomalien zu erstellen, welche

## Einführung

für die Benachrichtigungserstellung berücksichtigt werden.

Erstellung von eigenen Zählern auf Raspberry Pis. Auf den Mini-Computern können eigene Zähler für beispielsweise Strom oder Wärmemenge installiert werden. Diese Verbrauchswerte werden auch in die Datenbank eingepflegt und können für Auswertungen und Benachrichtigungen verwendet werden.

Die Androidapp bietet die Möglichkeit die zuvor in der Webpage erstellten Statistiken anzuzeigen. Hier gibt es auch die Möglichkeit die Statistik als Bild am Smartphone zu speichern. Für die bessere Kontrolle der Verbrauchswerte, werden die Benachrichtigungen mit allen relevanten Informationen auch am Smartphone angezeigt.

Im Administrationstool können neue Administratoren und Administratorinnen ernannt und konfiguriert werden. Anhand von Postleitzahlen werden diesen ihre Zuständigkeitsbereiche zugewiesen. Diese Systembetreuer/innen managen die Freigaben für das AEMS-System.

## **2 Individuelle Zielsetzungen des Projektteams**

### **2.1 Niklas Graf**

- Erstellung des BOTs, welcher die Verbrauchswerte der AMIS-Stromzähler von der NetzOnline Website abrufen und in der Datenbank speichert
- Programmierung des Backend für die beiden Webpages „AEMS- Advanced Energy Monitoring System“ und „AEMS - Admin“
- Entwicklung der aems-apilib für die Kommunikation zwischen den Clients und dem Java REST Service
- Grafische Umsetzung der Statistiken und Berichte inkl. Downloadfunktion dieser im Browser

### **2.2 Lukas Knoll**

- Entwicklung der Webpages „AEMS - Advanced Energy Monitoring System“ und „AEMS - Admin“
- Erweiterung und Adaption der Webpages auf XHTML für die Verwendung von JSF
- Programmierung diverser Funktionen für die Webpages mit Javascript und jQuery
- Entwicklung der gesamten Android-Application (Frontend und Backend)
- Konfiguration und Installation des Servers
- Projektmanagement und Controlling

### **2.3 Sebastian Mandl**

- Entwicklung des Datenmodells und der Datenbank
- Entwicklung der eigenen Zähler auf Raspberry Pls
- Erstellung des Java REST Dienstes
- Programmierung der Kommunikations-, und Verschlüsselungsstruktur

## 3 Umsetzung

### 3.1 Umsetzung Web-Application

#### 3.1.1 Frontend

##### 3.1.1.1 Allgemein

Die HTML Seiten entsprechen dem HTML 5 Standard. Für die Stylesheets wurde CSS in der Version 3 verwendet und als Design-Library Bootstrap in der Version 3.3.7. Es wurden 2 Websites programmiert. Die erste Website ist das AEMS-Tool für die Verwaltung des Energieverbrauchs und die zweite Website ist das Administrationstool für die Verwaltung der Zugriffsrechte.

##### 3.1.1.2 Verwendete Tools

Für die Erstellung der HTML-Webpages wurde das Programm Adobe Dreamweaver in der Version CC 2017 verwendet. Damit wurde der Aufbau der HTML-Seiten und die dazugehörigen Stylesheets gemacht. Für die Entwicklung diverser Javascript-Funktionen wurde das Notepad++ in der Version 7.5 verwendet.

##### 3.1.1.3 HTML Pages Struktur

Das Webtool AEMS dient der einfachen Übersicht der Verbrauchswerte, Erstellung von Berichten und Konfiguration von Benachrichtigungen. Diese Funktionalität wurde auf mehrere Seiten aufgeteilt. Der Aufbau dieser Webpages ist auf allen Seiten gleich.

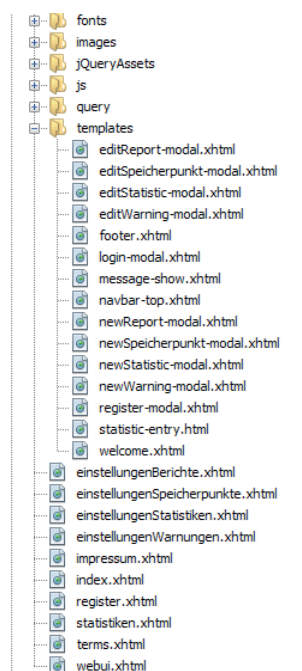


Abb. 1: Übersicht Frontend Dateien



### 3.1.1.4 HTML-Pages Grundgerüst

Die einzelnen Webpages bestehen nur aus dem Grundgerüst der für die Seite notwendigen Elemente. Alle komplexeren Elemente wurden in andere Dateien ausgelagert und werden mit dem Tag `<ui:include>` eingebunden. Dies sorgt für eine gute Übersichtlichkeit und eine einfache Verwendung der Elemente.

```
<head>
  <meta charset="utf-8"/>
  <meta http-equiv="Content-Type" content="text/html"/>
  <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>

  <title>AEMS - Advanced Energy Monitoring System</title>

  <link href="css/bootstrap.css" rel="stylesheet"/>
  <link href="css/main.css" rel="stylesheet" type="text/css"/>
  <link href="css/index.css" rel="stylesheet" type="text/css"/>
  <link href="css/login-modal.css" rel="stylesheet" type="text/css"/>
</head>

<body style="padding-top: 70px">
  <f:event type="preRenderView" listener="#{notificationBean.init}" />
  <ui:include src="/templates/navbar-top.xhtml"></ui:include>

  <c:if test="#{!user.loggedIn}">
    <ui:include src="/templates/welcome.xhtml"></ui:include>
  </c:if>
  <!-- Seiteninhalt -->
  <c:if test="#{user.loggedIn}">
    <article class="statistics">
      <ul class="listStyle">
        <c:forEach items="#{statisticDisplayBean.statistics}" var="statistic">
          <li class="statistikElement">
            <div class="statistic-container">
              <h2 id="h_#{statistic.name}" class="stat-title">#{statistic.name}</h2>
              <canvas id="#{statistic.id}" width="600" height="400"></canvas>
              <button class="btn btn-default download-pdf"><span class="glyphicon glyphicon-download-alt"></span> PDF Herunterladen</button>
            </div>
          </li>
        </c:forEach>
      </ul>
    </article>
  </c:if>
```

Abb. 2: Startseite AEMS

Die Grundstruktur der Seiten beinhaltet den HTML-Aufbau, die Imports der einzelnen Elemente und die Verweise auf die zu verwendenden Libraries, CSS-, und Javascript-Dateien.

Durch die Verwendung von `<ui:include>` werden die Elemente mit dem Tag `<ui:composition>` an die entsprechende Stelle geladen.

```
<ui:composition
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:f="http://xmlns.jcp.org/jsf/core"
  xmlns:h="http://xmlns.jcp.org/jsf/html"
  xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
  xmlns:c="http://xmlns.jcp.org/jsf/core"
  xmlns:pt="http://xmlns.jcp.org/jsf/passthrough">

  <c:if test="#{!register.isRegistering()}">
    <div class="welcome">
      <h1>Willkommen bei AEMS</h1>
      <p>
        Wir freuen uns, dass Sie unser neues System verwenden wollen.<br />
        Mit AEMS - Advanced Energy Monitoring System ist es Ihnen möglich, Ihre Verbrauchswerte grafisch aufbereiten und auswerten zu lassen.<br />
        Auch die Möglichkeit sich Benachrichtigungen und Warnungen bei abweichenden Verbrauchswerten anzeigen zu lassen ist durch dieses System möglich.<br />
        Für fortgeschrittene Benutzer gibt es auch noch die Möglichkeit sich selbst Zähler auf Raspberry-Pi's einzurichten und diese in das AEMS-System einzubinden.<br />
        Besuchen Sie uns auch im Android-App Store. Dort gibt es unter dem Titel "AEMS" auch noch eine App für Ihr Smartphone oder Tablet.
      </p>
    </div>
    <a href="#" data-toggle="modal" data-target="#login-modal" data-backdrop="false" data-dismiss="modal" class="btn btn-default">Login</a>
    <a href="#" data-toggle="modal" data-target="#register-modal" data-backdrop="false" data-dismiss="modal" class="btn btn-default">Registrieren</a>
  </c:if>
</ui:composition>
```

Abb. 3: UI-Composition

Diese UI-Compositions fungieren als ein Set von Elementen. Inhalte, wie der Seiteninhalt, oder ganze Formulare wurden in solche UI-Compositions ausgelagert.

### 3.1.1.5 Design

Für das Design einzelner Elemente der Webpages wurde die Designlibrary Bootstrap verwendet.

<https://getbootstrap.com/docs/3.3/getting-started/> [22.07.2017]

Weiters wurden eigene Designs und Designerweiterungen entworfen. Die Webpages wurden somit dynamisch gestaltet, um eine gute Übersichtlichkeit auf allen Geräten (PC, Notebook, Tablet und Smartphone) gewährleisten zu können.

```
.statistics .listStyle {
    list-style-type: none;
    margin-left: 3%;
}

@media screen and (min-width:760px) and (max-width:1200px)
{
    .statistics {
        margin-top: 4em;
        margin-left: 3%;
    }
    .statistic-container {
        width: 650px;
        margin-bottom: 5%;
        display: inline-block;
    }

    .statistics .listStyle {
        list-style-type: none;
        margin-left: 3%;
    }
}

.actionBox {
    border-radius: 5px;
    background-color: orange;
    color: black;
    padding: 15px;
    text-align: center;
    font-size: 2em;
    width: 100px;
    text-decoration: none;
}
```

Abb. 4: CSS

Für eine gute Verwendbarkeit auf allen Gerätegrößen, wurde darauf geachtet den Elementen und Abständen zwischen den Elementen, keine festen Größen zu geben, sondern diese mit Verwendung von prozentuellen Verhältnissen besser zu strukturieren.

Um das Design auf den verschiedenen Bildschirmgrößen weiter zu verbessern, wurden die Eigenschaften @media (min-width) and (max-width) verwendet. Damit ist es möglich das selbe Elemente für verschiedene Displaygrößen anzupassen.

Der CSS-Code für das Design wurde nur in die dafür vorgesehenen CSS-Dateien und nicht in die HTML-Pages geschrieben.

Als Beispiel für ein Element, für welches Bootstrap in Kombination mit eigenen CSS Styles verwendet wurde dient die Navigation-Bar.

```
<nav class="navbar navbar-default navbar-inverse navbar-fixed-top navbarMainFormat">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed"
        data-toggle="collapse" data-target="#topFixedNavbar1"
        aria-expanded="false"><span class="sr-only">
        Toggle navigation</span>
      <span class="icon-bar"></span><span class="icon-bar">
      </span><span class="icon-bar"></span></button>
      <h:link outcome="index" class="navbar-brand titleFormat">
      
    </h:link>
    </div>
    <!-- Collect the nav links, forms, and other content for toggling -->
    <c:if test="#{user.loggedIn}">
    <div class="collapse navbar-collapse" id="topFixedNavbar1">
      <ul class="nav navbar-nav navbar-right">
        <!-- Menü Statistiken -->
        <li class="navBarItems"><h:link outcome="statistiken">Statistik</h:link></li>
        <!-- Menü Einstellungen -->
        <li class="dropdown"><a href="#" class="dropdown-toggle navBarItems"
          data-toggle="dropdown" role="button"
          aria-haspopup="true" aria-expanded="false">
          Einstellungen<span class="caret"></span></a>
          <ul class="dropdown-menu">
            <li><h:link outcome="einstellungenSpeicherpunkte">Speicherpunkte</h:link></li>
            <li role="separator" class="divider"></li>
          </ul>
        </li>
      </ul>
    </div>
    </c:if>
  </div>
</nav>
```

Abb. 5: Navigation-Bar

Hier ist ein Ausschnitt aus der Navigation-Bar zu sehen.

Für das Grunddesign wurde hier Bootstrap verwendet. Für eine andere Anordnung, andere Größe der Elemente und das Verhalten auf verschiedenen Displaygrößen, wurden eigene Styles entworfen. Dies gilt für den Grundaufbau der Navigation-Bar, wie auch für die darin enthaltenen Elemente.

### 3.1.1.6 Modals

Um eine gute Übersichtlichkeit auf den Websites zu gewährleisten, wurden die Eingabefelder nicht in den Websites selbst integriert, sondern als Dialoge (Modals) ausgelagert. Diese Modals öffnen sich beim Klick auf die Buttons für die Erstellung oder Bearbeitung von Elementen, wie Statistiken, Berichten, oder Benachrichtigungen.

```
<button id="createStatistic" class="btn btn-default ownButton"
  data-toggle="modal" data-target="#newStatistic-modal"
  data-backdrop="false" data-dismiss="modal">
  Neue Statistik anlegen</button>
```

Abb. 6: Modal aufruf

## Einführung

Der Aufruf eines Modals funktioniert wie folgt:

Man muss einen Data-Toggle angeben. Das Element, welches den Data-Toggle bekommt, wird somit fokussiert. Durch die Eingabe von „modal“ bekommt somit das Modal welches aufgerufen wird den Fokus.

Unter Data-Target muss man anschließend das Element angeben, welches Aufgerufen werden soll. Also die Id des Modals.

Mit Data-Backdrop gibt man an, ob der Hintergrund auf Grau gesetzt werden soll, oder nicht. Diese Auswahl führt zu einer besseren Übersicht, ob das Modal fokussiert ist, oder nicht.

Mit dem Befehl Data-Dismiss fügt man dem Modal einen Button mit einem „X“ in der Mitte hinzu, um das Modal zu schließen.

```
<div class="modal fade" id="newStatistic-modal"
  role="dialog" aria-labelledby="myModallLabel"
  aria-hidden="true" style="display: none;">
  <div class="modal-dialog">
    <div class="loginmodal-container">

      <h1>Neue Statistik erstellen</h1><br />

      <h:form id="newStatistic">
        <h4>Statistikname</h4>
```

Abb. 7: Modal

Hier ist nun das Modal selbst zu sehen. Dieses besitzt die Id, welche beim Modalaufruf als Data-Target angegeben wurde.

Durch den Befehl role= "dialog" wird das Modal in einem neuen Fenster (Dialog) geöffnet und nicht in die Website selbst eingebettet.

Falls das Modal nicht aufgerufen werden kann, wird stattdessen der Text, welcher in aria-labelledby steht angezeigt.

Der Befehl aria-hidden sorgt dafür, dass der Inhalt nicht von einem Screen-Reader vorgelesen wird.

Für das Design der Modals wurde eine Library aus dem Internet verwendet. Weitere Designanpassungen wurden aber auch selbst vorgenommen.

<https://bootsnipp.com/snippets/featured/clean-modal-login-form> [25.07.2017]

### 3.1.1.7 Startseite

Zu Beginn werden auf der Startseite nur ein Text über die Funktionen des AEMS-Systems und die Funktionen für Login und Registrieren bereitgestellt.

Der Seiteninhalt der Startseite wird erst nach der Anmeldung oder Registrierung angezeigt. Diese Funktionalität wird wieder mit ui:include und den entsprechenden ui:compositions gelöst.

```
<c:if test="#{!user.loggedIn}">
  <ui:include src="/templates/welcome.xhtml"></ui:include>
</c:if>
<!-- Seiteninhalt -->
<c:if test="#{user.loggedIn}">
  <article class="statistics">
    <ul class="listStyle">
      <c:forEach items="#{statisticDisplayBean.statistics}" var="statistic">
        <li class="statistikElement">
          <div class="statistic-container">
            <h2 id="h_#{statistic.name}" class="stat-title">#{statistic.name}</h2>
            <canvas id="#{statistic.id}" width="600" height="400"></canvas>
            <button class="btn btn-default download-pdf">
              <span class="glyphicon glyphicon-download-alt"></span>
              PDF Herunterladen</button>
          </div>
        </li>
      </c:forEach>
    </ul>
  </article>
</c:if>
```

Abb. 8: Anzeige Startseite

Hier ist zu sehen, dass es eine Abfrage gibt, ob der User angemeldet ist. Wenn nicht, wird die Startseite mit den Infos über das AEMS-System angezeigt über die Einbindung des welcome-modals. Wenn der Nutzer angemeldet ist, erscheinen direkt seine für die Startseite konfigurierten Statistiken.

### 3.1.1.8 Einstellungen - Funktionen

Für die Erstellung und Bearbeitung von Statistiken, Berichten und Benachrichtigungen gibt es jeweils entsprechende Seiten. Für eine einfache Verständlichkeit der Funktionen (Bearbeiten, Löschen, Zur App hinzufügen, Zur Startseite hinzufügen) - Funktionen abhängig von den verwendeten Tools - werden Icons mit Tooltips verwendet.

```

<h:form>
  <div id="allList" class="list-group statisticsList">
    <c:forEach items="#{statisticBean.statistics}" var="item">
      <div class="list-group-item">
        <h4 class="list-group-item-heading">#{item.name}</h4>
        <p class="list-group-item-text">#{item.annotation}</p>
        <div class="btn-glyph">
          <div class="edit-statistic btn btn-default"
            data-id="#{item.id}" title="Statistik bearbeiten"
            data-toggle="modal" data-target="#editStatistic-modal"
            data-backdrop="false" data-dismiss="modal">
            <span class="glyphicon glyphicon-pencil"></span></div>
          <h:commandLink action="#{statisticActionBean.addToAndroid(item.id)}">
            <button class="btn btn-default" title="Statistik zu Android-App hinzufügen">
              <span class="glyphicon glyphicon-phone"></span></button>
          </h:commandLink>
          <h:commandLink action="#{statisticActionBean.addToIndex(item.id)}">
            <button class="btn btn-default" title="Statistik zu Startseite hinzufügen">
              <span class="glyphicon glyphicon-home"></span></button>
          </h:commandLink>
          <h:commandLink action="#{statisticActionBean.remove(item.id)}">
            <button class="btn btn-default" title="Statistik verwerfen">
              <span class="glyphicon glyphicon-trash"></span></button>
          </h:commandLink>
        </div>
      </div>
    </c:forEach>
  </div>

```

Abb. 9: Funktionen Webpages

Für die Funktionen werden Buttons verwendet, mit den entsprechenden Icons. Je nachdem, welche Funktion aufgerufen wird, öffnet sich ein Modal zur Bearbeitung des Eintrags, oder es wird direkt eine Aktion ausgeführt (zum Beispiel Löschen).

Für die Icons wird die Library Bootstrap Glyphicon verwendet.

[https://www.w3schools.com/bootstrap/bootstrap\\_ref\\_comp\\_glyphs.asp](https://www.w3schools.com/bootstrap/bootstrap_ref_comp_glyphs.asp) [11.12.2017]

### 3.1.1.9 Mobile Functions

Für eine leichtere Verwendung auf Mobilien Geräten, wurden einige Elemente speziell für mobile Browser angepasst. Als Beispiel die Fußzeile.

```

$(document).load($(window).bind("resize", startFormat));

function startFormat() {
  footerFormat();
  navbarFormat();
}

function footerFormat() {
  if (/Mobi/.test(navigator.userAgent)) {
    $('footer').removeClass("footer");
    $('footer').removeClass("navbar-fixed-bottom");
  }
}

```

Abb. 10: Mobile Function

In diesem Javascript-Code wird überprüft, ob die Website von einem Mobilien Browser aufgerufen wird. Wenn der Browser eine mobile Version ist, wird die Fußzeile nicht mehr am unteren Rand des Displays fixiert. Das sorgt dafür, dass am Smartphone mehr inhaltsrelevante Daten angezeigt werden können und der Platz nicht durch Standardelemente verschwendet wird.

#### **3.1.1.11 Notifications**

In der Navigation-Bar wird man über Funktionen des AEMS-Systems benachrichtigt. Zum Beispiel bei einer Warnung über zu hohen Verbrauch, oder wenn ein neuer Bericht zum Download zur Verfügung steht.

Für die Erstellung von Benachrichtigungen wird die Library Noty.js verwendet. Diese ist für das Frontend der Notifications, wie auch das Backend zuständig. Funktionalität und aussehen der Notifications wurden mit eigenen Funktionen angepasst.

<https://ned.im/noty/#/> [05.01.2018]

#### **3.1.1.10 Vergleich AEMS und AEMS-Admin**

Für die designtechnische Umsetzung der beiden Webseiten AEMS und AEMS-Admin wurden die selben Technologien verwendet. Der Unterschied der beiden Tools liegt nur in ihrer Funktionalität, nicht aber in ihren Designs.

### 3.1.2 Schnittstelle XHTML

Um für die Webpages eine Funktionalität zu erhalten, wird mit Java Server Faces, kurz JSF gearbeitet. Die Voraussetzung für die Verwendung von JSF ist XHTML. XHTML ist eine Abwandlung von HTML-Code.

Im eigentlichen Sinne ist XHTML, HTML-Code im XML-Format.

Während die Richtlinien bei HTML nicht so streng sind, wird der XHTML-Code strikter auf Korrektheit überprüft. Während bei HTML Tags nicht zwingend geschlossen werden müssen, ist dies bei XHTML zwingend erforderlich. Auch müssen Attributswerte zwingend unter Hochkommas gestellt werden.

Im Grunde kann HTML-Code leicht in XHTML-Code umgewandelt werden. Jedoch ist darauf zu achten, dass nicht alle Tag-Attribute, welche es in HTML gibt, auch in XHTML vorhanden sind.

Im Kapitel 3.1 gezeigter Code ist bereits XHTML-Code. Dort wurde bereits mit Java Beans gearbeitet, welche im nächsten Kapitel erklärt werden.

Würde HTML-, anstatt XHTML-Code für die Programmierung verwendet werden, wäre der Code für den Interpreter unleserlich.

[https://www.w3schools.com/html/html\\_xhtml.asp](https://www.w3schools.com/html/html_xhtml.asp) [23.09.2017]



### **3.1.3 Backend**

## 3.2 Umsetzung Android-App

### 3.2.1 Allgemein

Die Android-App ist dafür gemacht, die Nutzerschaft bei abweichendem Energieverbrauch darüber zu benachrichtigen. Weiters werden die zuvor bereits in der Website angelegten und für die App freigegebenen Statistiken angezeigt. Diese können auch direkt heruntergeladen werden. Entwickelt wurde die App im Androidstudio mit der Version 2.3.3. Die Kompatibilität der App wird in den Androidversionen 4.4 Kitkat bis 8.0 Oreo gewährleistet.

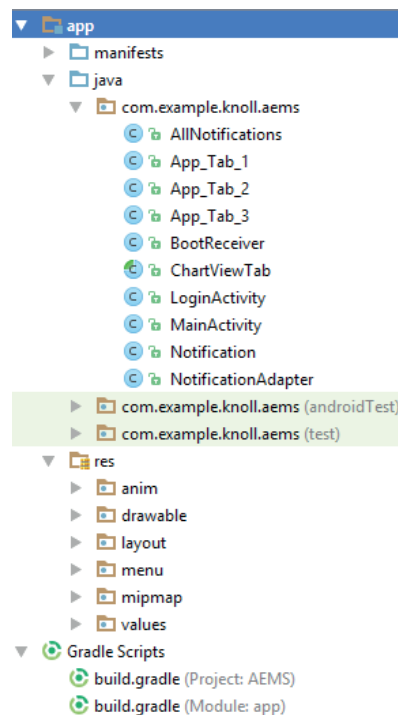


Abb. 11: Android-App Struktur

Der Aufbau der App-Struktur ist wie im obigen Bild zu sehen.

Im Manifest müssen alle verwendeten Activities und Permissions eingetragen werden. Permissions dienen dazu, der App nur die Rechte, welche sie auch wirklich braucht zu geben. Als Beispiel das Recht Daten auf dem Telefonspeicher abzulegen.

Im Ordner „Java“ sind alle Klassen und Activities zu finden. Hier wird die eigentliche Logik der App programmiert.

Im Verzeichnis „res“ befindet sich der grafische Teil der App. Unter Layouts werden die Designs der einzelnen Seiten und Fragments (Teilelemente) definiert.

Unter „menu“ werden Design und Inhalt der Menüs programmiert und unter values können weitere Dinge wie Farben, oder Styles definiert werden.

Gradle ist ein Build-System. Hier müssen alle Libraries, welche verwendet werden, eingetragen werden.

### 3.2.2 Main Activity

Die Main-Activity einer Android-App ist die erste Klasse welche beim Ausführen der App gestartet wird. Darin wird die Hauptfunktionalität des Programms definiert. Der Aufbau der App, die Menüstruktur, wie auch der Aufruf weiterer Activities wird hier gemacht.

#### 3.2.2.1 Tabbed Activities

Für die Anzeige der Statistiken wird die Technologie der Tabbed Activities verwendet. Diese Technologie ermöglicht es in waagrechte Richtung zwischen den Statistiken hin und her zu wischen.

Hierfür müssen die einzelnen Statistik Tabs als eigene Activity-Klassen definiert und in der Main-Activity aufgerufen werden. Das heißt es müssen Instanzen der „ChartViewTabs“, also der einzelnen Statistik-Activities in der Main-Activity gemacht werden.

Weiters müssen ein „SectionPagerAdapter“ und ein „ViewPager“ angelegt werden. Diese werden für den Vorgang des Wechsels der Statistiken benötigt. Diese Tabs müssen anschließend auch Ihren Activity Klassen zugewiesen werden.

```
private SectionsPagerAdapter mSectionsPagerAdapter;  
private ViewPager mViewPager;  
  
private ChartViewTab tab1;  
private ChartViewTab tab2;  
private ChartViewTab tab3;  
  
tab1 = new App_Tab_1();  
tab2 = new App_Tab_2();  
tab3 = new App_Tab_3();
```

Abb. 12: Anlegen der Statistik Tabs

## Einführung

Als nächstes muss der App gesagt werden, welche Statistik bei einem Wischen als nächstes angezeigt werden soll und welche die aktuelle Statistik ist. Dies funktioniert mit einem sogenannten `SectionPagerAdapter`. Dieser wählt das benötigte Fragment (in unserem Fall die benötigte Statistik) und zeigt dieses an.

Im nächsten Schritt merkt sich der `ViewPager` das gerade aktuelle Fragment (aktuelle Statistik).

In `getCount()` wird festgelegt, wie viele verschiedene Tabs es geben darf.

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {

    public SectionsPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {

        switch (position){
            case 0:
                return tab1;
            case 1:
                return tab2;
            case 2:
                return tab3;
            default:
                return null;
        }
    }

    public ChartViewTab getCurrentTab() {
        return (ChartViewTab) getItem(mViewPager.getCurrentItem());
    }

    @Override
    public int getCount() {
        // Show 3 total pages.
        return 3;
    }
}
```

Abb. 13: SectionPagerAdapter

### 3.2.2.2 Download von Statistiken

Um die in den Tabs angelegten Statistiken herunterladen zu können, wird die zurzeit angezeigte Statistik benötigt. Diese haben wir uns wie unter 3.2.2.1 beschrieben in einem `ViewPager` gespeichert. Über diesen `ViewPager` wissen wir jetzt, welche Statistik heruntergeladen werden soll.

```
mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());

// Set up the ViewPager with the sections adapter.
mViewPager = (ViewPager) findViewById(R.id.container);
mViewPager.setAdapter(mSectionsPagerAdapter);

FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        ChartViewTab currentTab = mSectionsPagerAdapter.getCurrentTab();
        Snackbar.make(view, currentTab.getStatisticTitle() + " wird heruntergeladen!", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();

        switch (mViewPager.getCurrentItem()) {
            case 0:
                downloadStatistic(tab1);
                break;
            case 1:
                downloadStatistic(tab2);
                break;
            case 2:
                downloadStatistic(tab3);
                break;
        }
    }
});
```

Abb. 14: Selektion der Statistik für den Download

Zuerst wird ein `SectionPagerAdapter` angelegt. Dieser Adapter muss anschließend dem Pager zugewiesen werden.

Anschließend wird ein `FloatingActionButton` angelegt. Beim Klick auf diesen Button soll die entsprechende Statistik heruntergeladen werden. Wenn man auf diesen Button klickt, wird dessen `onClick` Methode aufgerufen.

In der `onClick`-Methode wird zuerst der gerade ausgewählte Tab bestimmt. Anschließend wird mit dem Befehl „`Snackbar.make()`“ ein Balken am unteren Displayrand erzeugt, welcher anzeigt das eine bestimmte Statistik heruntergeladen wird.

Im Anschluss wird mit einem „switch“ festgelegt, welche Statistik heruntergeladen werden soll. Hier wird dann die `downloadStatistic`-Methode mit dem entsprechenden Statistik-Parameter aufgerufen.

```
private void downloadStatistic(ChartViewTab tab) {
    boolean saveToSd = false;
    boolean canSaveImage = false;
    Chart chart = tab.getChart();
    Bitmap image = chart.getChartBitmap();
    String filename = tab.getStatisticTitle();
    File filePath = null;

    if(saveToSd) {
        ActivityCompat.requestPermissions(this, new String[] {android.Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
        filePath = new File(Environment.getExternalStorageDirectory(), "/AEMS/");
        filePath.mkdirs();

        File imageFile = new File(filePath, filename + ".jpg");
        OutputStream stream = null;
        try {
            stream = new FileOutputStream(imageFile);
            image.compress(Bitmap.CompressFormat.JPEG, 100, stream);
            stream.flush();
            stream.close();
        } catch(IOException e) {
            e.printStackTrace();
        }
    } else {
        if (hasStoragePermission()) {
            String url = MediaStore.Images.Media.insertImage(getContentResolver(), image, filename, "Hello");
            if(url != null){
                canSaveImage = true;
            }
            else if(url == null) {
                canSaveImage = false;
            }
        } else {
            ActivityCompat.requestPermissions(this,
                new String[] {Manifest.permission.WRITE_EXTERNAL_STORAGE},
                12);
        }
    }

    doGenerateNotification(canSaveImage, filename);
}
```

Abb. 15: Statistik Download

In der Methode „downloadStatistic“ wird nun die Statistik heruntergeladen. Zuerst muss der App die Berechtigung gegeben werden Daten auf den Smartphonespeicher zu speichern. Hierfür muss eine Permission für „WRITE\_EXTERNAL\_STORAGE“ gesetzt werden. Anschließend muss noch ein Pfad bestimmt werden, unterm welchem die Statistiken gespeichert werden sollen.

dem imageFile werden der Pfad unter welchem das Bild gespeichert werden soll, der Dateiname und die Dateierdung mitübergeben.

Nun wird über einen FileStream die Statistik zu einem JPEG-Bild konvertiert.

Wenn der Nutzer der App die Berechtigung zum Speichern von Dateien auf den Smartphonespeicher erteilt hat, wird die Statistik unter dem angegebenen Pfad gespeichert.

Im Anschluss wird die Methode doGenerateNotification aufgerufen. Diese bekommt die Attribute canSaveImage und den Dateinamen mit. Hier wird im Anschluss an das Speichern der Statistik eine Benachrichtigung erzeugt, ob das Speichern der Statistik erfolgreich verlaufen ist.

<https://stackoverflow.com/questions/649154/save-bitmap-to-location> [01.09.2017]

### 3.2.3 Statistik Tab Activity

#### 3.2.3.1 Allgemein

In den Statistic Tab Activities werden die Statistiken erstellt, welche in der Main Activity aufgerufen und in die entsprechenden Tabs eingefügt werden.

Hier gibt es zwei Arten von Statistiken. Die einfachen Charts, welche nur aus einer Statistik bestehen und die CombinedCharts, bei welchen mehrere Statistiken in einer Grafik kombiniert werden können.

Als Library wird MPAndroidChart verwendet.

<https://github.com/PhilJay/MPAndroidChart> [24.08.2017]

#### 3.2.3.2 Einfache Bar Charts

Einfache Charts bestehen aus nur einem Set von Daten, wie zum Beispiel dem Stromverbrauch.

```
public class App_Tab_1 extends ChartViewTab {

    public App_Tab_1() {
        super(R.layout.app_tab_1, R.id.chart1);
    }

    @Override
    public void onCreateChart(Chart chart) {
        List<BarEntry> entries = new ArrayList<BarEntry>();
        entries.add(new BarEntry(1f, 2));
        entries.add(new BarEntry(2f, 1));
        entries.add(new BarEntry(4f, 15));
        entries.add(new BarEntry(5f, 13));
        entries.add(new BarEntry(7f, 4));
        entries.add(new BarEntry(8f, 5));
        entries.add(new BarEntry(10f, 7));
        entries.add(new BarEntry(11f, 8));

        BarDataSet dataSet = new BarDataSet(entries, "Stromverbrauch");
        dataSet.setColor(Color.RED);
        dataSet.setValueTextColor(Color.BLUE);

        BarData barData = new BarData(dataSet);
        chart.setData(barData);
        chart.invalidate();
    }

    @Override
    public String getStatisticTitle() {
        return "Strom1";
    }
}
```

Abb. 16: Einfache Bar Charts

Bei einfachen Bar Charts müssen in einer Liste die Werte aller Balken angegeben werden. Hierfür wird der Liste ein neues Bar-Element hinzugefügt, welchem dann die Werte für die X-, und Y-Koordinaten gegeben werden.

Anschließend wird aus den Entries ein Data-Set mit einem Namen erzeugt. Es können auch noch weitere Eigenschaften, wie die Farbe der Bars vergeben werden. Mit `chart.setData()` wird der Inhalt für die Statistik gesetzt und mit `chart.invalidate()` wird die Statistik neu geladen.

### 3.2.3.3 Combined Charts

Für die Verwendung von mehreren Statistiken in einer Grafik ist es notwendig sogenannte Data-Sets zu erstellen. Mehrere Sets können zu einer Grafik zusammengestöpselt werden.

```
private BarData generateBarData() {

    ArrayList<BarEntry> entries1 = new ArrayList<BarEntry>();
    ArrayList<BarEntry> entries2 = new ArrayList<BarEntry>();

    entries1.add(new BarEntry(1f, 2));
    entries1.add(new BarEntry(1f, 6));
    entries1.add(new BarEntry(1f, 4));

    entries2.add(new BarEntry(1f, 2));
    entries2.add(new BarEntry(1f, 6));
    entries2.add(new BarEntry(1f, 5));

    BarDataSet set1 = new BarDataSet(entries1, "Aktuelle Werte");
    set1.setColor(Color.rgb(60, 220, 78));
    set1.setValueTextColor(Color.rgb(60, 220, 78));
    set1.setValueTextSize(8f);
    set1.setAxisDependency(YAxis.AxisDependency.LEFT);

    BarDataSet set2 = new BarDataSet(entries2, "Vorperiode");
    set2.setColor(Color.rgb(61, 165, 255));
    set2.setValueTextColor(Color.rgb(61, 165, 255));
    set2.setValueTextSize(8f);

    float groupSpace = 0.5f; // Space between Bar Groups
    float barSpace = 0.1f;
    float barWidth = 0.7f;

    BarData barD = new BarData(set1, set2);
    barD.setBarWidth(barWidth);

    // Makes BarData object grouped
    barD.groupBars(0, groupSpace, barSpace); // Start at x = 0
}
```

Abb. 17: Combined Chart - Bar Set

Für Charts vom selben Typ muss immer ein Set erstellt werden. Wie im obigen Code zu sehen, gibt es zwei Bar-Charts, welche zusammengefügt werden.

Es ist für jeden Chart eine Liste zu erstellen, welche mit den entsprechenden Daten befüllt werden muss.

Anschließend werden die Einstellungen für ein DataSet vorgenommen (Farbe der Balken, Ausrichtung der Legende, ...).

Zum Schluss muss ein BarData Object gemacht werden, welchem die beiden DataSets zugewiesen werden.

Mit dem Befehl `barD.groupBars` werden die Balken nach einem definierten Schema gruppiert.



```

private LineData generateLineData() {

    LineData lineD = new LineData();

    ArrayList<Entry> entries = new ArrayList<Entry>();

    entries.add(new Entry(1f, 21));
    entries.add(new Entry(3f, 23));
    entries.add(new Entry(5f, 24));

    LineDataSet dataSet = new LineDataSet(entries, "Außentemperatur");
    dataSet.setColor(Color.rgb(240, 140, 70));
    dataSet.setLineWidth(2.5f);
    dataSet.setCircleColor(Color.rgb(240, 238, 70));
    dataSet.setCircleRadius(5f);
    dataSet.setFillColor(Color.BLUE);
    dataSet.setMode(LineDataSet.Mode.CUBIC_BEZIER);
    dataSet.setDrawValues(false);
    dataSet.setValueTextSize(10f);
    dataSet.setValueTextColor(Color.BLACK);
    dataSet.setAxisDependency(YAxis.AxisDependency.RIGHT);

    lineD.addDataSet(dataSet);
}

```

Abb. 18: Combined Chart - Line Graph

Um in eine Grafik nicht nur den Energieverbrauch eines Zählertyps und dessen Werte der Vorperiode als Balkendiagramm einfließen zu lassen, kann auch noch ein Liniendiagramm für Anomalien miteingebunden werden.

Das System ist hier wieder das Selbe. Es muss eine Liste für alle Werte angelegt werden. Diese Liste wird dann einem LineDataSet Objekt zugewiesen. Im Anschluss können wieder Einstellungen für den Diagramm-Typ gewählt werden. Zum Beispiel die Strichdicke, Farbe, Legende und ob die Werte direkt im Diagramm dabeistehen sollen.

```

CombinedData data = new CombinedData();

data.setData(generateBarData());
data.setData(generateLineData());

xAxis.setAxisMaximum(data.getXMax() + 0.2f);

chart.setData(data);
chart.invalidate();

```

Abb. 19: Combined Data

Im Hauptprogramm werden beide Methoden aufgerufen. Die des Balkendiagramms und die des Liniendiagramms. Diese verschiedenen Chart-Typen werden dann einem CombinedData-Objekt mitgegeben. Anschließend müssen die Daten dieses CombinedData-Objekts noch dem Diagramm selbst mitgegeben werden.

Mit dem Befehl `chart.invalidate()` wird die Statistik neu geladen.

### 3.2.4 Login Activity

Für das Login wird ein von Google vorgefertigtes Template verwendet, welches jedoch sehr stark angepasst wurde. Beim Start der App wird man auf die Loginseite gelotzt. Auf dieser kann man sich mit den selben Zugangsdaten, wie auf der Website anmelden.

Die Zugangsdaten werden über SharedPreferences gespeichert, damit man sich bei der Verwen-

```
public boolean validate() {
    boolean valid = true;

    String user = _inputUsername.getText().toString();
    String password = _passwordText.getText().toString();

    if (user.isEmpty()) {
        _inputUsername.setError("Geben Sie einen Benutzernamen ein");
        valid = false;
    } else {
        _inputUsername.setError(null);
    }

    if (password.isEmpty()) {
        _passwordText.setError("Geben Sie ein Passwort ein");
        valid = false;
    } else {
        _passwordText.setError(null);
    }

    return valid;
}
```

Abb. 20: Login Validate

Damit man überhaupt einen Anmeldeversuch machen kann, wird überprüft, ob der Benutzer/ die Benutzerin überhaupt einen Benutzernamen und ein Passwort eingegeben hat. Wenn nicht, wird gar keine Login-Anfrage an den Server geschickt und man bekommt den Hinweis, Nutzerdaten einzugeben.

Da der Benutzername keinem festen Schema (z.B. E-Mail) entspricht ist eine Validierung auf den Inhalt der Textfelder nicht notwendig.

```
public void onLoginFailed() {
    Toast.makeText(getBaseContext(), "Login fehlgeschlagen - Überprüfen Sie Ihre Logindaten", Toast.LENGTH_LONG).show();
    _loginButton.setEnabled(true);
}
```

Abb. 21: Login-Failed

Bei der Eingabe falscher Nutzerdaten wird man darauf aufmerksam gemacht, dass die Logindaten falsch eingegeben wurden.

```
public void login() {
    Log.d(TAG, "Login");

    if (!validate()) {
        onLoginFailed();
        return;
    }

    _loginButton.setEnabled(false);

    final ProgressDialog progressDialog = new ProgressDialog(LoginActivity.this,
        R.style.Theme_AppCompat_Light_Dialog);
    progressDialog.setIndeterminate(true);
    progressDialog.setMessage("Authentifizierung...");
    progressDialog.show();

    final String email = _inputUsername.getText().toString();
    final String password = _passwordText.getText().toString();

    //Login to AEMS
    BigDecimal key = null;
    try {
        DiffieHellmanProcedure.sendKeyInfos(new Socket(InetAddress.getByName("localhost"), 9950));
        key = KeyUtils.salt(new BigDecimal(new String(DiffieHellmanProcedure.confirmKey())), email, password);
    } catch (Exception e) {
        e.printStackTrace();
    }

    BigInteger keyInt = key.unscaledValue();
    byte[] sharedSecretKey = keyInt.toByteArray();
}
```

Abb. 22: Login in AEMS

Beim Loginversuch wird zuerst die Methode `validate()` aufgerufen, welche überprüft, ob Nutzerdaten eingegeben wurden.

Anschließend erscheint am Display ein Progressdialog, welcher darüber informiert, dass gerade ein Anmeldeversuch gemacht wird.

Um die Zugangsdaten zu verschlüsseln, wird die selbst entwickelte DiffieHellman-Library verwendet. Hier werden der Nutzernamen, das Passwort und ein zufällig generierter Salt dazu verwendet, die Sicherheit zu garantieren.

Anschließend wird der Key, welchen man erhält auf ein `byte[]` gecastet, um ihn für das Login verwenden zu können.

```

AemsAPI.setUrl("https://api.aems.at");

AemsLoginAction loginAction = new AemsLoginAction(EncryptionType.AES);
loginAction.setUsername(email);
loginAction.setPassword(password);

AemsResponse response = null;
try {
    response = AemsAPI.call0(loginAction, sharedSecretKey);
    httpCode = response.getResponseCode();
    responseText = response.getResponseText();
    responseDecryptedText = response.getDecryptedResponse();
} catch (IOException e) {
    e.printStackTrace();
}

new android.os.Handler().postDelayed(
    new Runnable() {
        public void run() {
            // On complete call either onLoginSuccess or onLoginFailed
            if(httpCode == 200){
                onLoginSuccess(email, password);
            }
            else{
                onLoginFailed();
            }
            progressDialog.dismiss();
        }
    }, 500);

```

Abb. 23: Loginprozess

Mit dem oben erhaltenen Key wird man dann beim AEMS-Dienst angemeldet. Diese Kommunikation geschieht über die eigens entwickelte aems-apilib.

Wenn als responseCode 200 zurück kommt, war das Login erfolgreich und die Methode onLoginSuccess wird aufgerufen. Wenn der Responsecode nicht 200 ist, kommt man über die Methode onLoginFailed() zur Loginseite zurück und wird benachrichtigt, dass das Login nicht erfolgreich war.

```

public void onLoginSuccess(String email, String password) {
    _loginButton.setEnabled(true);

    //Save Logindata in SharedPreference
    sharedPreferences = getSharedPreferences(PREFERENCE_KEY, MODE_PRIVATE);
    String user = sharedPreferences.getString("EMAIL", null);
    String passw = sharedPreferences.getString("PASSWORD", null);

    if(user == null && passw == null){
        CheckBox checkBoxRememberMe = (CheckBox) findViewById(R.id.checkBoxRememberLogin);

        if(checkBoxRememberMe.isChecked()){
            sharedPreferences = getSharedPreferences(PREFERENCE_KEY, MODE_PRIVATE);
            sharedPreferences.edit().putString("EMAIL", email).commit();
            sharedPreferences.edit().putString("PASSWORD", password).commit();
            sharedPreferences.edit().putBoolean("REMEMBERLOGIN", true).commit();
        }
    }

    finish();
}

```

Abb. 24: onLoginSuccess()

Wenn der Loginvorgang erfolgreich war, wird `onLoginSuccess()` aufgerufen. Hier wird abgefragt, ob der Benutzer/die Benutzerin das Feld „Benutzerdaten merken“ angehakt hat. Wenn ja, werden die Benutzerdaten in einer `SharedPreferences` gespeichert. Somit muss man sich nicht bei jedem Appstart neu anmelden.

In der Main-Activity wird beim Start der App überprüft, ob die Nutzerdaten in einer `SharedPreferences` gespeichert wurden. Wenn ja, wird der Aufruf der `Loginactivity` übersprungen.

Außerdem gibt es im Menü die Möglichkeit sich abzumelden. Beim Klick auf abmelden, wird die `SharedPreferences` gelöscht und man gelangt wieder zum Loginbildschirm.

### 3.2.4.1 Menü

Im Menü gibt es die Möglichkeit sich abzumelden und sich alle Benachrichtigungen anzeigen zu lassen.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();
    if (id == R.id.action_logout){
        //Delete Logininformation
        sharedPreferences = this.getSharedPreferences(PREFERENCE_KEY, MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.clear();
        editor.commit();

        Intent intent = new Intent(this, LoginActivity.class);
        startActivity(intent);
    }
    if(id == R.id.action_showNotifications){
        Intent intent = new Intent(this, AllNotifications.class);
        startActivity(intent);
    }
    return super.onOptionsItemSelected(item);
}
```

Abb. 25: Optionsmenü

Im Menü wird überprüft, welcher Eintrag ausgewählt wurde. Beim Klick auf das Element mit der id `action_logout`, wird die `SharedPreferences` mit den Nutzerdaten gelöscht und man gelangt zum Loginbildschirm.

Beim Klick auf das Element mit der id `action_showNotifications`, wird eine Activity aufgerufen, in welcher alle Benachrichtigungen in einer Liste angezeigt werden. Beim Klick auf einen Listeneintrag, bekommt man eine Ansicht mit den Details einer Benachrichtigung (betroffene Zähler, Grund der Benachrichtigung, Anmerkung,...).

### 3.2.5 Background Task Notifications

Da man auch ohne geöffnete App benachrichtigt werden will, wenn der Energieverbrauch abweicht, musste ein Background Task entwickelt werden, welcher im Hintergrund, ab dem Start des Smartphones mitläuft und überprüft, ob es Benachrichtigungen gibt.

Hierfür musste im Manifest die Permission für RECEIVE\_BOOT\_COMPLETED gesetzt werden. Dies ist notwendig, da die App ansonsten nicht ab dem Start des Smartphones im Hintergrund laufen darf.

Für diese Aufgabe sind 2 Klassen notwendig. Ein Bootreceiver und ein NotificationAdapter.

Der Adapter ist für das Aussehen und den Inhalt der Notifications zuständig.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        convertView = View.inflate(context, R.layout.notification_list_style, null);
    }

    ImageView images = (ImageView) convertView.findViewById(R.id.imageView);
    TextView title = (TextView) convertView.findViewById(R.id.textViewTitle);
    TextView info = (TextView) convertView.findViewById(R.id.textViewInformation);
    TextView notType = (TextView) convertView.findViewById(R.id.textViewNotificationType);

    images.setImageResource(elemPicture.get(position));
    title.setText(elemTitle.get(position));
    info.setText(elemInfo.get(position));
    notType.setText(elemType.get(position));

    return convertView;
}
```

Abb. 26: NotificationAdapter

Hier wird das AEMS Logo gesetzt, der Titel der Benachrichtigung bestimmt, die Zusatzinfo, welche mit angezeigt werden soll und der Benachrichtigungstyp.

Im Bootreceiver wird überprüft, ob es neue Benachrichtigungen gibt und wenn ja, werden diese generiert und am Smartphone angezeigt. Hierfür wird ein NotificationCompatBuilder benötigt.

Beim Klick auf eine Benachrichtigung kommt man in die Liste aller Benachrichtigungen, um sich alle seine vorhandenen Benachrichtigungen ansehen zu können.

## **4      Verwendete Libraries**

## **5      Technologien**



## **6 Ergebnisse**

### **6.1 Erreichte Ziele**

- Erstellung einer Website mit einer einfachen Übersicht über seine Energieverbrauchswerte.
- Benachrichtigungssystem bei Verfügbarkeit neuer Berichte und für abweichende Verbrauchswerte.
- Entwicklung eines Administrationstools für die Benutzerverwaltung.
- Entwicklung eines Plugins, welches die Möglichkeit bietet eigene Zähler auf Raspberry Pis zu definieren.
- Entwicklung einer Android-App für Benachrichtigungen am Smartphone und Einsicht in die Statistiken.
- Entwicklung einer modularen Anomalieerkennung, welche durch Erstellung von Scripts erweitert werden kann.

### **6.2 Auswirkung**

Durch das einfache Monitoring der Verbrauchswerte und die Benachrichtigung bei zu hohem Verbrauch, soll Energie eingespart werden können. Dies bringt mehrere positive Aspekte mit sich:

- Einsparung von Energiekosten
- Vorteil für die Umwelt durch sinkenden Energieverbrauch

### **6.3 Abschluss**

Mit Stand 25. März 2018 wurde das Projekt abgeschlossen.

## **6.4 Statement des Auftraggebers**

Als Auftraggeber möchte ich dem Projektteam zu ihrer Leistung gratulieren.

Während der Projektdauer waren die Maturanten hoch motiviert. Die Kommunikation, vor allem über E-Mail funktionierte perfekt und zeitnah. Dringende Fragen wurden unverzüglich beantwortet.

Alle 3 Maturanten wirken sehr kompetent, hatten sich die Aufgaben für mich sehr schlüssig verteilt.

Die Software lässt hinsichtlich der geplanten Funktionalität keine Wünsche übrig. Meiner Überzeugung nach ist das entwickelte Tool für die Praxis bestens geeignet. Nach einer inhaltlichen Vorstellung beabsichtigen 3 weitere öö. Klima-und Energiemodell-Regionen einen Einsatz mit ca. 1.500 Zählern durchzuführen.

Jedes Unternehmen kann sich glücklich schätzen, einen dieser 3 Absolventen für sich als Mitarbeiter gewinnen zu können.

(Pözlberger Herbert 2018, E-Mail)

## **7 Danksagung**

Ein ausdrücklicher Dank gebührt unserem Auftraggeber Herrn Ing. Herbert Pözlberger, MSc für die Bereitstellung dieses Projekts als Diplomarbeit. Wir möchten uns sehr herzlich dafür bedanken, dass wir dieses Projekt in Kooperation mit der Energiegenossenschaft Eferding realisieren durften.

Vielen Dank für das tolle Projekt und die sehr gute Zusammenarbeit!

Ein Dank gebührt auch unserem Betreuungslehrer, Herrn DI Josef Doppelbauer, welcher uns bei Fragen immer mit Rat und Tat zur Seite stand.

## Quellen und Literaturverzeichnis

**Printquellen**

**Digitale Quellen**

# Verzeichnis der Abbildungen, Tabellen und Abkürzungen

Ut erem sequibus remquam el ipsaere perrovi.....9

A b k ü r z u n g	Bedeutung
USB	Unitversal Serial Bus
RAM	Reandom Access Memory
SQL	Structured Query Language