to:Huus

Projektdokumentation

Entwicklung eines Smart-Homes als Webapplication

Browser: Chrome, OS: Windows, DB: config/dbConf.json, Startprozedur: Main.go

Dokumentation zur Hausarbeit für das
Modul Webprogrammierung an der Hochschule Flensburg

von **Sven Kuhlmann**

MatrikelNr: 610292
Student: Sven Kuhlmann
Geboren am: 09.01.1994 in Leer
Durchführungszeitraum: 15.11.2017 – 20.12.2017

INHALTSVERZEICHNIS

1. EINFÜHRUNG

Fachbegriffe werden in keinem Glossar genauer beschrieben, weshalb diese und verwendeten Abkürzungen in den Fußnoten ausgeschrieben werden.

Quellcode wird logisch auf das wesentliche begrenzt und reduziert. Dies ist mit „…“ in der gesamten Dokumentation angegeben. Ist der Schnitt eindeutig logisch, so ist dieser nicht angegeben.

Die Informationen richten sich selbstverständlich an Frauen und Männer gleichermaßen. Im Text wurde jedoch zugunsten der Lesbarkeit und aus sprachlichen Gründen nur die männliche Form gewählt.

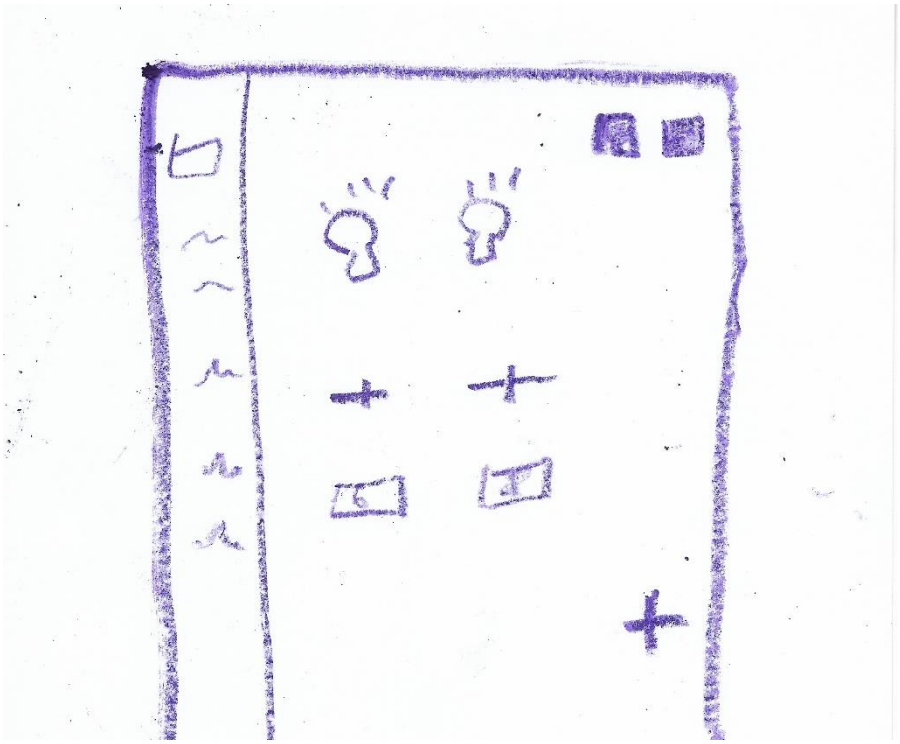
2. LAYOUT

2.1. ENTWURF

Zu Beginn des Projektes wurde in der Vorbereitung von mir eine Skizze zum Layout erstellt. Durch diese Skizze wird schon vor der Programmierung überlegt, in welche Richtung die Software gehen soll. Dadurch werden sich Gedanken gemacht und man hat damit bereits eine Vorstellung von der Umsetzung. Anhand der Skizze habe ich mir einen Header[1] mit Navigation, an der linken Seite positioniert, vorgestellt. Dieses harmoniert sehr gut mit den fixen Buttons, die auf der rechten Seite positioniert sind. Über einen Button lässt dann zudem auf die UI des Simulators wechseln. Das User Interface in sich, ist eine Single-Page[2] Anwendung. (Siehe folgendes Skizze)

[1] Abschnitt im rechten Bereich der Seite

[2] Eine Webseite die durch nachladen von Inhalt auf realisiert wird

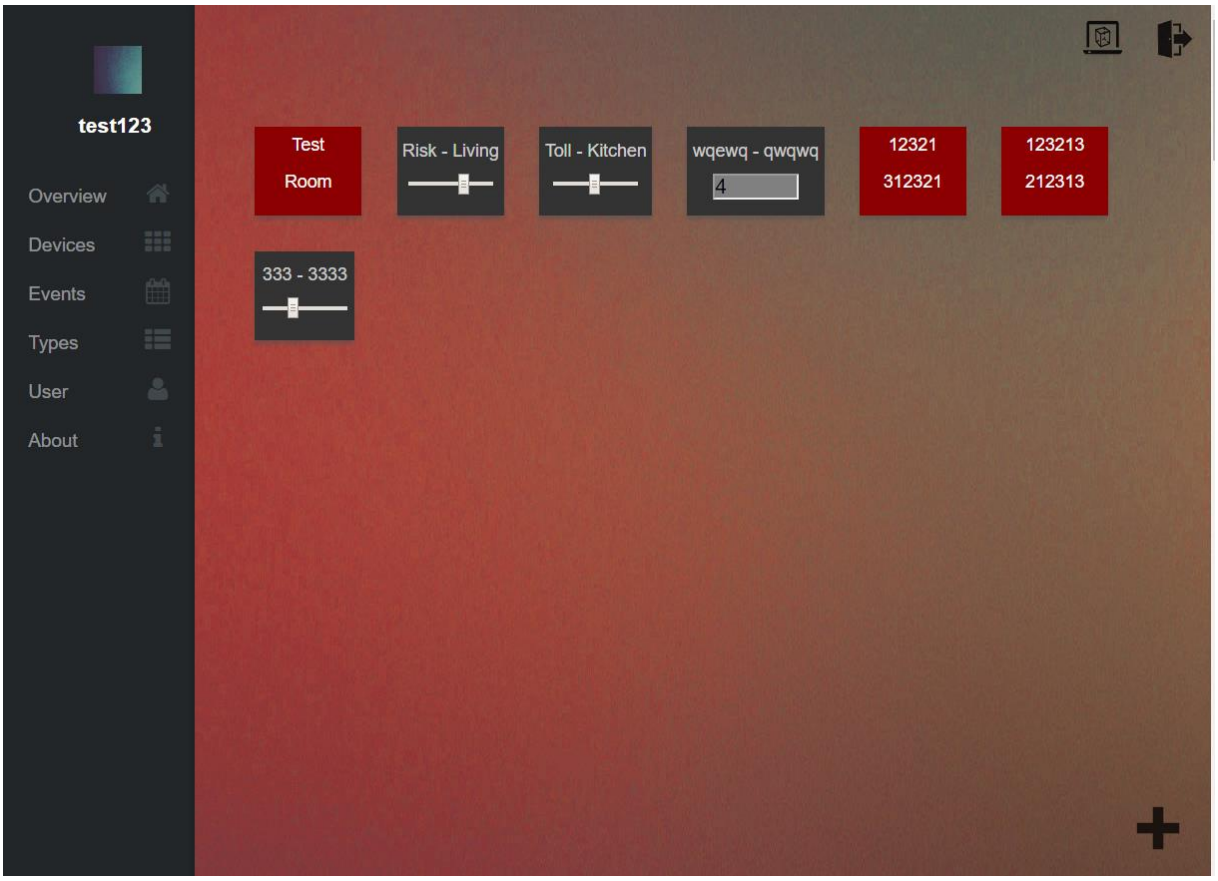


Entwurfsskizze

2.2. UMSETZUNG

Die Umsetzung des Layouts erfolgte sehr nah an dem Entwurf. Lediglich an der Navigation hat sich etwas geändert. Außerdem ist die Übersicht der Geräte und deren Status anders ausgefallen.

Die Navigation auf der rechten Seite wurde um ein paar Elemente, zur besseren Bedienung, ergänzt. (Siehe folgendes Bild)



Umgesetztes Layout von to:Huus

Alle Bereiche wurden hierbei logisch und sinnvoll platziert. Es wird dabei viel Wert auf die Erfahrungen der Nutzer gelegt. Durch die gewählte Platzierung, die das lange suchen der wichtigsten Elemente vermeidet, kann sich der Nutzer leicht zurechtfinden.

Alle Farben, die in den Settings durch das Layout geändert werden können, stehen in einem angenehmen Kontrast und Verhältnis. Durch das Testen mit einigen Nutzern, konnten diese als sehr angenehm beurteilt werden.

Das Layout wurde in der Struktur mit HTML5 sowie JS(ES6) und vom Design mit CSS3 umgesetzt. Zusätzlich wurde FontAwesome für ein schöneres Auftreten genutzt.

2.3. BEDIENKONZEPT

Über Buttons, die fix positioniert sind, lassen sich die Basis-Aktionen ausführen. Simulator aufrufen, Logout und das Hinzufügen von Items. In der Übersicht werden alle Geräte mit ihrem aktuellen schalt Status gelistet. Hier kann durch einfache Interaktion der Status, mittels eines Ajax[3]-requests, geändert werden. Über die Navigation oder das Scrollen wird zu der Übersicht der einzelnen Elemente gewechselt. Hier stehen weiter Interaktion zur Verfügung.

3. AJAX

3.1. UI

Mittels AJAX werden in der UI die elemente in einem Interval aktuallisiert. Die Items, die als JSON ankommen, werden anschließen mithilfe von DOM zusammengebaut.

```
…
let getString = "AllDevices";
let data = "?Get="+getString;
let url = address + "/ui/get" + data;
let xhr = new XMLHttpRequest();
…
```

Zudem werden sämtliche weitere Modifikation, die das Editieren und Löschen betreffen, mithilfe von Ajax realisiert.

3.2. SIM

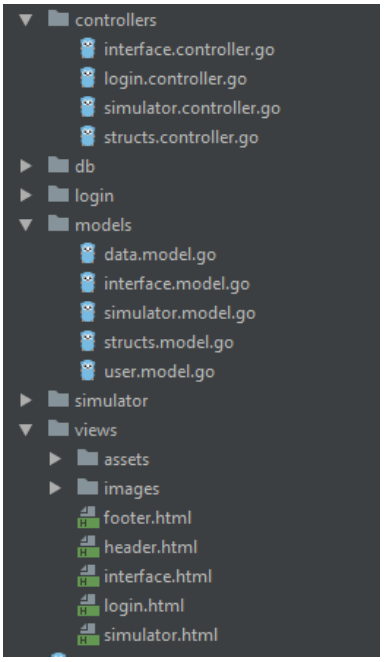
Im Simulator-Interface werden mithilfe von Ajax die zustände des Simulators nachgeladen.

4. STRUKTUR

4.1. MVC

Bei der Struktur wurde MVC realisiert. Hierbei sind die funktionen in jeweiligen Klassen der Datenmodifizierung (Models), zur übersetzung und dem austausch zwischen den Schichten (Controllers) und der Visualisierung (Views) aufgeteilt.

Model	Controller	View



4.2. PAKETE

Folgende Pakete, die dem Standard abweichen, werden genutzt:

- gopkg.in/mgo.v2 (Public)
- login (Eigene)
- db (Eigene)

[3] Asynchron Jason and XML

4.3. HANDLER / TEMPLATES

Die registrierten Handler leiten die Request und die richtigen Controller weiter, wo sie weiterverarbeitet werden.

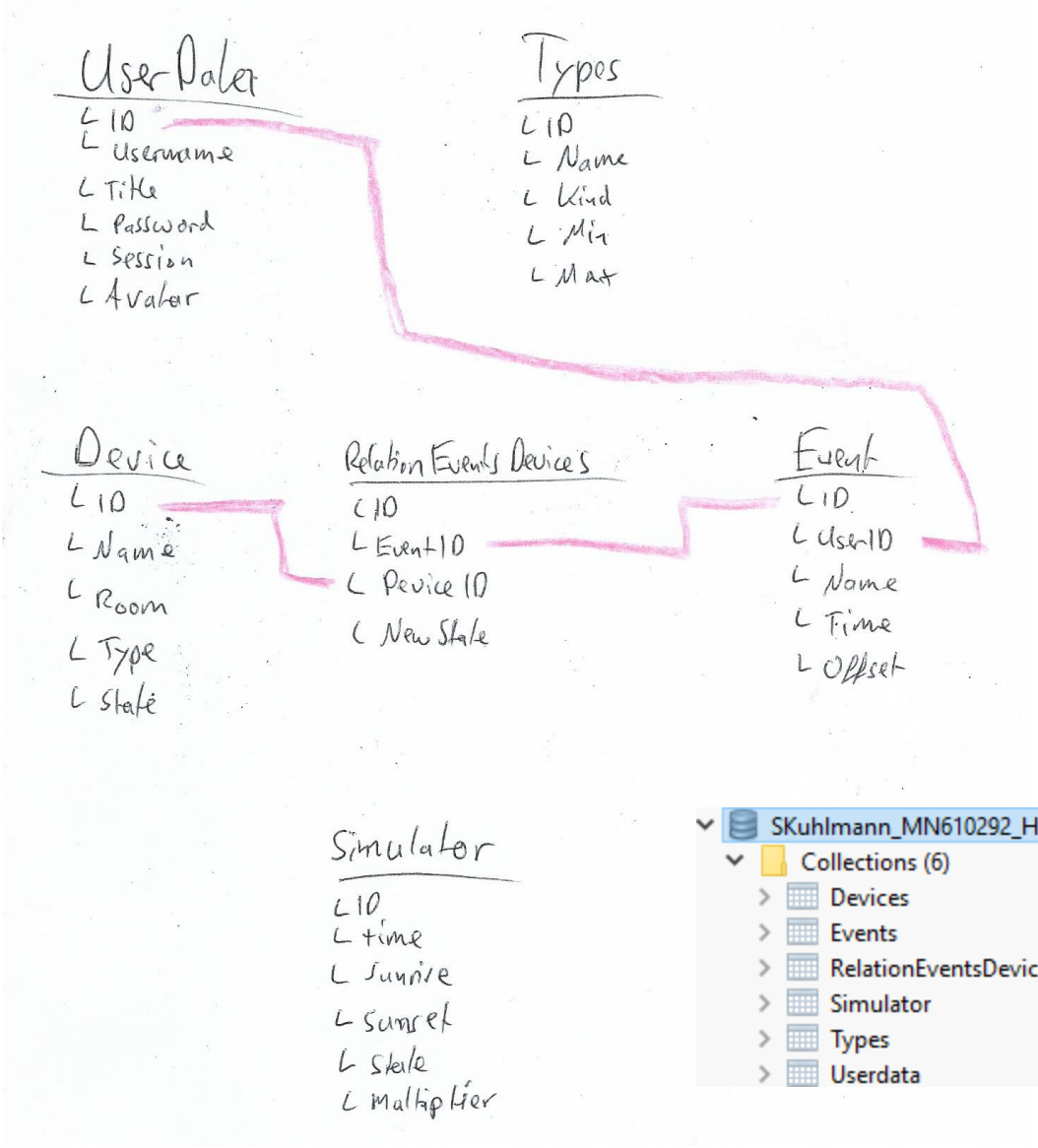
```
http.Handle("/images/", http.FileServer(http.Dir("./toHuus/views/")))
http.Handle("/assets/", http.FileServer(http.Dir("./toHuus/views/")))
http.Handle("/avatar/", http.FileServer(http.Dir("./toHuus/conf/")))
http.HandleFunc("/", controllers.CheckLogin)
http.HandleFunc("/ui", controllers.InterfaceHandler)
http.HandleFunc("/ui/user", controllers.UserHandler)
http.HandleFunc("/ui/add", controllers.AddHandler)
http.HandleFunc("/ui/get", controllers.GetHandler)
http.HandleFunc("/ui/del", controllers.DelHandler)
http.HandleFunc("/ui/set", controllers.StateHandler)
http.HandleFunc("/sim", controllers.SimulatorHandler)
http.HandleFunc("/sim/data", controllers.DataHandler)
http.HandleFunc("/sim/get", controllers.SimGetHandler)
http.HandleFunc("/sim/set", controllers.SimSetHandler)
```

Templates wurde in drei ebenen aufgeteilt. Header – Content – Footer. Dabei enthält der Header bekannte Informationen zur Initialisierung auf dem Aufbau. Das Content-Template ist in allen drei Seiten unterschiedlich. Das heißt: für das Login, User Interface und Simulator Interface werden unterschiedliche Inhalte durch die Templates geladen, der Aufbau bleibt aber durch einen einheitlichen Header gleich.

5. DATENBANKSTRUKTUR

Die Datenbank besteht aus folgenden Collections: Geräten (Devices), Szenen (Events), Gerätetypen (Types), Simulationsstatus (Simulator), Benutzerdaten (UserData) und Scenen-Relationen (RelationEventsDevices).

Die Daten der Collections können dem folgenden Bild entnommen werden (Skizze).
Besonders wichtig ist hierbei die Relation. Diese ist für das Zusammenspiel zwischen Events und Devices. Für jedes Gerät einer Szene gibt es eine Relation.



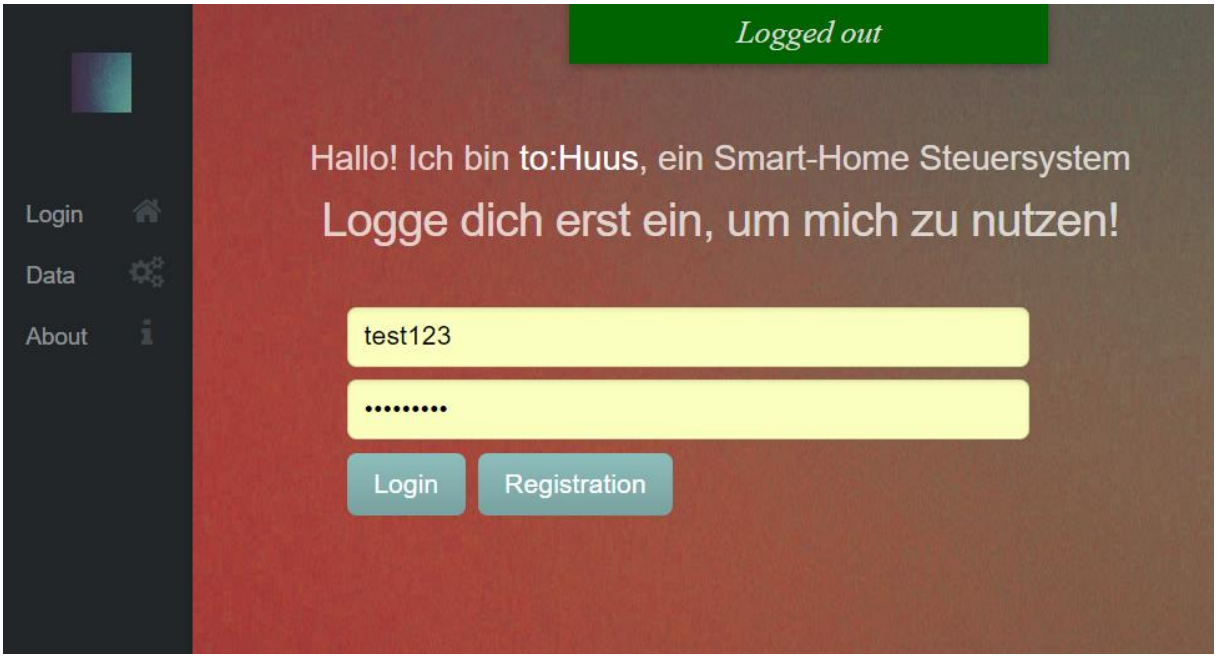
6. XML

Für den Import werden die Daten in einer Funktion gelesen und geparst. Anschließend werden die Daten in die Datenbank gelesen. Für den Export werden alle Daten in einen Structs gelesen und mithilfe eines Korrekten Headerinformationen an den Client gesendet.

```
w.Header().Set("Content-Disposition", "attachment; filename=toHuus.xml")
w.Header().Set("Content-Type", "application/octet-stream")
w.Header().Add("Access-Control-Expose-Headers", "Content-Disposition")
// Parse and send
xml, _ := xml.Marshal(data)
io.Copy(w, strings.NewReader(string(xml)))
```

7. BENUTZERVERWALTUNG

Bevor sich ein Benutzer anmelden kann, muss sich dieses Registrieren. Anschließen muss er seine Daten erneut eingeben, um diese zu bestätigen und sich zusätzlich ein zu loggen.



10. FAZIT / BESONDERHEITEN

10.1. PROBLEME

Es sind einige Probleme bei dem Programmieren und Scripten aufgetaucht. Dies ist bei einem solchen Umfang sehr wahrscheinlich. Die meisten Fehler/Probleme sind nur sehr klein gewesen und konnten oft durch das Internet schnell gelöst werden.

Probleme gab es besonders bei dem Konvertieren und Arbeiten mit der Zeit. Als Lösung wurde hier die Zeit in Sekunden umgerechnet.

Das größte Problem meinerseits war das Zeitmanagement. In der letzten Woche anzufangen, grenzt einen enorm ein. Einige Funktionen können aus Zeitmangel nicht geleistet werden.

10.2. BESODNERHEITEN

- **Statusmeldungen / Fehlermeldungen:**

Bei erfolgreichen Aktionen werden hier entsprechende Meldungen ausgegeben. Diese werden kurzzeitig in einem Dialog eingeblendet.

- **Userverwaltung:**

Gute Struktur der Datenbank ermöglichen Zusatz Funktionen wie das setzten eines Avatar-Bildes

- **Stay in:**

Auch wenn du die Seite verlässt oder neu Lädst, wird durch eine Session dein Login kurzzeitig zurückgehalten und du bleibst drin! So wird das einfache weiterarbeiten ermöglicht.

- **Responsive:**

Die ganze Webapplication ist responsive aufgebaut. Dadurch ist sie für alle Geräte, sowie Smartphones aufrufbar. Dies wurde mit CSS gelöst.

10.3. NICHT ERFÜLLT

- **Simulator**

Durch Zeitmangel konnte ich den Simulator nicht zuende führen. Aus irgendeinem Grund werden die Szenen nicht mehr umgesetzt. Das modifizieren des Simulators funktioniert hingegen noch.

- **Min, Max**

Es sind keine Min und Max werte für die meisten Inputs gesetzt.

Allerdings lassen sich sämtliche Funktionen einfach nachtragen. Dies ist besonders durch die Modulare Programmierung einfach gegeben. Mit mehr Zeit wären diese Funktionen auch kein Problem gewesen.

11. ABSCHLUSS

11.1. SIMULATOR

Der Simulator wird in einem Thread automatisch von der Main gestartet.

11.2. RESPONSIVE

Die Seite ist durch ihr Responsive-Layout auch für mobile Geräte geeignet. Durch einen in CSS-gelösten Menü-Button lässt sich dort die ausgeblendete Navigation einfach einblenden.

11.3. SOFTWARE-REFERENZEN

Bei der Entwicklung wurden die IDEs *Brackets* und *GoLand von Intellij* verwendet. Als Browser kam nur *Google Chrome* zum Einsatz. Für die Bildbearbeitung wurde *GIMP2* genutzt. Für die Erstellung der Dokumentation wurde *Microsoft Word* genutzt.

11.4 QUELLENVERWEIS

Alle verwendeten Bilder und Informationen stammen von mir oder stehen unter einer Creative Commons Zero Lizenz (CC0) und sind somit frei für dritte.