



Technische Universität Berlin

# Praxisprojekt Anwendungssysteme

Connected Living Advisor

*Ein Wegweiser für SmartHomes / Smart Spaces*

## LEITUNG

PROF. DR. SAHIN ALBAYRAK  
[SAHIN.ALBAYRAK@DAI-LABOR.DE](mailto:SAHIN.ALBAYRAK@DAI-LABOR.DE)

SEBASTIAN AHRNDT  
[SEBASTIAN.AHRNDT@DAI-LABOR.DE](mailto:SEBASTIAN.AHRNDT@DAI-LABOR.DE)

JAKOB SCHOFER  
[JAKOB.SCHOFER@CONNECTED-LIVING.ORG](mailto:JAKOB.SCHOFER@CONNECTED-LIVING.ORG)

## ERSTELLT VON

MICHAEL CHLEPAKOV - 330254  
TIM MALTE GRÄFJE - 348865  
TIM MALESKA - 356211  
MOHAMED MEGAHED - 342655  
MARVIN PETZOLT - 350455  
TIMO RUNCK - 359651  
OSKAR SCHLÖSINGER - 347855  
MARINA SCHNEIDER - 309060

17. MÄRZ 2017

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>6</b>
1.1 Motivation . . . . .	6
1.2 Über den Projektpartner . . . . .	7
1.3 Über das Team . . . . .	8
<b>2 Ausgangssituation</b>	<b>11</b>
2.1 Philosophie . . . . .	11
<b>3 Zielbestimmungen</b>	<b>13</b>
3.1 Abgrenzung . . . . .	14
3.2 Anforderungen . . . . .	14
3.2.1 Muss-Kriterien . . . . .	14
3.2.2 Kann-Kriterien . . . . .	16
3.2.3 Fertigstellungskriterien . . . . .	16
3.3 Ideeentwicklung . . . . .	16
3.4 Inspirationen . . . . .	17
<b>4 Produkteinsatz</b>	<b>18</b>
4.1 Zielgruppen . . . . .	18
4.1.1 Anwendungsbenutzer . . . . .	19
4.1.2 Community . . . . .	20
4.1.3 Produkthersteller . . . . .	22
<b>5 Aufbau des Beratungsprozesses</b>	<b>23</b>
5.1 Onboarding . . . . .	23
5.2 Szenarioauswahl . . . . .	24
5.3 Produktauswahl . . . . .	25
5.4 Übersicht . . . . .	26
<b>6 Datenbank und Modellierung</b>	<b>27</b>
6.1 Allgemein . . . . .	27
6.1.1 Features . . . . .	27
6.1.2 MetaDevice . . . . .	28
6.1.3 Broker . . . . .	28
6.1.4 Endpoint . . . . .	28
6.1.5 Scenario . . . . .	28
6.1.6 Rating . . . . .	29
6.1.7 Model . . . . .	29
<b>7 Matchmaking</b>	<b>38</b>

7.1	Anforderungen an den Matchmaking Algorithmus . . . . .	39
7.2	Szenario auswählen . . . . .	40
7.2.1	Implementation von Metadevices . . . . .	42
7.2.2	Finden von Produktkompatibilitäten . . . . .	43
7.2.3	Auswahl der optimalen Lösung . . . . .	45
7.3	Implementation mehreren Szenarien . . . . .	46
7.3.1	Merging von Metadevices . . . . .	46
7.3.2	Produkttypfilter . . . . .	48
7.4	Produktalternativen . . . . .	49
7.5	Berechnung von Preis, Erweiterbarkeit und Effizienz . . . . .	51
7.5.1	Herausforderungen und Designentscheidungen . . . . .	52
7.6	Zusammenfassung . . . . .	54
<b>8</b>	<b>Infrastruktur - Backend</b>	<b>55</b>
8.1	Django . . . . .	55
8.2	Memcached . . . . .	56
8.3	Swagger . . . . .	56
<b>9</b>	<b>Anleitung - Backend</b>	<b>58</b>
9.1	Installation . . . . .	58
9.1.1	Abhängigkeiten . . . . .	58
9.1.2	PostgreSQL . . . . .	60
9.1.3	Andere Datenbanken . . . . .	60
9.2	Start . . . . .	61
<b>10</b>	<b>Deployment</b>	<b>62</b>
10.1	Aktuelle Konfiguration . . . . .	62
10.1.1	Nginx, Gunicorn und PostgreSQL . . . . .	62
10.1.2	Git Repositories und Hooks . . . . .	63
10.2	Optimale Konfiguration . . . . .	63
10.2.1	Anwendungsserver . . . . .	66
10.2.2	Nginx . . . . .	66
10.2.3	Statische Inhalte . . . . .	66
10.2.4	Database-Backend . . . . .	66
10.2.5	Cache-Backend . . . . .	67
10.2.6	Load-Balancer . . . . .	67
10.3	Admin . . . . .	67
10.3.1	Startseite Verwaltung . . . . .	67
10.3.2	Änderungen vornehmen . . . . .	67
<b>11</b>	<b>Infrastruktur Frontend</b>	<b>70</b>

11.1	Angular 2 . . . . .	70
11.2	npm . . . . .	71
11.3	TypeScript . . . . .	71
11.4	JQuery . . . . .	72
11.5	Bootstrap . . . . .	72
11.6	Material Kit . . . . .	72
<b>12</b>	<b>Aufbau</b>	<b>73</b>
<b>13</b>	<b>Komponenten</b>	<b>74</b>
13.1	Index-Seite . . . . .	74
13.2	Onboarding . . . . .	76
13.3	Szenarioauswahl . . . . .	77
13.3.1	Szenarioliste . . . . .	77
13.3.2	Filterfunktion . . . . .	77
13.3.3	Hausansicht . . . . .	77
13.4	Produktauswahl . . . . .	78
13.5	Übersicht . . . . .	79
13.5.1	Druckansicht . . . . .	80
<b>14</b>	<b>Anleitung - Frontend</b>	<b>81</b>
<b>15</b>	<b>Tests</b>	<b>86</b>
15.1	Backend . . . . .	86
15.2	Frontend . . . . .	87
15.3	Bugs . . . . .	89
<b>16</b>	<b>Ablaufplan</b>	<b>90</b>
16.1	Zeitplan . . . . .	90
16.2	4. Meilenstein . . . . .	92
16.3	5. Meilenstein . . . . .	92
16.4	6. Meilenstein . . . . .	92
16.5	Projektabschluss . . . . .	93
16.6	Ausblick . . . . .	93
<b>17</b>	<b>Anhang</b>	<b>94</b>
17.1	Wichtige URLs . . . . .	94
17.2	Glossar . . . . .	94
17.3	Präsentationsfolien . . . . .	98
17.3.1	4. Meilenstein . . . . .	98
17.3.2	5. Meilenstein . . . . .	104
17.3.3	6. Meilenstein - Interne Präsentation . . . . .	120

## Abbildungsverzeichnis

1	Klassendiagramm des gesamten Models . . . . .	37
2	Beispielanfrage an den „Suggestions“ Endpoint. . . . .	38
3	Werte die in der Szenarioauswahlphase benutzt werden . . . . .	41
4	Formel um den Abstand zwischen $\vec{b}$ und $\vec{s}$ zu finden . . . . .	41
5	Die Renovierungspräferenz restriktiert die Menge der möglichen Implementationen eines Metadevices. . . . .	42
6	Produktpräferenzen unterteilen sich in „Preis“, „Erweiterbarkeit“ und „Effizienz“. . . . .	44
7	Formel für die Berechnung der Wertigkeit $R_{U_{pref}}$ eines Produktsets. . . . .	45
8	Beispielanfrage an den /suggestions Endpoint. . . . .	48
9	Beispielanfrage an den /final_product_list Endpoint. . . . .	50
10	Swagger Methodenübersicht der API . . . . .	57
11	Django package update skript . . . . .	59
12	PostgreSQL Konfiguration settings.py . . . . .	60
13	Frontend deployment Skript . . . . .	63
14	Backend deployment post-receive hook . . . . .	64
15	deployment_database_backup.sh Backup von der Datenbank in eine JSON Datei. . . . .	64
16	deployment_script_production.sh Löschen der Datenbank und neu laden aus der vorhandenen JSON Datei. . . . .	65
17	restart_server.sh Neustart des NGINX-Servers . . . . .	65
18	Willkommensseite . . . . .	68
19	Admin Listenübersicht einer Entität/Relation . . . . .	69
20	Darstellung der vier Beratungsphasen . . . . .	75
21	Vorteile von DONAALDA als Berater . . . . .	76

# **1 Einleitung**

## **1.1 Motivation**

Ziel des Projektes ist die Erweiterung eines Assistenzsystems, das dem Verbraucher Informationen über vernetzte Produkte und Services aus dem Smart-Home Bereich und deren Einsatzmöglichkeiten in allen Lebensbereichen verständlich zur Verfügung zu stellt. Interessenten haben die Möglichkeit, sich über die Systeme online zu informieren, einen Überblick zu erhalten welche SmartHome Systeme und -Lösungen welche Hersteller auf dem Markt existieren, welche Vor- und Nachteile sie haben und welche Lösungen für den Verbraucher in seiner individuellen Lebenssituation am besten geeignet sind. Die Informationsbereitstellung und Aufklärung über die Mehrwerte der Technologien und deren Einsatzmöglichkeiten führen zum Abbau von Hemmnissen auf Seiten des Verbrauchers. Die Use-Case-basierte Darstellung der Technologien erlaubt es dem Nutzer, sich in das Szenario hineinzuversetzen und die Mehrwerte eindeutig zu erkennen. Das Projekt ist auf einem bestehenden System aufgebaut, erweitert dieses inhaltlich und visuell und entwickelt ein Mechanismus zur Produkt-Recommendation. Zusammen mit dem Industriepartner Connected-Living e.V. wurde darauf hingearbeitet, die Lösung auf der Connected Living ConnFerence 2017 einem interessierten Publikum erstmals vorzustellen.

Das Projekt findet im Rahmen des Moduls Praxisprojekt Anwendungssysteme an der TU Berlin statt und wird vom Fachgebiet Wirtschaftsinformatik – Information Systems Engineering (ISE) an der Fakultät IV betreut. Der Lehrstuhl erforscht die software-technische Gestaltung und interdisziplinäre Bewertung von verteilten, auf Plattformen basierten IT-Systemen in anspruchsvollen, zukunftsweisenden Anwendungsdomänen – innovativ, technikorientiert, mit Praxisbezug und mit dem Anspruch auf internationales Spitzenniveau. Das Fachgebiet ist seit 2014 unter Leitung von Prof. Dr.-Ing. Stefan Tai; Praxisprojekt Anwendungssysteme wird gemeinsam mit Dr.-Ing Frank Pallas angeboten. Es erlaubt Studenten diverse Facetten einer realen Projektarbeit kennenzulernen (Entscheidungsfindung, Meetings, Analyse und Lösungsfindung, Präsentation von Ergebnissen, Umsetzung von Prototypen) und fördert teamorientiert komplexe Aufgaben im Kontext der Gestaltung und Bewertung von IT-Systemen zu bewältigen.

Institut Wirtschaftsinformatik und quantitative Methoden  
Information Systems Engineering  
Einsteinufer 17  
10587 Berlin

<http://www.ise.tu-berlin.de/>

## 1.2 Über den Projektpartner

Connected Living ist mit mehr als 50 Mitgliedern die größte Open-Innovation-Plattform zur Entwicklung von Partnerschaften und Lösungen für das Smart-Home und digital vernetzte Leben. Im Connected Living Mitgliedernetzwerk haben die Partner die Möglichkeit, gemeinsam mit Unternehmen und Verbänden sowie Forschungseinrichtungen in gewinnbringenden Kooperationen und Projekten kundenzentrierte, intuitiv nutzbare Technologien und Lösungen zu entwickeln, um die steigenden Marktwachstumspotentiale zu erschließen. Mit zahlreichen Mitgliedern aus unterschiedlichsten Segmenten ist Connected Living interdisziplinär aufgestellt und ermöglicht einen ganzheitlichen Blick auf das vernetzte Leben. Die Bandbreite des Netzwerks reicht von Institutionen, Verbänden und Vertretern der Wohnungswirtschaft über Energieversorger, Hausgerätehersteller, Telekommunikationsdienstleister, Anbieter von Sicherheitstechnologien und Unternehmen aus dem Gesundheitssektor bis hin zu Beratungsunternehmen, Banken und Versicherungen. Mitglieder sind u.a. die Deutsche Telekom, Vodafone, Telefónica, Cisco, SAP, Vattenfall, RWE, AOK Bundesverband, Sanofi, Johanniter, Allianz, Miele, Comdirect, Schwäbisch Hall, Gesamtverband Deutscher Wohnungswirtschaft und viele weitere.

Connected Living e.V.  
Helmholtzstraße 2 - 9  
10587 Berlin

<http://www.connected-living.org/>

## 1.3 Über das Team

*Michael Chlepakov*, Backend - Wirtschaftsinformatik, BSc

Durch Teilnahme am letztsemestrigen Ambient Assisted Living Projekt konnte ich erste Einblicke in das Themengebiet Smart Home bekommen. Dies hat mein Interesse an der Branche im Allgemeinen und an der Webentwicklung im speziellen geweckt und daher war ich an einer Weiterführung unseres Projektes interessiert. Spannend war auch die Frage, wie wir mit dem von uns entwickelten Demonstrator weiter verfahren werden. Hier wollte ich natürlich wieder dabei sein und habe mich dann auf eine der freien Plätze beworben.

*Tim Malte Gräfje*, Backend - Informatik, MSc

Mein Interesse an diesem Projekt war hauptsächlich technischer Natur. Ich hatte schon öfter in kleineren Freizeitprojekten mit Django gearbeitet und wollte diese Fähigkeiten nun in einem größeren Projekt anwenden und vertiefen. Außerdem hat mich die Aussicht den Matching Algorithmus zu designen fasziniert. Zu guter Letzt wollte ich es mir nicht entgehen lassen, lustige Gifs in die Team interne Slack Gruppe posten.

*Tim Maleska*, Frontend - Wirtschaftsinformatik, BSc

Die Vernetzung im Smart Home für den Verbraucher so angenehm wie möglich zu gestalten, war meine Motivation für das PAS-Projekt Connected Living. Neben spannenden Technologien, der engen Zusammenarbeit mit dem Kunden und der Herausforderung, das Projekt eigenständig zu organisieren und zu gestalten.

*Mohamed Megahed*, Frontend - Wirtschaftsinformatik, BSc

Die Projektarbeit war aus meinem Interesse an Webentwicklung mit einer Praxiserfahrung ansprechend und hat sich als gute Auswahl ergeben, da ich zu Semesterbeginn an dem Modul Cloud Services Engineering and Management interessiert war. Eine Gruppenarbeit hatte meine Vorstellung erfüllt insofern, dass wir meistens selber die konkreten Aufgaben gesucht haben. Dank einer guten Zusammenarbeit mit dem Team hat das auch die Produktivität erhöht.

*Marvin Petzolt*, Backend - Informatik, MSc

Mein Interesse an diesem Projekt bestand vorwiegend in dem Anwendungsbereich Smart Home, welches meiner Meinung nach den nächste Schritt in die Richtung personalisiertes modernes Leben darstellt. Da ich schon meine Bachelorarbeit in diesem Bereich geschrieben habe, hatte ich schon Vorwissen über diese Branche. Vor allem hat mich aber das Matchmaking fasziniert, da ich mich gerne mit komplexen Fragestellungen beschäftige.

*Timo Runck*, Backend/Frontend - Wirtschaftsinformatik, BSc

Bereits letztes Semester habe ich an diesem Projekt teilgenommen und nun die Möglichkeit zu haben das bestehende System zu erweitern und weiterzuentwickeln hat mich gereizt. Im Semester zuvor habe ich bereits Backentechnologien wie Django und Python kennengelernt. Damit haben wir zu dem Zeitpunkt allerdings keinen anspruchsvollen Algorithmus entwickelt, um passende Produkte, den Anforderungen der Nutzer entsprechend, matchen zu können. Die Umsetzung eines solchen Algorithmus zu sehen hat mich interessiert. Aber auch die Möglichkeit im Frontend eine benutzerfreundlichere Oberfläche umzusetzen und dabei mit neuen Technologien, wie bspw. Angular 2 zu arbeiten, empfand ich als überaus spannend, da ich vorher überwiegend im Backend gearbeitet habe und ich noch Erfahrungen im Frontendbereich sammeln wollte.

*Oskar Schlösinger*, Frontend - Wirtschaftsinformatik, BSc

Durch verschiedene Kurse des Lehrstuhls im Bereich Smart Home habe ich Interesse auch an diesem Modul entwickelt. Da ich im vorherigen Semester zusammen mit einem Teil der Gruppe ein „Proof of Concept“ des Projektes erfolgreich erarbeitet habe, war ich durch mein Fachwissen in diesem spezifischen Bereich zusätzlich motiviert, mich weiterhin und intensiv mit den Problemstellungen und möglichen Lösungsansätzen in der Theorie und insbesondere in der praktischen Umsetzung zu beschäftigen.

*Marina Schneider*, Frontend - Wirtschaftsingenieurwesen, BSc

Durch meine Teilnahme im AAL-Projekt vor zwei Jahren konnte ich schon erste Einblicke ins Connected Living-Bereich bekommen. Mein Interesse an dem aktuellen Projekt hat vor allem seine Verbundenheit mit den Themen IoT und SmartHome geweckt, die immer mehr an Bedeutung in der Forschung und Industrie zunehmen. Auch das Sammeln von Praxiserfahrung in den Bereichen Webdesign und -entwicklung, die Ausübung der Kreativität, und die Ideensammlung und Arbeit in einem Team, wo man voneinander lernen kann, waren ein bedeutsamer Aspekt des Projekts, den ich in das weitere Berufsleben mitnehmen kann.

## 2 Ausgangssituation

Das Projekt ergänzt die Idee des Vorsemesters, nämlich vom Modul Ambient Assisted Living im SoSe 2016, an dem einige Teammitglieder teilgenommen haben. In diesem Semester war die Herausforderung, einen Advisor zu entwickeln und für reale Anwendungsfälle in Betrieb setzen zu können, was im Vergleich zu vielen Modulen über den Rahmen einer Veranstaltung hinausgeht und dementsprechend den Ablauf einer Projektarbeit in der Industrie gut simuliert hat. Der Kunde war durch die Projektbetreuer sowohl vom DAI-Labor als auch von Connected Living vertreten, die einerseits eher Richtlinien als Vorschriften bezüglich Technologie und Aufgabenbereiche gegeben haben, und andererseits uns mit nötigen Ressourcen versorgt haben wie z.B. Designberatung und Hardware für die Vorstellung auf der Connected Living Conference 2017. Im Prinzip war viel eigenständige Arbeit angesagt, da ein solches Tool nicht auf dem Markt existiert. Das hat im Laufe des Projekts viel Entscheidungstreffen mit und ohne den Betreuer erfordert. Das hatte manchmal frustrierende Momente zufolge, da man sich nicht an vorherigen Marktanalysen oder Nutzerberichte ausrichten konnte, hatte aber auch den Vorteil, dass wir mehr Gestaltungsfreiraum hatten.

### 2.1 Philosophie

*Berater, der / Ratgeber; jemand, der (berufsmäßig auf seinem Fachgebiet) Rat erteilt.*

Im Kontext eines Händlers, hat ein Berater im Alltag hat die wichtige Funktion, dem Verbraucher möglichst ausführliche Informationen zukommen zu lassen, damit dieser eine bessere Entscheidung treffen kann. Hier kann einiges schief gehen. z.B. können die Anforderungen vom Verbraucher zum Berater falsch vermittelt bzw. falsch verstanden werden. Außerdem kann die Information vom Berater subjektiv, falsch oder irrelevant sein. Diese Interaktion ist auch von Berater zu Berater Unterschiedlich. Unsere Philosophie war diesen Beratungskontext möglichst real zu halten und dabei aus solchen Problemen einen Vorteil für unseren Online-Berater zu schaffen. Daraus haben sich folgende Stärken ergeben:

- Ein Berater, der jederzeit und überall verfügbar ist. Das erspart mehrere Fahrten zum Elektrofachgeschäft

- Ein Berater, der die gleiche Information an dem Nutzer vermittelt und keine Voreingenommenheiten oder Präferenzen für den Vertrieb bestimmter Produkte sorgt
- Ein Berater, der ein Werkzeug ist, dass dem Nutzer die Freiheit gibt, mit eigenen Konstellationen herumzuexperimentieren.

### **3 Zielbestimmungen**

Ausgangspunkt dieses PAS-Projekts, betreut vom AOT Lehrstuhl / DAI-Labor von Prof. Albayrak, war die Weiterentwicklung eines rudimentären Smart Home Advisors, um die Technologie im Bereich Smart Home für den Verbraucher beherrschbar zu machen.

Im Sommersemester 2016 formulierte der Verein Connected Living e.V. (nachfolgend CL) die Vision, den Bereich Smart Home mit einem Advisor für Kunden attraktiver zu gestalten. Der Advisor soll Informationen über Produkte und Services aus dem Smart Home bzw. Connected Living Umfeld und deren Einsatzmöglichkeiten in allen Lebensbereichen zur Verfügung zu stellen. Somit kann sich der Kunde über die Hersteller und Einsatzmöglichkeiten informieren und erfährt, welche Lösungen auf dem Markt existieren. Entsprechend angegebener Präferenzen erfolgt dann eine Empfehlung geeigneter Produkte und Services, welche die individuellen Kundenwünsche bestmöglich erfüllen können. Diese werden anhand von vordefinierten Use Cases angeboten, um ein besseres Bild über die Möglichkeiten und Lösungen seines Smart Homes bekommen. Der Advisor übernimmt hierbei die Funktion eines Beraters, vergleichbar mit einem Konfigurator für Produkte (z.B. Fahrzeuge); allerdings mit weiteren beratenden und informierenden Funktionen.

Der Auftraggeber erhofft sich dadurch, Hemmungen von Kunden, die wenig technische Expertise besitzen, abzubauen. So soll der Kunde darüber aufgeklärt werden, dass Mehrwerte meist nur in Kombination einzelner Services & Partnerschaften im Bereich Smart Home erreichbar sind. Aufgrund bislang zumeist proprietärer Lösungen am Markt konnten für den Endkunden bislang meist keine konkreten Mehrwerte geschaffen werden.

Der Advisor soll seinen Beitrag zu dem übergeordneten Ziel, eine starke Marktdurchdringung von Smart Home Produkten und Services zu erreichen, leisten.

Im Sommersemester 2016 wurde in einem Modul am Lehrstuhl Agententechnologien in betrieblichen Anwendungen und der Telekommunikation somit ein Proof-of-Concept für einen Advisor als Web-App entwickelt (Advisor 1.0).

Ziel war es nun, das Konzept und den vorhandenen Prototypen weiterzuentwickeln, aufdass die Vision des Connected Living e.V. erfüllt werde.

### **3.1 Abgrenzung**

Mit dem Advisor soll Kunden die Hemmschwelle zum SmartHome bzw. dessen Produkten abgebaut werden. Es ist jedoch nicht das Ziel, dass der Kunde die vorgeschlagenen Produkte direkt im Advisor kaufen kann. Dafür gibt es eine schier unbegrenzte Anzahl von Händlern oder Online Shops.

Über die vorgeschlagenen Szenarien erhält der Kunde Anwendungsmöglichkeiten die er in seinem Zuhause umsetzen kann. Auch hier unterstützt der Advisor nicht mit automatischen oder manuellen Anleitung zur Umsetzung der Szenarien.

### **3.2 Anforderungen**

Nach der Durchsicht der Ergebnisse aus dem Vorsemester wurde klar, dass einige grundlegende Elemente im Konzept und der Prototyp überarbeitet werden mussten. Neben der sehr beschränkten Dynamik im Vorschlagsprozess, war insbesondere der Prototyp kein Advisor im Sinne des Projektpartners. Als Anforderungen wurden nachfolgende Kriterien extrahiert, weiterentwickelt und von Connected Living definiert und folglich in diesem Projekt erfolgreich umgesetzt.

Die Muss-Kriterien wurden gemeinsam zu Beginn des Projekts mit dem Auftraggeber und dem Lehrstuhl abgestimmt. Hierbei spielten organisatorische und technische Aspekte eine Rolle. Zudem wurde einige Anforderungen aus dem Vorsemester nicht mehr weiterverfolgt, dazu gehörte z.B. ein spezieller Bereich für Hersteller zum einpflegen von Produkten. Im weiteren Verlauf des Projekts wurden wenige Kriterien nachgebessert bzw. an die aktuellen Bedürfnisse des Auftraggebers angepasst.

#### **3.2.1 Muss-Kriterien**

Wir haben die folgenden funktionalen und nicht-funktionalen Anforderungen identifiziert:

#### **Funktionale Anforderungen**

- Der Nutzer soll in vier einfachen Schritten zu seinem Smart Home geführt werden

- Der Advisor ist als Web-App in Form einer Single Page Application zu entwerfen
- Anhand von Nutzereingaben sollen passende Produkte gefunden werden
- der Advisor soll Informationen über die gefundenen Produkte anzeigen
- Der Advisor soll einen Konfiguratorcharakter haben, der jedoch zusätzliche beratende Fähigkeiten anbietet.
- Der Advisor soll Szenarien oder Anwendungsfälle verstehen können
  - Dabei werden dem Kunden Anwendungsfälle im Smart Home präsentiert, die er anzeigen und auswählen kann. Entsprechend der Auswahl der Anwendungsfälle wird die optimalste Auswahl von Produkten über ein dynamisches Matching dem Kunden angeboten. Dieser erhält die Möglichkeit, die Produktauswahl dann zu kaufen um die gewählten Anwendungsfälle umzusetzen.
- Während des Beratungsprozess sollen dem Benutzer die Gerätetypen der gefundenen Produkte in eine anschaulichen Art und Weise präsentiert werden.

### Nicht-Funktionale Anforderungen

- **Zuverlässigkeit:** Advisor sollte immer funktionsbereit und nutzbar sein
- **Wartbarkeit:** das stetige Einpflegen neuer Produkte und Szenarien bzw. bestehende Einträge können jederzeit geändert werden
- **Effizienz:** Kunde sollte möglichst wenig Schritte durchlaufen/Klicks tätigen müssen und schnellstmöglich das Beratungsergebnis erhalten
- **Übertragbarkeit:** Der Advisor sollte sich möglichst leicht in die Connected Living Umgebung integrieren lassen.
- **Immersive User Experience:** Der Nutzer sollte eine eizigartige und ungewöhnliche Erfahrung auf der Seite haben. Der Nutzer soll nicht mit Informationen überladen werden. Der Nutzer braucht nur relevante Fragen zu beantworten, dessen Auswirkung direkt auf seine Ergebnisse nachvollziehbar ersichtlich sind.

### **3.2.2 Kann-Kriterien**

- Ein Nutzer soll die Möglichkeit haben, sein Warenkorb in einem Kundenkonto zu speichern
- Um dem Kunden ein noch besseres Verständnis von Möglichkeiten und Anwendungen von Produkten im SmartHome zu bieten, kann der Kunde über einen 3D-Viewer seine Wohnsituation individuell nachbilden. Anhand dessen kann der Kunde ein virtuelles SmartHome kreieren.
- Darauf aufbauend könnte daraus ein Verweis zu Onlineshops erstellt werden bzw. zu den Hersellerwebseiten verlinken

### **3.2.3 Fertigstellungskriterien**

- Der Code ist ausführbar und in das Repository eingespielt.
- Alle Akzeptanzkriterien erfüllt
- Tests erfolgreich durchgeführt

## **3.3 Ideeentwicklung**

Die Entwicklung der Ideen erfolgte in den ersten Wochen des Projekts und orientierten sich an den Anforderungen des Auftraggebers. Obwohl das Team am Anfang mit einer ziemlich ähnlichen Einstellung in das Projekt zusammenkam, war es nicht einfach die Konkretisierung der Features in den ersten Wochen auf einen gemeinsamen Nenner zu bringen.

Nach der Aufteilung in einer Frontend- und einer Backendgruppe haben wir eine Art pflichtenheftmäßige Konkretisierung geführt (siehe im Projekt `design.tex`). Mit entsprechenden Diagrammen und Mock-up Designs, waren die wöchentlichen Team- und Gruppentreffen eine Basis für Brainstorming, Auswahl und Filterung. Zu den anfänglichen Vorschlägen zählten Technologien (z.B. Verwendung von OpenGL), sowie Konzepte (z.B. Information Retrieval, Usability). Aus dem DAI-Labor gab es Hilfestellung in Form eines Leitfadens über User-centered Design / User-driven experience.

Es mussten Entscheidungen getroffen werden, wie der Umfang der Fragen, die dem Nutzer anfänglich gestellt werden, in welcher Reihenfolge sie gestellt werden, welche Grundannahmen wir über den Nutzer treffen bzw. wer überhaupt unsere Nutzer sind. Wichtig dabei war, Flexibilität und Skalierbarkeit beizubehalten. Dies führte des Öfteren zu kontroversen Diskussionen über die genannten Fragen bzw. welche Ansätze man zur Lösung weiterer Probleme verwenden sollte. Einschränkungen waren meistens der Umfang des Projekts sowie die Zeit und manchmal das fehlende Know-How. Demzufolge war eine Einteilung in Kann- und Muss-Kriterien essenziell und nicht selten intern eine gute Verhandlungsbasis. Dazu war es auch wichtig zu ermitteln, welches Teammitglied welche Fähigkeit mit sich bringt und wer welche Lernziele hat. Für die Zusammenarbeit war auch eine fließende Kommunikation sowie ein demokratisches Wesen notwendig. Die größte Herausforderung war das Umgehen mit der Entwicklung einer Idee, die noch nicht soweit auf dem Markt ausgereift ist. Da das Projekt auch keine Vorläufer hat, war es einerseits manchmal demotivierend, sich aus der Umgebung keine Vergleiche ziehen zu können. Andererseits sehen wir die Stärke unseres Teams darin daraus gelernt zu haben, die Flexibilität unseres Designs als Chance zu betrachten.

### 3.4 Inspirationen

**IOLite** - für die Visualisierung <http://iolite.de>

**QIVICON** - für die Schritte [http://www.smarthomeberater.info/advisor/use\\_case\\_interests](http://www.smarthomeberater.info/advisor/use_case_interests)

**Mehrere Fluggesellschaften** - für die Preisanpassung bei der Produktauswahl - Bsp.: <http://www.klm.de>

**Live Beispiele aus JavaScript Libraries** - z.B. SnapSVG, DataDriven-Documents <https://d3js.org> - <http://snapsvg.io>

## 4 Produkteinsatz

### 4.1 Zielgruppen

Das Thema SmartHome nimmt immer mehr an Popularität zu und damit die Vernetzung von Haustechnik und -geräten. Ob Licht, Heizung, Belüftung, Waschmaschine oder Fernseher: alles kann via Internet kontrolliert und gesteuert werden – sei es mobil per Smartphone und Tablet oder zu Hause mit dem Computer.

Anbieter von SmartHome-Technologien bauen auf großes Nachfragepotential – und das zu Recht, denn den Ergebnissen des W3B Report “SmartHome – Connected Home” zufolge, ist jeder zweite Online-Nutzer an deren Einsatz interessiert. Aufbauend auf diesem Potenzial kann auch der SmartHome Advisor vom großen Interesse bei den Nutzern sein, unabhängig von ihrem Beruf, Alter und früheren Erfahrungen mit SmartHome Technologien. Deswegen wurde unser Advisor als ein möglicher Ansatz gesehen, die Anbieter bzw. Hersteller von SmartHome Technologien und die Endnutzer zusammen zu bringen.

DONAALDA soll jedem Nutzer ermöglichen in die spannende Welt des SmartHomes einzusteigen. Dabei kann der Nutzer während der Auswahl sich eine Liste von Szenarien heraussuchen, sich ein interaktives SmartHome zusammenstellen und dieses Thema interaktiv erleben. Durch drei Grundfragen können personalisierte Benutzungsszenarien vorgeschlagen werden. Deswegen spielt es keine Rolle, ob der Nutzer ein Kind, ein erfahrener Man mit der technischer Ausbildung oder ein Senior ist.

Auf der anderen Seite soll DONAALDA Herstellern ermöglichen, ihre Produkte szenarienbasiert einzupflegen. Wichtig dabei sind ausführliche Beschreibungen und technische Charakteristiken der eingepflegten Produkte. Anhand von gesammelten Auswahl- und Kaufstatistiken der Nutzer solle den Herstellern die Trends aktueller Nachfrage vermittelt und somit eine gezielte Produktbetreuung und Verkauf ermöglicht werden. Eine direkte Kommunikation von Nutzern und Herstellern innerhalb des Advisors Community bringt nur Vorteile für diese beiden Zielgruppen von DONAALDA.

#### 4.1.1 Anwendungsbenutzer

Bei der Entwicklung von DONAALDA steht der Benutzer im Mittelpunkt. Die Plattform sollte dem Nutzer ein erlebnisreiches und informatives Eintauchen in die SmartHome Welt ermöglichen. Zunehmend löst sich dabei die Trennung von Produkten und Dienstleistungen in abgeschlossene Lebensbereiche auf. Für die Endnutzer entfalten sich so neue Dimensionen von Komfort, Gesundheit, Energie und Sicherheit. Die Hauptziele davon sind das Leben des Benutzers qualitativ zu verbessern und statt alltäglichen Aufgaben zuhause die Zeit für wichtigere Dinge nutzen zu können.

Hier ist nur kleiner Ausschnitt der SmartHome Lösungen, die Benutzer auf der DONAALDA Plattform interessieren könnten:

- von der Couch aus im Wohnzimmer die Heizung höher drehen,
- aus dem Zug kontrollieren, ob Kaffeemaschine und Licht auch wirklich ausgeschaltet wurden, oder
- sich mal genauer anschauen, welche Geräte zu welcher Tageszeit eigentlich wie viel Strom verbrauchen.

Möglich macht dies moderne SmartHome Technologie, mit der Geräte im Haushalt ferngesteuert und automatisiert werden können. Die User Experience von SmartHome Systemen beginnt aber bereits vor der eigentlichen Nutzung, ja sogar vor dem Kauf: mit dem interaktiven SmartHome Advisor wurde der wesentliche Grundstein dafür gelegt, welche Gefühle mit den SmartHome Produkten zusammen verkauft und für wie hochwertig und kompliziert diese empfunden werden können. Aus Kundensicht ist, neben Geschäften in denen die Technik ausprobiert werden kann, die Online-Recherche ein essentieller Weg die Möglichkeiten eines SmartHomes näher kennenzulernen. Einer der ersten wichtigen Schritte der Customer-Journey von SmartHome Systemen ist die Recherche nach dem passenden Anbieter bzw. nach dem passenden System.

In der Recherche phase können daher folgende Bewertungsmaßstäbe für die Nutzer als wichtig angesehen werden:

- Finde ich konkrete Anwendungen und Situationen, in denen das SmartHome System mir einen Mehrwert für meinen Komfort gibt?
- Wird übersichtlich dargelegt, welchen Funktionsumfang das Produkt hat?

- Wird klar, wie kompliziert das System zu installieren ist? Kann ich die Kosten, die entstehen, nachvollziehen? Sind diese einmalig oder monatlich?
- Was ist mit dem Thema „Datensicherheit“? Wird dies transparent gemacht?
- Ist die vorhandene Technik in meinem Haus kompatibel? (z. B. die Heizung, die Lichtelektronik oder die Fenster?)
- Kann ich die Produkte verschiedener Hersteller bei dem Kauf kombinieren und später erfolgreich installieren?

DONAALDA's potentielle Zielgruppen unterscheiden sich deutlich hinsichtlich ihres technischen Know-Hows, ihres Budgets und den bereits zu Hause vorhandenen Geräten. Zudem unterscheiden sich die Bedürfnisse deutlich voneinander, die durch SmartHome Systeme befriedigt werden können. Darauf basierend haben sich überwiegend bei der Entwicklung des Advisors die vier Bedürfnisse Sicherheit, Komfort, Gesundheit und Energiesparen etabliert.

#### 4.1.2 Community

Durch die Integration der so genannten online Benutzercommunity kann DONAALDA als SmartHome Advisor verbessert werden. Das soll als ein extra Bereich in der Anwendung gesehen werden, wo die Nutzer bei verschiedenen Fragen oder Problemen per Chat oder FAQ-Interaktion mit einander oder mit den Herstellern kommunizieren können.

Interessiert sich der Nutzer einmal für ein bestimmtes Produktpaket etwas näher oder wird es wieder etwas technischer und komplexer, dann kann er immer eine Frage an die Community stellen. Zum einen taucht nun das Thema Kompatibilität für den Benutzer auf und dies ist auf unterschiedliche Art und Weise zu verstehen:

- Ist sein Haus bereit für SmartHome? (z. B. die Heizung, die Fenster, die Beleuchtung)
- Kann er das SmartHome mit seinem aktuellen Smartphone steuern oder braucht er ein anderes Gerät oder gar eine separate Fernbedienung?
- Hat er teilweise vielleicht schon Technik, die er verwenden kann, wie z. B. seinen WLAN-Router?

- Kann er die Produkte selbst installieren oder benötigt er dafür einen Techniker?
- Wie sicher ist ein SmartHome?

Auch bei diesen Fragen gilt, dass er in seiner Situation abgeholt und in die Lage gebracht werden muss, die neuen Produkte auf seinen persönlichen Kontext beziehen zu können. In den Punkten Datenschutz bzw. Datensicherheit könnten die DONAALDA-Benutzer auch einige Bedenken haben, auch wenn das Thema für sie persönlich nicht sehr interessant ist. Um etwaige Bedenken der Nutzer bezüglich Datenschutz und Privatssphäre auszuräumen können die Benutzer Informationen Hinweise und Erlebnisse austauschen. Dabei übernehmen Hersteller eher die Rolle von Unterstützern und Informanten, als von Verkäufern ihrer Produkte, und etablieren sich dadurch für die Nutzer als zuverlässige Denstleister, denen man vertrauen kann.

Später soll es den Benutzern der DONAALDA-Plattform möglich sein eigene Szenarien zu erstellen und diese mithilfe von vorhandenen SmartHome Geräten zu verwirklichen. Dadurch entsteht eine großartige Möglichkeit für die Hersteller, direkt auf die entstandenen Kundenbedürfnisse zu reagieren und ihr eigenes Angebot an Geräten und Dienstleistungen zu verbessern. Durch Etablierung dieser Community innerhalb der DONAALDA-Plattform haben Nutzer und die Hersteller direkten Draht zu einander und profitieren beidseitig.

#### **4.1.3 Produkthersteller**

Wenngleich auch schon über die Rolle der Hersteller innerhalb der Advisor-Plattform gesprochen wurde, muss auch definiert werden mit welcher Motivation und welchen Ergebnissen eine solche Kooperation mit Produktherstellern angefangen werden kann. Zu den aktuell beliebtesten Themen, die im SmartHome Welt immer mehr an Bedeutung gewinnen, gehören Sicherheit, Gesundheit, Energiesparen und Komfort. Aber auch Bereiche wie Versicherungen, Service und Wartung, erweiterte Garantieleistungen sowie Ambient Assisted Living rücken mehr und mehr in den Fokus des Interesses. Zudem, wie schon oben erwähnt, erweisen sich viele online Angebote auf den anderen SmartHome Portalen als wenig flexibel und erfüllen hinsichtlich Vertrieb, Preis und Technik nicht die Erwartungen der Nutzer. Auch hier offenbart sich eine Bedarfslücke, die von Sicherheits- sowie Telekommunikationsunternehmen, Energieversorgern oder Versicherungen geschlossen werden könnte. Allerdings existiert noch keine Lösung auf dem Markt, die die Bedürfnisse von Verbrauchern zu hundert Prozent erfüllt. Die DONAALDA Platform kann einige dieser Kundenseitigen Unsicherheiten durch direkten Kontakt zwischen Kunden und Herstellern beseitigen.

## 5 Aufbau des Beratungsprozesses

Der Beratungsprozess der in DONAALDA implementiert wird, besteht im Groben aus 4 verschiedenen Phasen.

1. Die Onboardingphase in der grundlegende Benutzerpräferenzen abgefragt werden, die den Benutzer durch den restlichen Beratungsprozess begleiten.
2. Die Szenarioauswahlphase in der der Benutzer basierend auf seinen Präferenzen Funktionalitätspakete (s.g. Szenarien) vorgeschlagen bekommt und sich so sein persönliches Szenarioportfolio zusammenstellen kann.
3. Die Produktauswahlphase, in welcher der Benutzer die Produkte, die sein Szenarioportfolio implementieren, genauer spezifizieren kann, indem vorgeschlagene Produkte durch kompatible Alternativen ersetzt werden.
4. Eine finale Übersichtsseite, die der Benutzer ausdrucken und zu einem Fachmann bringen kann, der ihn bei der Beschaffung, Installation und Einrichtung der Produkte unterstützen kann.

Jede dieser Phasen wird im Folgenden genauer beschrieben werden. In einem normalen Interaktionsverlauf durchläuft der Benutzer jede Phase einmal. Es ist allerdings auch möglich zu einer vorherigen Phase zurückzukehren, um etwas zu ändern.

### 5.1 Onboarding

Das Ziel der Onboardingphase ist es, grundlegende Informationen darüber zu sammeln, was dem Benutzer besonders wichtig ist, um in den nachfolgenden Phasen relevante Vorschläge für Szenarien und Produkte zu liefern. Bei der Auswahl der Fragen war es uns wichtig, den Benutzer nicht durch einen endlos langwierigen Fragenkatalog abzuschrecken und trotzdem genügend Informationen zu erhalten, um für den Nutzer sinnvolle Vorschläge zu geben. Speziell wird nach folgenden Dingen gefragt:

**Kategorienpräferenz** Szenarien in DONAALDA sind in vier Kategorien (Sicherheit, Gesundheit, Komfort und Energie) unterteilt (siehe auch Sektion 6). Der Benutzer bewertet jede Kategorie mit einer persönlichen Bewertung von 1 bis 10, wobei 10 bedeutet, dass die Kategorie dem Nutzer besonders wichtig ist. Damit kann für jedes Szenario geprüft werden, wie gut es auf die Benutzerpräferenzen passt.

**Produktpräferenz** Wenn DONAALDA Produktmengen sucht, die eine gegebene Menge von Szenarios implementieren, gibt es unter Umständen mehrere mögliche Implementierungen. In so einem Fall wird dann die optimale Produktmenge bezüglich Preis, Energieverbrauch oder Erweiterbarkeit ausgewählt. Welches Kriterium verwendet wird, wird durch die Produktpräferenz festgelegt.

**Renovierungspräferenz** Die (vielleicht unglücklich benannte) Renovierungspräferenz gibt an, ob Produkte, deren Installation destruktive Maßnahmen wie das Verlegen von Kabeln durch Wände erfordert, akzeptabel für den Benutzer sind. Wählt der Benutzer hier „Nein“, so werden im folgenden Beratungsprozess nur Produkte verwendet, die ohne solche Maßnahmen installierbar sind.

Andere Fragen wie nach der Anzahl der Räume im Haus wurden während der Entwicklung auch diskutiert, aber dann nach Festlegung der Ziele des Projekts abgelehnt. Ebenso wurde eine Frage, ob Kinder Teil des Haushalts seien, abgelehnt, weil sie zu spezifisch und somit die „Kosten“ einer zusätzlichen Frage nicht wert war.

Nachdem der Benutzer die drei Grundfragen beantwortet hat, bekommt er eine Übersicht über seine Antworten. Wenn er damit zufrieden ist, kann er zur Szenarioauswahlphase voranschreiten.

## 5.2 Szenarioauswahl

In der Szenarioauswahlphase werden dem Benutzer basierend auf seinen Präferenzen bis zu sechs Szenarien vorgeschlagen. Zu jedem dieser Szenarien wird zusätzlich (in einem aufklappbaren Fenster) eine Beschreibung, die enthaltenen Produkttypen sowie die Passgenaugigkeit des Szenarios auf die Benutzerpräferenz angezeigt. Der Benutzer kann ein Szenario zu seinem Szenario-„Warenkorb“ hinzufügen, worauf die Vorschläge neu berechnet werden, um die Kompatibilität zwischen den Szenarien im Warenkorb und den vorgeschlagenen Szenarien sicherzustellen.

Die Vorschläge enthalten auch den voraussichtlichen Preis, der beim Hinzufügen des Szenarios anfiele. Da Szenarien nicht an bestimmte Produkte gebunden sind, sondern dynamisch implementiert werden, ist auch der angezeigte Preis nicht fest. Abhängig vom Inhalt des Warenkorbs und der Benutzerpräferenz können für das selbe Szenario unterschiedliche Preise berechnet werden. So kann es zum Beispiel sein, dass teurere Produkte gewählt werden, wenn der Benutzer mehr Wert auf einen niedrigen Energieverbrauch, als auf den günstigen Preis legt. Genauso kann es sein, dass ein Basisgerät bereits von einem Szenario im Warenkorb verwendet wird und deshalb nicht zum Preis mitzählt.

Der Benutzer kann während der Szenarioauswahl seine Onboardingpräferenzen nachträglich anpassen und die vorgeschlagenen Szenarien mit zwei Filtern weiter einschränken. Diese sind:

**Subkategorienfilter** Zusätzlich zu den Kategorien sind Szenarien in Subkategorien unterteilt. Anders als bei Kategorien handelt es sich dabei aber um eine einfache  $n$  zu  $n$  Beziehung. Ein Szenario ist entweder in einer Subkategorie oder nicht (siehe Sektion 6). Werden Subkategorien im Filter angegeben, werden nur noch Szenarien vorgeschlagen, die in mindestens einem der angegebenen Subkategorien sind.

**Produkttypfilter** Der Benutzer kann auch nach Szenarien filtern, die einen speziellen Produkttyp (z.B. Waschmaschine) verwenden. Szenarien sind an Features und nicht Produkttypen gebunden und nur durch implementierende Produkte werden Produkttypen zugeordnet. Es wird der Produkttypfilter bei dem Szenario mitgeschickt, um sicher zu stellen, dass jede Implementierung dieses Szenario immer diesem Filter genügt.

Wenn der Benutzer mindestens ein Szenario ausgewählt hat, kann er in die Produktauswahlphase übergehen.

### 5.3 Produktauswahl

Da DONAALDA schon in der Szenarioauswahlphase zur Sicherstellung der Kompatibilität Produktmengen sucht, die den Warenkorb implementieren, kann dem Benutzer direkt eine gültige Lösung angezeigt werden. Diese ist allerdings noch formbar und könnte sich unter Umständen noch komplett ändern. Der Benutzer hat daher zunächst die Möglichkeit, ein paar der angebotenen Produkte festzulegen, was bedeutet, dass sie im weiteren Prozess nicht durch andere Produkte ersetzt werden werden.

Der Benutzer kann sich auch zu jedem der angezeigten Produkte, wenn vorhanden, Alternativen anzeigen lassen. Wählt er dann eine dieser Alternativen aus, wird die durch das Produkt bestimmte Funktionalität auf das Alternative Produkt festgelegt und die Produktmenge wird neu berechnet. Dabei können sich unter Umständen alle nicht festgelegten Produkte ändern. Dies ist erwünscht, da das eine größere Flexibilität ermöglicht. Ansonsten wäre der Benutzer auf Produkte beschränkt, die mit der zuerst vorgeschlagenen Basisstation kompatibel sind. Da der Benutzer nur wenig Einfluss auf die von ersten Vorschlägen von DONAALDA hat, wäre dies nicht akzeptabel.

Wechselt der Benutzer aus dieser oder der Übersichtsphase wieder zurück zur Szenarioauswahl, so werden die festgelegten Produkte verworfen. Der Grund dafür ist, dass sich mit der Szenariomenge auch die Funktionalitäten, die von einzelnen Produkten implementiert werden müssen, ändern. Die alten Produktzuordnungen wären dann nicht mehr akkurat.<sup>1</sup>

Ist der Benutzer zufrieden mit seiner Auswahl kann er zur Übersichtsphase weitergehen.

## 5.4 Übersicht

In der finalen Übersichtsphase bekommt der Benutzer eine Zusammenfassung seiner ausgewählten Produkte mit jeweiliger Beschreibung, zugehöriger Szenarien sowie einer Preisübersicht. Mit dieser Übersicht soll der Benutzer eine Vorstellung einer kompatiblen Produktzusammensetzung für seine gewünschten Szenarien bekommen. Nicht eingeschlossen sind weder Installations- noch weitere Beratungskosten. Außerdem sind die Preise grobe Richtwerte. Des Weiteren empfiehlt es sich, die finale Kompatibilität von einem menschlichen Berater oder expliziten Empfehlungen von Firmen prüfen zu lassen.

---

<sup>1</sup>Zudem hätte das die Implementierung des Backends deutlich komplizierter gemacht.

# 6 Datenbank und Modellierung

Nachfolgend wird das aktuelle Modell und die wichtigsten Entitäten näher erläutert.

## 6.1 Allgemein

Zu Projektstart haben wir natürlich das gesamte Repository inklusive des Models übernommen. Im Verlauf des Projektes wurde das Modell evolutio-när an die geplanten Features angepasst und parallel Altlasten entfernt. Das aktuelle Model wird in der nachfolgenden Tabelle 1 genauer dargestellt und erläutert. Allerdings müssen vorab die für das nachfolgende Kapitel Matching 7 besonders relevanten Entitäten und Beziehungen genauer erläutert werden.

### 6.1.1 Features

Features sind die tragenden Elemente des Matchings. Jedes Feature stellt eine elementare Funktionalität dar. Hierbei kann es sich um Funktionalitäten wie „Video aufzeichnen“ oder in „bestimmten Farben leuchten“ handeln. Durch die Darstellung dieser elementaren Aktionen als Feature lassen sich Produkte exakt beschreiben. Denn Produkte implementieren eins oder mehrere Features, welche dann wiederum von Szenarien verwendet werden können, um die benötigte Funktionalität zu beschreiben.

Da Features extrem vielfältig sein können, hilft es, wenn man diese in Featurekategorien einteilt. So kann man Features sinnvoll nach Einsatzzweck kategorisieren und aber auch die vom Eintragenden intendierte Funktionalität vom Namen herleiten. Vorerst haben wir eine Unterteilung aller von uns ver-wendeten Features in „controls“, „notifies“, „operates“, „records“ und „senses“ vorgenommen. Diese sind aber frei nach neu hinzustößenden Funktionalitäten erweiterbar. Grundsätzlich dient diese Unterteilung in Featurekategorien auch nur der Übersichtlichkeit und Verständlichkeit. Beispiele für korrekt definierte Features sind: „notifies\_sendSMS“, „operates\_Licht“, „records\_Video“, „senses\_Luft\_Feuchtigkeit“, ...,

### **6.1.2 MetaDevice**

MetaDevices sind Objekte die eine Featuremenge beschreiben. Sie dienen als Abstraktion von tatsächlich existierenden Produkten, denn um Szenarien korrekt beschreiben zu können, benötigt man keine konkreten Produkte, sondern nur gewisse Funktionalitäten von den verwendeten Geräten. So kann man für Szenarien fein granulare Bedingungen an Produkte formulieren und diese dann als MetaDevice speichern. Produkte sind dann tatsächliche Implementationen von der durch die MetaDevices beschriebenen Featuremengen, denn beide Begriffe werden durch ihre zugehörigen Features definiert. Da nun Szenarien MetaDevices erfordern, können konkrete Produkte, welche die gleichen oder mehr Features implementieren, nun auf diese Objekte zugewiesen werden. So ermöglicht man dem Interessenten die Wahl zwischen mehreren Alternativen aus der gesamten Datebank, anstatt manuelle Alternativen konfigurieren zu müssen.

### **6.1.3 Broker**

Broker sind MetaDevices, wie im Kapitel 6.1.2 beschrieben, welche die Kommunikation zwischen Produkten entweder steuern oder zumindest weiterleiten können. Sie können allerdings auch, wie jedes andere MetaDevice auch, tatsächliche Features anbieten. Broker unterstützen mindestens ein Protokoll im Leadermodus. Konkret implementierte Beispiele sind alle Basisstationen wie das „Miele@Home Gateway“ oder die „IOLITE Box“.

### **6.1.4 Endpoint**

Hierbei handelt es sich um MetaDevices, wie in Kapitel 6.1.2 beschrieben, welche lediglich ihren Featurekatalog anbieten, allerdings nicht als Kommunikationsswitch agieren können.

### **6.1.5 Scenario**

Ein Szenario ist eine konkrete Ausprägung einer klar definierten Smart-Home Funktionalität. Szenarien sind extrem kompakt und genau definiert. Ein Beispiel wäre „Wenn mein Wecker klingelt, soll die Kaffemaschine angeschaltet werden.“. Dieses Szenario besteht aus mehreren MetaDevices:

- Einen HCC (Home Control Center), welches die Kommunikation steuern kann
- Einem Wecker, welcher ein Signal an das HCC senden kann, dass die Weckzeit erreicht ist
- Einer Kaffeemaschine, die ein Steuersignal empfangen kann, dass es Zeit ist den Kaffeekochprozess zu starten

Dieses Szenario repräsentiert auch den gewünschten Komplexitätsgrad an Szenarien im Allgemeinen gut, denn nur durch klar erläuterten Zweck lassen sich klare Einsatzgebiete definieren und exakt abbilden. Szenarien bestehen aus MetaDevices, welche die für die Teilfunktionalität benötigten Features implementieren. Ein Szenario besteht immer aus einem Metabroker und mindestens einen Metaendpoint. So lassen sich alle erdenklichen Szenarien strukturiert und gut verständlich darstellen. Ein weiterer Vorteil dieses Vorgehens ist, dass noch keine konkreten Produkte eingebunden werden müssen. So ergeben sich die konkreten Produkte während des Beratungsprozesses direkt aus den in der Datenbank aktuell hinterlegten Optionen.

### 6.1.6 Rating

Hiermit soll die Zugehörigkeit von Szenarien zu den Kategorien beschrieben werden. Szenarien können unterschiedlich starke Zugehörigkeitsausprägungen zu den unterschiedlichen Kategorien haben, aber haben einen Wert zwischen 0 und 10. Diese Zugehörigkeit ist notwendig, damit bei der Suche nach vorzuschlagenden Szenarien die Nutzerpräferenzen korrekt, zumindest grob, einbezogen werden können.

### 6.1.7 Model

Entität/Beziehung	Attribut	Definition
Category	name picture backgroundPicture short description	Name der Kategorie Repräsentatives Bild, welches die Kategorie darstellt Hintergrundbild für die jeweilige Kategorie Beschreibt die Kategorie kurz in bis zu 80 Zeichen

<b>Entität/Beziehung</b>	<b>Attribut</b>	<b>Definition</b>
	- description	Völländige Beschreibung der Kategorie
	- iconString	Iconname des verwendeten Frontend Icons für die Kategorie
Scenario		
	- name	Eindeutiger einzigartiger Name des Szenarios
	- url name	Legacy, da nun keine eigenen Szenarioseiten mehr existieren
	- description	Völlständige Beschreibung des angebotenen Szenarios
	- picture	Bild welches das Szenario repräsentativ darstellt
	- provider	Szenarien werden von unterschiedlichen Entitäten bereitgestellt, dies referenziert den jeweiligen Verantwortlichen, der dieses Szenario erstellt hat
	- categories	Referenziert die Zugehörigkeit des Szenarios zu den unterschiedlichen Kategorien mit Wertung zwischen 0 und 10
	- meta broker	Repräsentiert den MetaBroker, welcher die für dieses Szenario benötigten Features für einen Broker bereitstellt
	- meta endpoints	Stellt die Wahl aller für dieses Szenario benötigten Meta-Devices dar
	- subcategory	Gibt die genaue Zugehörigkeit zu einer der SubCategories an
	- in shopping basket of	Legacy, da zwischenzeitig mit Sessions gearbeitet wurde, war es nötig zu speichern in welchen Sessions welche Szenarien vorhanden sind

<b>Entität/Beziehung</b>	<b>Attribut</b>	<b>Definition</b>
	thumbnail	Automatisch errechnetes Thumbnails des Attributes picture
	title	Kurzer Titel des Szenarios zur Darstellung im Frontend
ScenarioCategoryRating		
	scenario	Das zuzuordnende Szenario
	category	Die zugehörige Kategorie zu dem jeweiligen Szenario
	rating	Die Passgenauigkeit des Szenarios zu der Kategorie, dies ist für das ScenarioMatching 7.2
Subcategory		
	name	Name der Subkategorie
	url name	Legacy, Subkategorien hatten ursprünglich eigene Landing Pages
	short description	Kurzbeschreibung der Subkategorie (z.B. Worum geht es in Schutz vor Wasser und Feuer)
	picture	Bild welches die Subkategorie am besten darstellt
	belongs to category	Die Zugehörigkeit zu einer bestimmten Kategorie
	used as filter by	Legacy, während einer kurzen Sessionbasierten Matchingphase mussten natürlich auch die bereits ausgewählten Filter einer Session zugewiesen werden
SubcategoryDescription		
	belongs to subcategory	Referenziert die Subkategorie, zu der die Beschreibung gehört
	description	Volltext der die Subkategorie beschreibt
	image	Bild das die Subkategorie beschreibt

Entität/Beziehung	Attribut	Definition
	thumbnail	Automatisch errechnetes thumbnail aus dem attribut image
	left right	Ursprünglich als formatierungsattribut fürs Frontend gedacht, sollte das Bild entweder Rechts- oder Linksbündig darstellen
	order	Stellt sicher, dass SubcategoryDescriptions zur selben Subcategory in einer bestimmten Reihenfolge dargestellt werden (Sortierung nach aufsteigenden Int Werten)
Product		
	name	Name des Produkts, frei wählbar und Dopplungen möglich
	provider	Hersteller dieses Produkts, bezieht sich auf die in der Datenbank vorhanden Provider entitäten
	product type	Referenziert den Produkttyp zu dem dieses Produkt zählt (Lediglich relevant für die im Haus angezeigten Produktarten)
	used as product type filter by	Legacy da nicht mehr mit Sessions gearbeitet wird, aber wurde zu dem damaligen Zeitpunkt verwendet um die von jeder Session verwendeten Filter zu referenzieren
	svg id	ID der SVG der Hausgrafik, die zu dem ProductType gehört
	icon	Icon welches zu dem ProductType gehört
Provider		

<b>Entität/Beziehung</b>	<b>Attribut</b>	<b>Definition</b>
ProviderProfile	name	Name des Providers
	is visible	Legacy, im ursprünglichen Projekt hatten Provider ihre eigene Landing Page, welche sie selber gestalten konnten und manuell sichtbar/unsichtbar setzen konnten
	public name	Ursprünglich ein Attribut der Entität ProviderProfile, aber mit Löschung dieser zu Provider portiert. Provider Namen beinhalten auch die Rechtsform der Organisation, welche in der url unnötig ist, daher hatten diese einen separaten Profil Namen
Employee	logo image	Entstammt auch dem ProviderProfile und sollte dem Provider eine Möglichkeit geben ein eigenes Logo zu präsentieren. Wird nun verwendet um auf der Indexpage die unterschiedlichen Provider anzulegen
	employer	Der zugehöriger Provider, welcher diesen Employee beschäftigt
UserImage	belongs to user	Welchem Nutzer dieses Bild zugeordnet ist
	image	Das Bild selber
Comment	comment from	Referenziert den Nutzer welchen den Kommentar verfasst hat

Entität/Beziehung	Attribut	Definition
	comment title	Titel des abgegeben Kommentars
	comment content	Eigentlicher Textinhalt des Kommentars
	rating	Die Bewertung die dieser Kommentar abgegeben hat, zwischen 0 und 5
	creation date	Zeitpunkt der Abgabe des Kommentars
	page url	Auf Grund der vorherigen Verwendung der Django Template Render Engine konnte man die Kommentare den automatisch erzeugten URLs direkt problemslos zuweisen - nun allerdings irrelevant geworden
Question		
	description	Beschreibung der Frage zur Erläuterung mit Hilfe des Tooltips
	icon name	Name des Icons für diese Frage - zumeist das Infosymbol
	question text	Der eigentliche Fragetext
	title	Titel welcher zu der Frage im Onboarding angezeigt werden soll, Definiert den Titel des Teilschritts des Sliders der Onboarding Komponente
	answer presentation	Die Art der Frage, also handelt es sich um einen Schieberegler, Checkboxen, Radiobuttons oder ein Dropdownfeld
	order	Die Reihenfolge in der diese Frage im Onboarding präsentiert werden soll
	rating min	Bezieht sich lediglich auf SliderQuestions, repräsentiert den minimalen Wert des Sliders

<b>Entität/Beziehung</b>	<b>Attribut</b>	<b>Definition</b>
	- rating max	Auch dieses Attribut hat nur Relevanz für SliderQuestions, repräsentiert den maximalen Wert des Sliders
Answer	description	Beschreibung der Antwort für den im Frontend angezeigten Tooltipp, soll dem Nutzer diese Antwortmöglichkeit detailliert erläutern
	- belongs to question	Referenziert die Frage zu der diese Antwort gehört, damit diese gebündelt präsentiert werden und eine Relation hergestellt werden kann - auch zur Speicherung der bereits abgegebenen Antworten eines Nutzers relevant
	answer text	Der eigentliche Antworttext, welcher dem Nutzer angezeigt wird - so kurz wie möglich
	icon name	Icon welches zur Repräsentation der Antwort verwendet werden soll oder kann
Protocol	name	Eindeutiger Name des Protokolls, auch bei unterschiedlichen Versionen des selben Protokolls sollte es im Namen beachtet werden - z.B. KNX-3.5.1
Feature	Name	Eindeutiger und vor allem klar erkennlicher Name des Features, damit der Zweck sofort ersichtlich ist
MetaDevice		

<b>Entität/Beziehung</b>	<b>Attribut</b>	<b>Definition</b>
	name	Name des MetaDevices, wird automatisch vom System beim erstellen generiert - abhängig von den ausgewählten Features und ob es sich um einen Broker oder nicht handelt
	is broker	Wahrheitswert ob es sich bei dem Gerät um einen Broker handeln soll oder nicht
	implementation requires	Repräsentiert die Features die für dieses Gerät zwingend erfüllt werden müssen

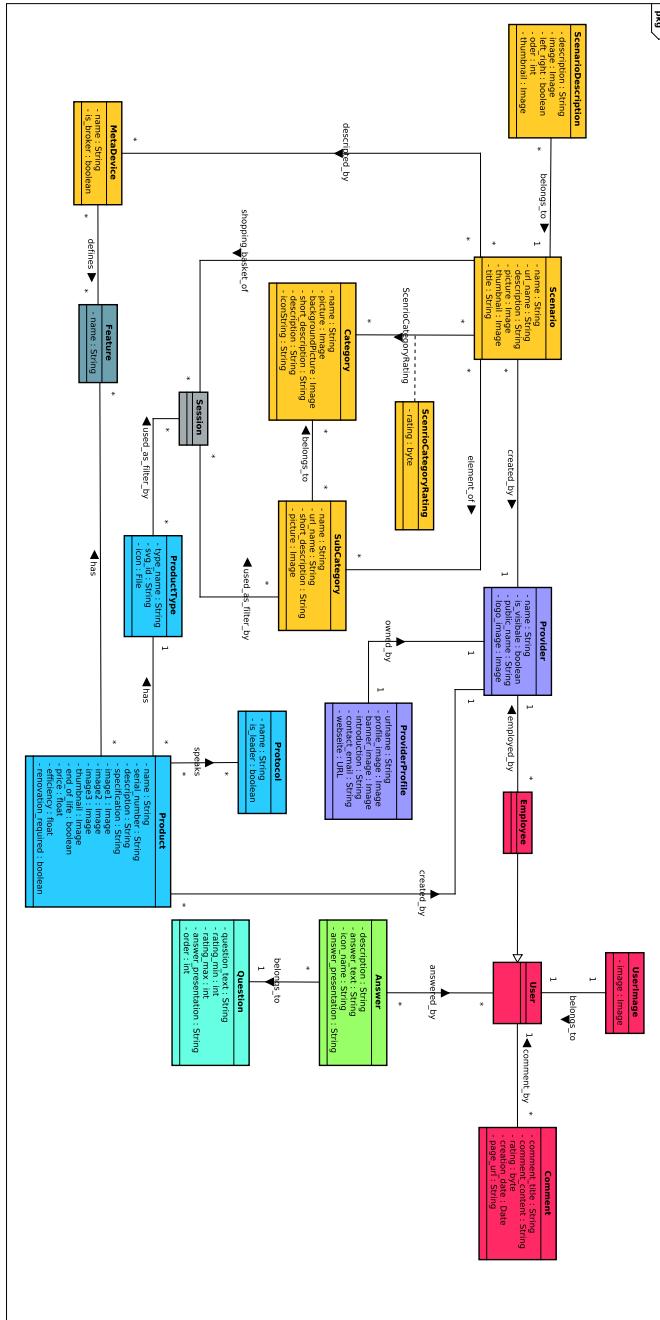


Abbildung 1: Klassendiagramm des gesamten Models

---

```

1  {
2      "scenario_preference": {
3          "Energie": 2,
4          "Komfort": 4,
5          "Gesundheit": 8,
6          "Sicherheit": 10
7      },
8      "product_type_filter": [],
9      "subcategory_filter": [],
10     "shopping_basket": [],
11     "product_preference": "Preis",
12     "renovation_preference": true
13 }
```

---

Abbildung 2: Beispielanfrage an den „Suggestions“ Endpoint.

## 7 Matchmaking

Den Kern des Backends bildet der s.g. Matchmaking Algorithmus. Dieser ist dafür verantwortlich zu einer gegebenen Menge von Szenarien (zueinander kompatible) Produkte zu finden, die die in den Szenarien geforderte Featuremenge bereitstellen. Dies ist nicht nur bei der Produktauswahl relevant sondern wird auch in der Szenarioauswahl verwendet um sicherzustellen, dass keine Inkompatibilitäten entstehen können.

In der Szenarioauswahlphase sendet das Frontend Anfragen an den `/suggestions` API Enpoint, die der Form in Listing 2 entsprechen. Das Backend verarbeitet die Anfrage um dem Benutzer eine sortierte Liste an Szenariovorschlägen zurückzugeben, sodass dieser sein Smart Home konfigurieren und personalisieren kann, sowie mehr über das Potenzial seines Hauses erfährt.

Dazu nimmt es die gesammelten Benutzerpräferenzen und spaltet die Eingaben in zwei Teile auf:

1. Die Angabe zu der Kategoriepräferenz wird benutzt um Szenarien zu sortieren und somit die am besten passenden Szenarien zu finden.
2. Die Produkt- und Renovierungspräferenz wird in der Implementierungsphase des Matchings benutzt, um aus den passenden Produktsets die optimale Lösung für den Benutzer auszuwählen.

Für den ersten Schritt nutzt der Algorithmus die Kategoriebewertungen der Szenarien aus der Datenbank<sup>2</sup>. Für den zweiten Schritt arbeitet er auf Produkten und deren zugehörige Attribute wie Preis, Energieverbrauch, unterstützte Protokolle und ob die Produkte fest im Haus installiert werden müssen oder als Plug-And-Play verwendet werden können. Zusätzlich werden auch die Produkttyp- und Subkategoriefilter aus der Szenarioauswahlphase berücksichtigt.

Neben der `/suggestions` Schnittstelle gibt es noch die `/final_product_list` und `/product_alternatives` Schnittstellen. Sie behandeln das Konfigurieren und Personalisieren des Smart Homes und bieten Informationen über die Produkte an die die Szenarien implementieren. In Sektion 7.4 werden diese Schnittstellen weiter erläutert.

In den nachfolgenden Sktionen werden die einzelnen Phasen des Algorithmus beschrieben und erläutert wie der Benutzer seine Ergebnisse erhält und wie validiert wird, dass das gewählte Szenario auch implementierbar ist. Vorher werden allerdings noch kurz die formalen Anforderungen an den Algorithmus besprochen anhand derer wir ihn ausgelegt haben.

## 7.1 Anforderungen an den Matchmaking Algorithmus

Bevor wir den tatsächlichen Algorithmus entworfen haben, haben wir uns zunächst überlegt, was dieser leisten muss. Allgemein haben wir dazu eine gültige Lösung als eine Menge von Produkten definiert, die einer Reihe von Anforderungen entspricht.

Zunächst ein wenig Notation um das definieren der Anforderungen einfacher zu machen. Wir schreiben  $p \models m$  gdw ein Produkt  $p$  alle features enthält, die von dem Metadevice  $m$  vorgeschrieben werden.  $p_1 \rightarrow p_2$  bedeutet, dass es ein Protokoll gibt, dass von  $p_1$  im Follower Modus und von  $p_2$  im Leader Modus unterstützt wird.  $p_1 \rightarrow p_2$  bedeutet also, dass sich alle Features von  $p_1$  über ein bestimmtes Protokoll von  $p_2$  abfragen und steuern lassen. Da wir definiert haben, dass Bridges (Produkte, die sowohl Protokolle im Leader als auch im Follower modus unterstützen) auch alle über ihre Leader Schnittstellen empfangenen Funktionalitäten über ihre Follower Schnittstellen verfügbar machen, können wir  $p_1 \rightarrow_P^* p_n$  für  $p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_{n-1} \rightarrow p_n$  mit  $p_2, \dots, p_{n-1} \in P$  schreiben um auszudrücken, dass das Produkt  $p_1$  seine Features über die Geräte  $p_2, \dots, p_{n-1}$  dem Produkt  $p_n$  bereitstellt.

---

<sup>2</sup>siehe Sektion 6

Wir definieren nun ein Matchingproblem als ein Tupel  $(S, r)$  wobei

- $S : \text{Szenario} \rightarrow \mathbb{P}(\text{Produkttyp})$  (eine partielle Abbildung von Szenarien auf Mengen aus Produkttypen) Szenario Warenkorb darstellt, der sowohl angibt, welche Szenarien implementiert werden sollen, als auch welche Produkttypfilter aktiv waren, als die entsprechenden Szenarien hinzugefügt wurden, und
- $r : \mathbb{B}$  ein Wahrheitswert ist, der die Renovationspräferenz des Benutzers darstellt.

Eine Lösung des Problems ist nun eine Menge  $L$  aus Produkten, so dass:

- Für jedes Metadevice  $m$  das in einem der Szenarien im Warenkorb vorkommt gibt es ein Produkt  $p \in L$ , dass dieses Implementiert ( $p \models m$ ). Dadurch wird sichergestellt, dass alle geforderten Funktionalitäten durch die Produkte in der Lösung implementiert werden.
- Für jedes Szenario gibt es ein Produkt  $p_b$ , dass die Basisstation dieses Szenarios implementiert. Zusätzlich gibt es für jedes andere Metadevice (Endpoints) in diesem Szenario ein Produkt  $p_e$  so dass  $p_e \rightarrow_L^*$ . Dadurch wird sichergestellt, dass alle auf der Basisstation des Szenarios laufenden Anwendungen zugang zu allen Geräten des Szenarios haben.
- Es gibt ein Produkt  $p_m \in L$ , so dass für alle anderen Produkte  $p \in L$   $p \rightarrow_L^* p_m$  gilt. Dadurch wird sichergestellt, dass das gesamte System über ein einzelnes Produkt steuerbar ist.

## 7.2 Szenario auswählen

Wenn der Benutzer die Onboardingphase verlässt, sollen ihm Szenarien präsentiert werden, die seinen Eingaben entsprechend am besten passen. Um das zu garantieren, muss nun aus der Liste der vorhandenen Szenarien die ausgewählt werden, die am besten zu der Kategoriepräferenz des Benutzers entsprechen. Dazu erhält der SzenarioSorting-Algorithmus den 4-Dimensionalen Vektor, der Benutzerkategoriepräferenz der in den einzelnen Zeilen Werte zwischen 1 und 10 beinhaltet, als Eingabe. In Abbildung 3 sind die für diesen Algorithmus verwendeten Eingaben zu sehen. Jede Zeile des Vektors beschreibt somit ein Rating<sup>3</sup>, das der Benutzer für die jeweiligen Kategorien gewählt hat.

---

<sup>3</sup>Rating beschreibt hier eine Wertigkeit zwischen 1 und 10, welches eine Zuordnung zu den einzelnen Kategorien „Energie“, „Gesundheit“, „Komfort“ und „Sicherheit“ beschreibt.

---

```

1  {
2      "scenario_preference": {
3          "Energie": 2,
4          "Komfort": 4,
5          "Gesundheit": 8,
6          "Sicherheit": 10
7      },
8      ...
9      "subcategory_filter": [],
10     ...

```

---

Abbildung 3: Werte die in der Szenarioauswahlphase benutzt werden

$$1 - \frac{\cos^{-1} \left( \frac{\vec{b} \cdot \vec{s}}{|\vec{b}| |\vec{s}|} \right)}{\frac{\pi}{2}} \quad (1)$$

Abbildung 4: Formel um den Abstand zwischen  $\vec{b}$  und  $\vec{s}$  zu finden

Anschließend wird jedes Szenario und auch das dazugehörige Rating der Kategorien aus der Datenbank abgefragt, welches einen ähnlichen Vektor ergibt. An dieser Stelle kann schon der Subkategoriefilter angewendet werden um nur Szenarien zu betrachten die auch in der Subkategorie liegen die dem Filter entsprechen. Der Benutzer kann so die Szenariosuche einschränken und gezielter nach Szenarien suchen<sup>4</sup>.

---

<sup>4</sup>Ein Benutzer der direkt aus dem Onboarding kommt wird diesen Filter noch nicht gesetzt haben, aber für spätere Aufrufe aus der Szenarioauswahlphase kann er hier schon mit betrachtet werden

```

8 ...
9 "renovation_preference": true
10 }

```

Abbildung 5: Die Renovierungspräferenz restriktiert die Menge der möglichen Implementationen eines Metadevices.

Um nun zu berechnen wie gut ein Szenario der Benutzerpräferenz entspricht, wird der Winkel zwischen dem Benutzerkategoriepräferenzvektor  $\vec{b}$  und dem Szenarioraitingvektor  $\vec{s}$  berechnet. In Abbildung 4 ist die Formel abgebildet, die den Winkel berechnet, auf Werte zwischen 0 und 1 beschränkt und anschließend invertiert, damit das das Ergebnis dem Benutzer in Prozent angezeigt werden kann. Dieses Ergebnis beschreibt wie gut das aktuelle Szenario zu den aktuellen Präferenzen passt. Anschließend wird die Liste der Szenarien anhand dieses Ratings sortiert und zurückgegeben. Wir haben uns für den Winkel anstelle einer anderen Metrik entschieden, da dadurch der Betrag der Bewertungen weg abstrahiert wird. Andere Metriken wie der Kosinus des Winkels oder der Euklidische Abstand ( $d(x, y) = \|x - y\|_2$ ) der normalisierten Bewertungsvektoren wären auch denkbar gewesen und würden die selbe Reihenfolge ergeben.

Nachdem die Szenarien gefunden wurden die am besten der Benutzerpräferenz entsprechen, wird im nächsten Schnitt validiert, dass jedes der gefundenen Szenarien mit zu den bereits ausgewählten Szenarien kompatiblen Produkten implementiert werden kann.

### 7.2.1 Implementation von Metadevices

Der erste Schritt um ein Szenario zu implementieren ist es, Produkte zu finden, die die Metadevices des Szenario implementieren. Ein Metadevice kann genau dann von einem Produkt implementiert werden, wenn alle Features des Metadevices vom Produkt bereitgestellt werden. Um für ein gegebenes Metadevice die implementierenden Produkte zu finden, wird daher die Produktdatenbank zunächst nach Produkten gefiltert, die alle Features des Metadevices bereitstellen. Als zusätzliche Optimierung wird danach noch sichergestellt, dass Produkte, die einen Broker implementieren sollen mindestens ein Protokoll im Leader Modus und Produkte die einen Endpoint implementieren sollen mindestens ein Protokoll im Follower Modus unterstützen.

An dieser Stelle wird auch die „Renovierungspräferenz“ (Abbildung 5) mit in Betracht gezogen. Hat der Benutzer hier „Nein“ angegeben, werden alle Produkte die Renovierungen erfordern würden aus der Betrachtung entfernt.

### 7.2.2 Finden von Produktkompatibilitäten

Nachdem nun bekannt ist, welche Produkte welche Metadevices implementieren, muss noch geprüft werden, wie es um die Kompatibilität der Produkte miteinander steht. Eine Anforderung an das Matching war, dass das ganze System durch nur eine Basisstation kontrollierbar sein soll. Daher werden zum Finden von untereinander kompatiblen Produkten alle infrage kommenden Basisstationen (i.e. die implementierenden Produkte jedes Metabrokers der zu implementierenden Szenarien) geprüft.

Angenommen man hat eine Metabroker Implementation  $B_p$  und betrachten eine aktuelle Metadeviceimplementation  $E_p$ . Dann betrachten wir alle Produkte in der Datenbank als gerichteten Graphen, bei dem  $E_p$  der Startpunkt und  $B_p$  das Ziel definiert. Die Knoten im Graphen sind Produkte und die gerichteten Kanten stellen dar, dass  $p_n$  mit  $p_{n+1}$  kommunizieren kann (dass heißt, dass es ein Protokoll bei  $p_n$  im Followermodus und bei  $p_{n+1}$  das gleiche Protokoll im Leadermodus existiert), wenn eine Kante von  $p_n$  nach  $p_{n+1}$  existiert.

Nun kann von  $E_p$  eine Tiefensuche gestartet werden und somit alle Pfade von  $E_p$  nach  $B_p$  gefunden werden. Alle Knoten die zwischen  $E_p$  und  $B_p$  liegen werden als Kommunikationsbrücken in die Lösungsmenge aufgeführt. Diese werden Bridges genannt, da sie nur die Kommunikation weiterleiten und keinen Einfluss auf die benötigten Featuremengen des Pfades haben.

Um sicherzustellen, dass der Pfad auch der Benutzerpräferenzen entspricht wird die Renovierungspräferenz mit übergeben um nur Bridges zu erhalten, die dieser Präferenz entsprechen.

Dieser Prozess wird für alle möglichen Implementierungen des Metaendpoints vollzogen, welches zu einer Lösungsmenge aus Pfaden vom Metaendpoint zu einer Implementation des Metabrokers führt.

```
7   ...
8   "product_preference": "Preis",
9   ...
```

Abbildung 6: Produktpräferenzen unterteilen sich in „Preis“, „Erweiterbarkeit“ und „Effizienz“.

Anschließend wird dieser Prozess für jeden Metaendpoint zu der aktuellen Brokerimplementation  $B_p$  durchgeführt und es entsteht für jeden Metaendpoint mehrere Lösungssets (eins für jede Implementation) welche mögliche Lösungspfade enthalten. Nun wird das Kreuzprodukt aller Lösungen gebildet um jede mögliche Lösung von  $B_p$  zu allen möglichen Endpointimplementatio-nen zu finden, welches dann ein gültiges Produktset genannt wird. Zum einen wird dadurch sichergestellt, dass jeder Endpoint implementiert wird und zum anderen, dass Pfade die zuerst unnötig lang erschienen, nun Bridges enthalten können, die auch von anderen Endpoints benutzt werden und somit ein wichtiges Glied in der Kette der Produkte ergeben.

Da die zu implementierenden Szenarien auch einen Produkttypfilter enthalten, muss noch sichergestellt werden, dass die gefundenen Lösungen diesem genügen. Die Lösungsmenge muss reduziert werden, sodass jedes Produktset mindestens ein Produkt beinhaltet, welches von einem bestimmten Produkttyp ist und somit nur noch gültige Ergebnisse aufweist. Dies wird für alle Produkttypfilter sichergestellt.

Nun erhält man Produktsets die valide Implementationen von jedem Metaendpoint zu der aktuellen Metabrokerimplementation enthalten und auch dem Produkttypfilter entsprechen. Um nun ein Produktset davon auszuwählen, wird eine Optimierungsfunktion auf diese Lösungsmengen angewendet (siehe Sektion 7.2.3).

Da bis jetzt nur eine Metabrokerimplementation betrachtet wurde, muss man den gesamten Vorgang auch für alle anderen Implementationen durchführen und anschließend alle Lösungen wieder in die Optimierungsfunktion eingeben um das beste Produktset zu finden, welches das Szenario implementiert und der Benutzerpräferenz am besten passt.

$$R_{U_{pref}} = \frac{1}{\sum_{p \in P_{set}} U_{pref}(p)} * 0.95^{Bridge(P_{set})} \quad \left. \right\} U_{pref} \in \{Preis, Effizienz\} \quad (2)$$

$$R_{U_{pref}} = 1 / \frac{Bridge(P_{set})^2}{\sum_{p \in P_{set}} U_{pref}(p)} \quad \left. \right\} U_{pref} \in \{Erweiterbarkeit\} \quad (3)$$

Abbildung 7: Formel für die Berechnung der Wertigkeit  $R_{U_{pref}}$  eines Produktsets.

### 7.2.3 Auswahl der optimalen Lösung

Die Auswahl einer optimalen Lösung aus einer Lösungsmenge muss in Abhängigkeit der Benutzerproduktpräferenz getroffen werden. Der Benutzer trifft die Entscheidung ob ihm der Preis, Erweiterbarkeit oder die Effizienz seiner Produkte am wichtigsten ist (Abbildung 6).

Der zweite wichtige Punkt, der in die Auswahl des besten Sets eingeschlossen wird, ist die Anzahl der Bridges. Desto mehr Bridges das Produktset enthält, je unattraktiver wird es für den Benutzer, da das Produktset Gräte beinhaltet, die für das Szenario keinen wirklichen Funktionalitätsgewinn darstellen.

Betrachten wir zuerst Effizienz und Preis, da diese sich in dem Formelaufbau nicht unterscheiden. Gegeben sei somit ein Produktset  $P_{set}$ , somit auch die Anzahl der Bridges in diesem Set  $Bridge(P_{set})$  und die Produktpräferenz des Benutzers  $U_{pref}$ . Die Formel 2 in Abbildung 7 stellt diese Variablen in Zusammenhang und berechnet einen Koeffizienten der die Wertigkeit dieses Sets widerspiegelt. Wird der Nenner sehr groß (teuer oder ineffizient) wird der Bruch sehr klein. Dazu multipliziert sich die Anzahl der Bridges als Exponenten einer kleinen Basis kleiner als 1. Dies führt dazu, dass wenn viele Bridges verwendet wurden, die Wertigkeit noch weiter sinkt. Desto kleiner die Wertigkeit, desto schlechter ist das Produktset für den Benutzer geeignet.

Der Preis und die Effizienz wird an dem Produkt abgelesen wohingegen die Erweiterbarkeit anhand der Anzahl von Leader- und im Followerprotokolle berechnet wird. Die Formel 3 in Abbildung 7 stellt die Berechnung der Wertigkeit unter der Benutzerpräferenz „Erweiterbarkeit“ dar.

Hier wurde eine andere Formel gewählt, da bei der Erweiterbarkeit mehr Protokolle besser sind als weniger. Auch muss beachtet werden, dass mehr Bridges auch mehr Protokolle bedeuten und somit muss es einen Schnittpunkt geben bei dem die Anzahl der Bridges stärker gewichtet als die Anzahl der Protokolle.

Im Zähler steht die Anzahl der Bridges zum Quadrat und im Nenner steht die Erweiterbarkeit. Da eine quadratische Funktion schneller wächst als eine lineare Funktion, wird schnell der Punkt erreicht wo die Anzahl der Bridges stärker gewichtet wird als die Anzahl der Protokolle und somit die gesamt Wertigkeit sinkt. Sollte dieser Punkt noch nicht erreicht worden sein, ist der Nenner größer als der Zahler wobei die gesamt Wertigkeit steigt. Hier bewegen sich die Wertigkeiten nicht in einem Bereich von 0 bis 1 sondern können auch größer als 1 werden.

Alternativ kann man die Exponenten variieren um schneller oder langsamer an den Schnittpunkt zu gelangen beidem die Anzahl der Bridges überwiegt. Wie schnell diese Funktion dann wächst ist irrelevant, da später nur die Reihenfolge der Wertigkeiten eine Rolle spielt und es somit keien Rolle spielt welche Metrik hier verwendet wird.

Nachdem nun jedem Produktset aus der Lösungsmenge eine Wertigkeit unter der Benutzerpräferenz zugeteilt wurde, wird das Produktset mit der größten Wertigkeit zurück gegeben.

## 7.3 Implementation mehren Szenarien

Damit der Benutzer auch mehr als nur ein Szenario in seinen Warenkorb legen darf, muss es eine Möglichkeit geben wie man gültige Implementationen zu mehreren Szenarien findet.

### 7.3.1 Merging von Metadevices

Da jedes Szenario unterschiedliche Metabroker und Metaendpoints definiert, muss diese Menge aus Metadevices zusammengeführt werden.

Das Ziel des Zusammenföhres der Metadevices ist es eine Menge von Metabroker und Metaendpoints zu erstellen, bei denen es keine Überschneidungen in der Featuremenge gibt. Dazu wird sich unabhängig voneinander die Metabroker und die Metaendpoints durchiteriert und nach Schnittmengen in der Featuremenge jedes Gerätes gesucht. Wenn die Featuremenge eines Metadevices  $M_x$  komplett in einem anderen  $M_y$  enthalten ist kann  $M_x$  durch  $M_y$  substituiert werden. Wenn es keine Schnittmenge gibt, wird das Metadevice als neues Element in die jeweilig Menge hinzugefügt.

Man erhält nun zwei Mengen: Zum einen die zusammengeföhrenen Metabroker und die zusammengeföhrenen Metaendpoints. Es können nun zwei Fälle eintreten:

1. Die Anzahl der zusammengeföhrenen Metabroker ist genau 1 und
2. Die Anzahl der zusammengeföhrenen Metabroker sind mehr als 1.

Wenn der erste Fall eintreten sollte heißt das, dass alle anderen Metabroker durch einen zusammengeföhrenen Metabroker dargestellt werden konnten. Wir können nun den zusammengeföhrenen Metabroker und die zusammengeföhrenen Metaendpoints als ein neues Metaszenario betrachten und es in den Matchingalgorithmus eingeben, da dieser nur auf Metadevices arbeitet und durch die zusammengeföhrenen Metadevices implizit mehrere Szenarien implementiert.

Wenn allerdings nicht alle Metabroker zusammengeführt werden konnten, wird ein Meta broker aus der Menge der zusammengeföhrenen Metabroker ausgewählt und alle anderen in die Menge der zusammengeföhrenen Metaendpoints übertragen. Anschließend wird der Algorithmus aufgerufen und es wird überprüft ob dieses Metaszenario implementierbar wäre. Zusätzlich zu dem normalen Ansatz ein Szenario zu implementieren, muss in diesem Fall noch überprüft werden, dass jeder zusammengeföhrt Metaendpoint auch mit seinem eigentlichen zusammengeföhrenen Metabroker kommunizieren kann. Durch eine weitere Pfadüberprüfung lässt sich dies validieren und es können alle Pfade gelöscht werden die diese Bedingung nicht erfüllen.

Wenn ein gültiges Produktset gefunden werden konnte, wird dieses wieder in eine Lösungsmenge aufgenommen. Nachdem alle zusammengeföhrenen Metabroker einmal als Mastermetabroker definiert waren, kann auf der gefunden Lösungsmenge wieder die Optimierungsfunktion aus der Sektion 7.2.3 aufgerufen werden, um das Produktset zu finden, welches am besten zu der Benutzerpräferenz entspricht.

```

9   ...
10  "shopping_basket": [
11  {
12    "scenario_id": 3,
13    "product_type_filter": [ 6, 9 ]
14  },
15  ],
16  ...
17 }

```

Abbildung 8: Beispielanfrage an den `/suggestions` Endpoint.

### 7.3.2 Produkttypfilter

Ein Produktset das in einem vorigen Schritt mit einem gewissen Produkttypfilter implementiert wurde, muss auch in den zusammengeführten Szenarien mit diesem Filter versehen werden, damit sichergestellt wird, dass ein Benutzer immer noch die gleiche Lösung erhält. Um diese Bedingung bildlicher zu gestalten sagen wir, dass ein Benutzer „Bob“ sich das Szenario „Steady Standby: In bestimmten Zeitfenstern sollen elektronische Geräte ausgeschaltet werden.“ mit dem Produkttypfilter „Waschmaschine“ ausgewählt hat. Woraufhin der Matchingalgorithmus das Produktset „IOLTIE“ als Broker und eine Waschmaschine von Samsung gefunden hat. Wählt nun Bob ein neues Szenario „Aufwachanreiz“ aus, welches beinhaltet, dass eine dimmbare Lampe langsam aufleuchtet, aus, dann würde der Algorithmus den vorigen Produkttypfilter vernachlässigen und anstatt der Waschmaschine eine dimmbare Lampe einsetzten. Bob würde sich wundern wo seine Waschmaschine hin verschwunden ist.

Deswegen erwartet das Matching als Shoppingbasket Elemente, die aus einer Szenario ID und einer Liste aus Produkttypfiltern bestehen. Die Beispielanfrage ist in Abbildung 8 dargestellt.

Um dies zu validieren muss herausgefunden werden zu welchen Szenario die zusammengeführten Metadevices gehörten und anschließend muss die Menge von Produkten, die genau diese zusammengeführten Metadevices implementieren, den Produkttypfiltern entsprechen.

## 7.4 Produktalternativen

In der Produktauswahlphase ist die Aufgabe des Backends, zu einem gegebenen Produkt Alternativen zu suchen. Prinzipiell kann ein Produkt durch ein anderes ausgetauscht werden, wenn es dieselbe Funktionalität (i.e. das selbe Metadevice) implementiert. Dabei ist allerdings zu beachten, dass das neue Produkt unter Umständen andere Protokolle unterstützt, was dazu führen kann, dass auch andere Geräte ausgetauscht werden müssen. Es ist auch möglich, dass es zu einem alternativen Produkt keine gültige Lösung gibt. In diesem Fall soll das Produkt dann gar nicht erst als alternative Vorgeschlagen werden.

Glücklicherweise sind sowohl das Finden von Alternativen als auch das Sicherstellen der Kompatibilität ohne größere Performanceeinbußen möglich, da wir ohnehin alle gültigen Lösungen auflisten um die für den Benutzer optimale Lösung auswählen zu können. Indem wir für jede gültige Lösung sammeln welche Produkte welche Metadevices implementieren und diese Teilergebnisse dann zusammenführen können wir auf einfache Weise alle gültigen Austauschmöglichkeiten berechnen.

Es kann sein, dass ein Produkt in einer gefundenen Lösung kein Metadevice implementiert. Dies ist zum Beispiel bei Bridges der Fall, die selbst keine Features bereitstellen aber zur Verbindung der verschiedenen Geräte untereinander notwendig sind. Falls solche Geräte in der angezeigten Produktliste vorhanden sind, sind die Buttons für das Festlegen des Produktes und das Anzeigen der Alternativen im Frontend gesperrt um anzudeuten, dass das Produkt nicht austauschbar ist.

Ebenso ist es möglich, dass ein Produkt mehrere Metadevices implementiert<sup>5</sup>. In diesem Fall wird nur nach Alternativen gesucht, die alle Metadevices implementieren. Die Begründung dafür ist, dass es den Benutzer verwirren könnte, wenn er sieht, dass er seine smartes Türschloss (mit integrierter Kamera) durch eine einfache Webcam ersetzen kann. Ein weiterer Spezialfall ist der Fall, dass zwei Produkte das selbe Metadevice implementieren. In diesem Fall kann das gemeinsam benutze Metadevice nicht in die Suche nach alternativen einfließen. Da eine Produktersetzung vorschreibt, dass die entsprechenden Metadevices nur durch das gewünschte Produkt implementiert werden dürfen (siehe nächster Absatz), wäre es ansonsten möglich widersprüchliche Anfragen zu stellen.

---

<sup>5</sup>Als Beispiel dafür kann man einen Wecker nehmen, der gleichzeitig auch ein Radio ist

```
1 {
2     "shopping_basket": [
3         "scenario_id": 1,
4         "product_type_filter": []
5     ],
6     "locked_products": [],
7     "product_preference": "Preis",
8     "renovation_preference": true
9 }
```

Abbildung 9: Beispielanfrage an den `/final_product_list` Endpoint.

Ausgewählte Alternativen und direkt festgelegte Produkte werden über die API als Zuordnungen von Listen aus Metadevice IDs zu Produkt IDs übergeben. Im Matchingalgorithmus werden diese Daten dann dadurch verwertet, dass nur das jeweils angegebene Produkt verwendet werden kann um die jeweiligen Metadevices zu implementieren (siehe Sektion 7.2.1). Dadurch wird auch sichergestellt, dass spätere Anfragen nach Produktalternativen nur Produkte liefern, die kompatibel zu den Produkten sind, die der Benutzer explizit ausgewählt hat.

Die Produktauswahlphase benutzt zwei API Schnittstellen:

- `/final_product_list` um die Liste an Produkten für den gegebenen Szenariowarenkorb und gegebene Benutzerpräferenzen zu erhalten. Ein Beispiel für die zur Schnittstelle gesendeten Daten ist in Abbildung 9 zu sehen.
- `/product_alternatives` um alternative Produkte für eine gegebene Liste von Metadevices (und einen gegebenen Szenariowarenkorb / Benutzerpräferenzen) zu finden.

## 7.5 Berechnung von Preis, Erweiterbarkeit und Effizenz

Da wir einen Smart Home Konfiguration anbieten wollen, haben wir uns stark an anderen Konfigurationen wie zum Beispiel in der Automobilindustrie oder der Informations-Technologie orientiert. Bei diesen Konfigurationen ist es üblich anzusehen, wie viel eine gewählte Komposte den Preis erhöhen oder absenken würde. Um dem Benutzer auch bei DONAALDA ein live-update anzusehen, was Preis, Effizienz und Erweiterbarkeit betrifft, zeigen wir bei den Szenarien die Differenz von dem Produktset des neu vorgeschlagenen Szenarios zu dem Produktset das schon im Warenkorb vorhandenen Szenarien an.

Dies ermöglicht es dem Benutzer direktes Feedback zu geben und ihm zu ermutigen sich mehr mit seinem Smart Home auseinander zu setzen. So kann zum Beispiel ein Szenario 0 Euro kosten, wenn alle Komponenten, die für dieses Szenario benötigt werden, schon im Produktset der Szenarien im Warenkorb vorhanden sind.

Da wir keinen Zustand über den aktuellen Benutzer speichern<sup>6</sup>, wird bei jeder Anfrage die Implementierung des Warenkorbs errechnet (analog kann dieses Produktset auch zwischengespeichert werden um unnötige doppelte Berechnungen zu vermeiden), dann das Produktset für vorgeschlagene Szenario in Kombination mit den schon im Warenkorb vorhanden Szenarien gefunden und abschließend die Differenz der beiden Produktsets berechnet. Daraus ergeben sich die Differenz der Effizienz, des Preises, und der Erweiterbarkeit, die wie schon beschrieben anhand der Anzahl der Unterstützen Protokolle der Produkte im Produktset abgelesen wird.

So können Fälle eintreten bei denen ein neues Szenario den Preis senken könnte, da nun mehrere Produkte zu einem zusammengefasst werden können und somit der Preis gesenkt werden kann. Dieser Fall sollte nur eintreten, wenn der Benutzer bei seiner Produktpräferenz nicht „Preis“ angegeben hat, da sonst immer direkt das Produktset mit dem günstigen Preis ausgewählt wird. Analog wird mit der Effizienz umgegangen.

---

<sup>6</sup>Stateless: Vorteil liegt bei dem geringeren Datenmanagement für jeden Benutzer und bringt somit einen Geschwindigkeitsgewinn.

### **7.5.1 Herausforderungen und Designentscheidungen**

Diese Implementierung hat gezeigt, wie man ein oder mehrere Szenarien implementieren kann, aber es bleiben noch offene Fragen und Probleme die sich aus diesem Ansatz ergeben. Um das Matching im Ganzen zu beschreiben und einen guten Rückschluss zu ziehen, müssen viele Komponenten mit betrachtet werden. Nicht nur die Herausforderungen, die während der Implementation aufgetreten sind, haben sich als schwierig heraus gestellt, sondern auch das Grunddesign war schon ein Meilenstein an sich.

Zuerst musste klargestellt werden an welchen Parametern das Matching von Szenarien zu Produkten durchgeführt werden soll. „Tags“ (Schlagwörter, die ein Szenario beschreiben) hätte man durchaus verwenden können, aber da Produkte nie genau auf ein Tag passen, hätte man sich überlegen müssen wie man Tags zusammengefasst oder wie man ähnliche Tags findet. Dieser Ansatz wurde verworfen, da er eine große Komplexität hat und somit schwierig zu kontrollieren ist.

Als zweiten Ansatz hatten wir statische Produktsets untersucht. Sie ergeben eine valide und verifizierte Lösung mit der wir als Administratoren garantieren können, dass sie auch die gewünschten Funktionalitäten erfüllen, aber diese fest definierten Sets sind unflexibel und skalieren nicht. Ein dauerhafter Aufwand wäre nötig um neue Produktsets auf Szenarien zu mappen und Produkthersteller könnten sich benachteiligt fühlen wenn ihr Produkt noch nicht in ein Produktset aufgenommen wurde.

Schlussendlich blieb nur das Matching auf Funktionalitäten übrig. Zwar kann jeder Hersteller neue exotische Funktionalitäten definieren, aber dann muss er auch damit rechnen, dass sein Produkt weniger häufig gefunden wird, da es weniger Szenarien gibt, die genau diese Funktionalitäten verlangen. So entsteht ein gegenseitiges Interesse der Community einfache Szenarien zu erstellen, damit diese von Produkten implementierbar sind, als auch von den Herstellern Funktionalitäten zu verwenden, die in der Datenbank schon bekannt sind.

Als nächstes wandten wir uns der Kommunikation zwischen den Produkten zu. Das Problem, dass mit diesen einhergeht ist, dass die Kommunikation Geräte spezifisch ist. Es gibt keinen einheitlichen Standard über den alle IoT Geräte kommunizieren können. Manche müssen per USB-Kabel angeschlossen werden, wieder andere verbinden sich mit Bluetooth, aber nur zu bestimmten Gateways usw. Einen realistischen Mittelweg zu finden, der alle Bedürfnisse aller Produkte befriedigt, war in diesem Projekt nicht zu erreichen. Es wurde zwar überlegt eine Schnittstellendefinierungssache zu nutzen um von jedem Gerät die Dockingports zu modellieren und dann nach möglichen Routen zu suchen, aber zum einen sind die unterstützten Protokolle im Netz schlecht bis gar nicht definiert und zum anderen hätte das Semester für diesen Ansatz nicht ausgereicht.

Als wir die Onboardingfragen designt haben, stand auch die Frage im Raum, ob wir die Größe der Wohnung mit einbeziehen sollten. Welches ein wichtiger Parameter und benötigte Information ist um das Haus des Benutzers zu modellieren. Argument für diesen Ansatz war es, dass einem Benutzer es nicht genüge nur eine Lampe im Haus zu haben, wenn er eigentlich sein ganzes Wohnzimmer erleuchten wolle. Um das aber im Backend realitätsnah abzubilden würde die Wohnungsgröße oder die Anzahl der Zimmer nicht ausreichen. Man müsste schon die Wohnung genau modellieren um festzustellen, wie viele Lampen der Benutzer benötigt um seinen Wünschen entsprechend einen Raum abzudecken. Diese Informationen lassen sich nur bedingt erfragen. Fragt man nach der Wohnungsgröße und der Anzahl der Zimmer könnte man noch genauer die Anzahl der Geräte bestimmten, aber trotzdem wäre diese Größe eine Schätzung. So haben wir uns entschlossen den Benutzer in nur drei Fragen zur Szenarioauswahlphase zu geleiten und ihm am Ende des Advisors mit einer Zusammenfassung auszustatten, die er dann einem Experten zeigen kann, der sich die Wohnung genau anschaut.

Desweiteren wird nicht betrachtet was passiert, wenn es mehrere Optionen für das Zusammenführen von ein Metadevice gibt. In der aktuellen Implementation wird das erste ersetzbare Metadevice genommen und mit diesem die Substitution vollzogen. Ob noch mehr Optionen für dieses Metadevice existieren würde, wird nicht betrachtet. In diesem Fall ist aber eine Warnung in den Logdateien zu finden. Um eine valide und abgeschlossene Lösung zu finden, müsste jede mögliche Substitutivoption betrachtet werden, welches aber dann zu einem größeren Berechnungsaufwand führt.

Allerdings erwies sich ein Faktor als problematischer als am Anfang vermutet: Die Komplexität. Wenn die Metabroker nicht zusammenführbar sind, muss das Matching für jeden möglichen Metabroker aufgerufen werden, welches dann wieder jede mögliche Implementierung dieses Metabrokers betrachtet usw. Desto mehr Metadevices ein Szenario hat desto exponentiell größer wird der Berechnungsaufwand. Deswegen war Performance Optimierung ein wichtiger Punkt bei der Implementierung. Eine Datenbankanfrage, die im Schnitt 500ms gebraucht hat, hat in der untersten Schleife das Matching zum abstürzten gebracht. Um trotzdem in Echtzeit antworten zu können wurde das Zwischenspeichern von Ergebnissen eingeführt. Ein Benutzer schickt in einer Session mehrfach die gleiche Anfrage an den Server. Um nicht jedes mal die Anfrage neu verarbeiten zu müssen wird das Ergebnis zwischengespeichert und für spätere Anfragen bereitgestellt. Eine effizientere und dennoch valide Lösung zu finden, wird für weitergehende Arbeiten offen gehalten.

## 7.6 Zusammenfassung

Nachdem die grundlegenden Herausforderungen in der Implementierung und im Design kurz erläutert wurden, bleibt die Frage ob das Matching realitätsnah ist. Dies lässt sich pauschal nicht beantworten. Für das Einsatzgebiet im Advisor ist der Algorithmus gut geeignet, da der Benutzer mehr über sein Haus lernen will und einen groben Überblick über die Kosten eines Smart Homes erlangen möchte. Zum modellieren eines Hauses/Wohnung eignet er sich allerdings nicht. Dazu müsste der Benutzer das System mit genausten Informationen über sein Haus anleiten, welches zum einen aus der Perspektive des Benutzers unrealistisch ist und zum anderen sich mit einem annehmbaren Brechungsaufwand nicht vereinbaren lässt.

Alles im allen erfüllt das Matching seinen Zweck. Szenarien können auf Produkte gemappt werden, Produktalternativen können errechnet werden und es kann (abstrahiert) validiert werden, dass die Lösung auch diese Szenarien abbilden kann. Trotzdem bleib Platz nach oben. Zum einen müssen die Produktschnittstellen richtig verarbeitet werden. Dazu sollte man sich einer Schnittstellenbeschreibungssprache bedienen, die diese Abhängigkeiten darstellen kann. Zum anderen sollte man sich anschauen an welchen Stellen man die Lösung abstrahieren kann und somit nicht immer das Optimum findet, aber dafür in den meisten Fällen. Da das Problem vom Finden von gültigen Produktmengen NP-Schwer ist, ist unser Lösungsansatz optimal solange  $P \neq NP$  angenommen wird.

## 8 Infrastruktur - Backend

### 8.1 Django

Zu Beginn des Projekts vor einem Jahr wurde uns freie Wahl unserer Werkzeuge und Frameworks gelassen. Nach kürzerer Recherche konnten wir die Auswahl der Verfügbaren Frameworks auf zwei reduzieren. Bei der Wahl zwischen Django und Ruby on Rails haben wir uns auf Grund der dahinterliegenden Programmiersprachen, Python und Ruby, dann schlussendlich für Python - und damit Django - entschieden. Hauptgrund war, dass ein Großteil des Teams bereits mit Python Erfahrungen sammeln konnte. Zusätzlich mussten wir auch relativ zügig loslegen, sodass der Einstieg in eine neue Programmiersprache uns dann doch als zu riskant erschien. Weitere Gründe für die Wahl von Python als präferierte Programmiersprache für das Backend Framework:

- Simpel und leicht verständliche Syntax
  - Keine end of line Zeichen wie Semikolons oder sonstige merkwürdige Konstellation von Satzzeichen wie ::
  - Keine umständlichen Konstruktion von Anweisungen nötig wie in z.B. VBA
  - Simpler Aufbau von Schleifen mit Mehrfachzuweisung für For-Each
  - Sofort ersichtliche Abfolge und Abhängigkeit von Anweisungen durch Pflichteinrückung
- Weitverbreitet, so z.B. auch im akademischen Umfeld bzgl. Machine Learning

Python, wie auch Ruby, ist eine Skriptsprache. Die Syntax ist klar und sehr übersichtlich. Das ermöglicht auch eine besonders einfache Weiterverwendung des Projekts, da auch Python eine Sprache ist, die noch einen weitaus breiteren Einsatzzweck hat. Django selber ist ein weit verbreitetes und beliebtes Backend Framework. Wie auch Ruby on Rails basiert es auf dem MVC-Schema und schafft somit eine gewisse Ordnung in der Codebasis. Es existiert eine immens große Anzahl an offen verfügbaren Packages, welche simpel und schnell einbindbar sind. Weiterhin existiert eine überaus detaillierte Dokumentation zu Python, sodass sehr schnell

## 8.2 Memcached

Mit Ausbau des Matchings wurden die Ladezeiten der Ergebnisse auch immer länger. Daher haben wir uns entschlossen, dass bereits errechnete Ergebnisse zwischengespeichert werden sollen, um Zugriffszeiten und Datenbankanfragen zu minimieren. Nach kurzer Recherche hat sich herausgestellt, dass Django nativ memcached unterstützt. Da wir allerdings noch Alternativen in Betracht ziehen wollten, haben wir weiterrecherchiert. Weitere Möglichkeiten wären:

- Redis als Cacheinstanz zu konfigurieren und parallel laufen zu lassen
- Eine Tabelle einer SQL-basierten Datenbank als Cachetabelle zu verwenden
- Eine NoSQL Datenbank als Cacheserver zu verwenden, z.B. MongoDB

Nach kurzer Analyse sind wir zu dem Schluss gekommen, dass wir memcached verwenden wollen. Grund hierfür war die native Out-of-the-box Unterstützung und das Wegfallen sehr zeitaufwendiger Konfiguration. Weiterhin gab es auch keine Gründe die gegen den Einsatz eines memcached Servers gesprochen hätten, da auf unserer Linux VM noch genug freier Arbeitsspeicher zur Verfügung stand.

## 8.3 Swagger

Swagger ist eine offene Schnittstellenbeschreibungssprache für REST APIs. Hierdurch werden die für andere Dienste zur Verfügung gestellten Schnittstellen programmiersprachenagnostisch für Mensch und Maschine zugleich bereitgestellt. Diese Schnittstellen sind ohne Zugang zum Quellcode oder der Quellcodedokumentation klar verständlich und nutzbar. Nutzer können auf einfachste Art und Weise auf die Funktionen zugreifen und über die API Beschreibung die erwarteten Eingabe- und Ausgabewerte abfragen. Durch eine einheitliche Beschreibungssprache ist sichergestellt, dass Nutzer und Anbieter gleichermaßen, exakt kommunizieren können. Da die REST API Swagger konform erstellt ist, bis auf die in Kapitel 7 näher erläuterten Funktionen, ist eine genaue Erläuterung der Funktionalitäten nicht notwendig. Anbei möchten wir aber trotzdem die Übersicht der zur Verfügung gestellten Methoden präsentieren.

## Pastebin API

**swagger**

Show/Hide | List Operations | Expand Operations

POST	/api/swagger/suggestions	Set the user preferences and return a page of scenario suggestions.
POST	/api/swagger/final_product_list	
POST	/api/swagger/product_alternatives	

---

**v1**

Show/Hide | List Operations | Expand Operations

GET	/api/v1/category/	
GET	/api/v1/category/{id}/	
GET	/api/v1/comment/	
GET	/api/v1/comment/{id}/	Returns a single Comment item
GET	/api/v1/product/	
GET	/api/v1/product/{id}/	
GET	/api/v1/productType/	
GET	/api/v1/productType/{id}/	
GET	/api/v1/provider/	
GET	/api/v1/provider/{id}/	
GET	/api/v1/providerProfile/	
GET	/api/v1/providerProfile/{id}/	
GET	/api/v1/questions/	
GET	/api/v1/questions/{id}/	
GET	/api/v1/scenario/	
GET	/api/v1/scenario/{id}/	
GET	/api/v1/subCategory/	
GET	/api/v1/subCategory/{id}/	
GET	/api/v1/subCategoryDescription/	
GET	/api/v1/subCategoryDescription/{id}/	

Abbildung 10: Swagger Methodenübersicht der API

## 9 Anleitung - Backend

### 9.1 Installation

Dieses Kapitel beinhaltet alle zur Installation des aktuellen Systems notwendigen Schritte und Pakete.

#### 9.1.1 Abhangigkeiten

Vor dem erstmaligen Start des Advisors mussen unterschiedliche Abhangigkeiten eingerichtet werden. Python muss mindestens in Version 3.5.1 vorliegen und die Path Variablen korrekt im Betriebssystem integriert sein. Der Python Package Manager PIP sollte idealerweise in einer zu der Python Version passenden Version installiert und wie auch schon Python selber in den Path Variablen stehen. Anschlieend kann entweder die requirements\_windows.txt oder requirements\_unix.txt per `pip install -r path/to/requirements.txt` oder `pip3 install -r /path/to/requirements.txt` installiert werden. Alternativ konnen auch die in der nachfolgenden Liste erwahnten packages manuell installiert werden. Frontendseitig muss node.js und npm zwingend installiert werden. Dafur muss zuerst `sudo apt-get install nodejs` und anschlieend `codesudo apt-get install npm` ausgefuhrt werden.

- Django >= 1.10.3
- django\_imagekit >= 3.3
- Pillow >= 3.2.0
- django-bower >= 5.1.0
- django\_static\_precompiler >= 1.4
- djangorestframework >= 3.5.3
- markdown >= 2.6.6
- django-filter >= 0.15.3
- coreapi >= 2.0.9
- django-crispy-forms >= 1.6.0
- django-rest-swagger >= 2.1.0
- drfdocs >= 0.0.11

---

```
1 import pip
2 from subprocess import call
3
4 for dist in pip.get_installed_distributions():
5     call ("pip3 install --upgrade " + dist.
          project_Name, shell=True)
```

---

Abbildung 11: Django package update skript

- future >= 0.16.0
- django-cors-headers >= 1.1.0
- python-memcached >= 1.58
- django-suit >= 0.2.23
- django-markdownx == 1.7
- django-material == 0.10.1
- openapi == 0.5.0
- jsonschema == 2.5.1

Achtung: Unter Unix-Systemen kann es zu Komplikationen bei der Installation von Pillow kommen, da hier die Abhängigkeiten nicht automatisch mitinstalliert werden. Es muss zumindest libjpeg per `apt-get install libjpeg` (bzw. der zu einem anderem Paketverwaltungssystem gehörende Aufruf) installiert werden.<sup>7</sup> Falls weitere Versionskonflikte mit den Djangopacketen auftreten sollten, hilft folgendes Python Skript:

---

<sup>7</sup>Mehr Informationen unter <http://pillow.readthedocs.io/en/3.1.x/installation.html#external-libraries>.

---

```

1 DATABASES = {
2     'default': {
3         'ENGINE': 'django.db.backends.postgresql',
4         'NAME': 'Datenbankname',
5         'USER': 'Nutzername der Datenbankbenutzerkontos
6             ',
7         'PASSWORD': 'Passwort des
8             Datenbankbenutzerkontos',
9         'HOST': 'IP des Datenbankservers',
10        'PORT': 'Port des Datenbankservers',
11    }
12}

```

---

Abbildung 12: PostgreSQL Konfiguration settings.py

### 9.1.2 PostgreSQL

Falls auf dieselbe Datenbankalternative gesetzt werden soll, müssen noch zwei weitere Installationen getätigt werden. Zum einen muss PostgreSQL<sup>8</sup> installiert werden und zum anderen dann noch das Package psycopg2<sup>9</sup>. Hierbei ist zu beachten, dass dieses selber kompiliert werden muss, bzw. ein pre-kompiliertes package für Windows<sup>10</sup>, welches auf die installierte Python Version abgestimmt sein muss, installiert werden muss. Dieses wird über `python easy_install ./path/to/binary` oder `python3 easy_install./path/to/binary` installiert. Abschließend muss in der `settings.py` noch folgendes Dictionary angepasst werden:

### 9.1.3 Andere Datenbanken

Grundsätzlich ist Django mit allen bekannteren Datenbankenkompatibel und alle benötigten Informationen zur Einrichtung sind auch gut über die Dokumentation erreichbar<sup>11</sup>.

---

<sup>8</sup>Zu finden unter <https://www.postgresql.org/download/>.

<sup>9</sup>Erreichbar unter <http://initd.org/psycopg/docs/install.html>.

<sup>10</sup>Verfügbar unter <http://www.lfd.uci.edu/~gohlke/pythonlibs/>.

<sup>11</sup>Verfügbar unter <https://docs.djangoproject.com/en/1.10/ref/settings/#std:setting-DATABASE-ENGINE>.

## 9.2 Start

Ausgeführt werden kann der Advisor über mehrere Wege. Entweder wird der Quellcode in eine IDE eingebunden und dann über die integrierten Run Befehle gestartet, oder er wird über die jeweilige Shell-Derivative durch `python /path/to/manage.py runserver [optional: Port Nummer die verwendet werden soll]` oder `python3 /path/to/manage.py runserver [optional: Port Nummer die verwendet werden soll]` gestartet.

# 10 Deployment

Nachfolgend werden die für das Deployment notwendigen Services und Dienste kurz erläutert. Anschließend werden noch Möglichkeiten zur Erweiterung und Modularisierung der Infrastruktur gegeben.

## 10.1 Aktuelle Konfiguration

Die für die aktuelle Konfiguration notwendigen Einstellungen der jeweiligen Dienste werden im Anschluss näher erläutert.

### 10.1.1 Nginx, Gunicorn und PostgreSQL

Die aktuell laufende Version wird über zwei Serverdienste realisiert. Das Django Projekt wird über Gunicorn per WSGI durch einen Unix Socket für Nginx erreichbar gemacht. Nginx verwaltet den externen Traffic und kommuniziert über diesen Socket mit dem Gunicorn-Anwendungsserver, um die angeforderten Inhalte darzustellen. Hierbei handelt es sich um ein Reverse-Proxy Setup. Der Vorteil ist, dass Nginx sich primär um das Bereitstellen von statischen Inhalten, wie JavaScript, css etc. kümmert und Gunicorn die dynamischen Anfragen für Nginx renderet. Das Frontend wird ebenso als separater Nginx Server zur Verfügung gestellt, das hat den Vorteil, dass man Frontend und Backend klar separiert und durch eindeutiges URL-Patterning klare Zuständigkeiten definieren kann. So lassen sich Frontend und Backend auch simpel auf unterschiedliche VMs aufteilen.

Zusätzlich wird noch eine SQL Datenbank in Form von PostgreSQL für das persistente Speichern von Daten verwendet und eine memcached Instanz für das caching von Datenbankanfragen. Es werden momentan zwei Amazon Cloudinstanzen verwendet. Eine Amazon EC2<sup>12</sup> Instanz und eine Amazon RDS Instanz<sup>13</sup>. Auf der EC2 Instanz wird alles bis auf die Datenbank realisiert, wofür dann die RDS Instanz hinzugenommen wird. Bei Änderung an statischen Inhalten muss noch immer die Subroutine `python /path/to/manage.py collectstatic` oder `python3 /path/to/manage.py collectstatic` ausgeführt werden.

---

<sup>12</sup>Informationen unter <https://aws.amazon.com/ec2/>

<sup>13</sup>Informationen unter <https://aws.amazon.com/rds/>

---

```

1 #!/bin/bash
2 while read oldrev newrev ref
3 do
4     if [[ $ref =~ .*/1000-frontend$ ]]; then
5         echo "Ref $ref received. Deploying
6             $ref to production..."
7         git --work-tree=/home/django/
8             html_frontend --git-dir=/home/
9                 django/deploy\_frontend.git
10                checkout $ref -f
11            cd /home/django/html_frontend/
12                frontend/
13            npm install
14            npm run tsc
15        else
16            echo "Ref $ref successfully received.
17                Doing nothing: only 1000-
18                    frontend may be deployed"
19        fi
20    done

```

---

Abbildung 13: Frontend deployment Skript

### 10.1.2 Git Repositories und Hooks

Backend als auch Frontend wurden in separate git repositories aufgeteilt. So konnten beide Teams bequem die vom Gitlab zur Verfügung gestellten Oberflächen nutzen, um zum Beispiel für jedes Issue automatisch einen eigenen Branch und Merge Requerst erstellen zu lassen. Jeder Push auf den Master Branch auf dem frontend git repository hat folgendes Skript aufgerufen:

Jeder Push auf das Backend repository auf dem Server hat über das post-receive hook, folgende verkettete Skripts aufgerufen:

## 10.2 Optimale Konfiguration

Die für eine bessere Infrastruktur notwendigen Veränderungen werden im Anschluss erläutert.

```
1 #!/bin/sh
2 sh ~/deployment\_database\_backup.sh
3 sh ~/deployment\_script\_production.sh
```

Abbildung 14: Backend deployment post-receive hook

```
1 #!/bin/sh
2
3 DIR=/home/django/donaalda
4
5 echo "Starting database backup"
6
7 . $DIR/advisorEnv/bin/activate
8
9 python3 $DIR/manage.py dumpdata --natural-foreign --
  natural-primary --format=json --indent=2 -e auth.
  Permission app auth sessions -o /home/django/
  database\_backup/$(date --iso-8601=seconds).json
10
11 deactivate
12
13 echo "Finished database backup"
```

Abbildung 15: deployment\_database\_backup.sh Backup von der Datenbank  
in eine JSON Datei.

```
1 #!/bin/sh
2
3 DIR=/home/django/donaalda
4
5 . $DIR/advisorEnv/bin/activate
6
7 pip3 install -r $DIR/requirements_unix.txt
8
9 python3 $DIR/manage.py collectstatic --no-input
10 python3 $DIR/manage.py migrate --no-input
11 python3 $DIR/manage.py flush --no-input
12 python3 $DIR/manage.py loaddata initial\_data
13
14 deactivate
15
16 sudo /home/django/restart\_server.sh
```

Abbildung 16: deployment\_script\_production.sh Löschen der Datenbank und neu laden aus der vorhandenen JSON Datei.

```
1 #!/bin/sh
2
3 service gunicorn restart
4 service nginx reload
5 service nginx restart
```

Abbildung 17: restart\_server.sh Neustart des NGINX-Servers

### **10.2.1 Anwendungsserver**

Zukünftig wäre anzupreisen, dass der Gunicorn-Anwendungsserver auf eine Amazon Beanstalk Instanz<sup>14</sup> ausgelagert wird. Dies würde fast sofortige Antworten auf Peak-Traffic ermöglichen, da diese in nur wenigen Minuten replizier- und abbaubar sind. Dies wäre sogar automatisierbar, sodass der Service unabhängig von aktuellem Traffic immer erreichbar wäre. Zusätzlich könnte man so die aktuelle Konfiguration noch etwas weiter modularisieren, um so zu einem noch ausfallsichereren Produkt auszubauen.

### **10.2.2 Nginx**

Weiterhin wäre es sinnvoll, auf ein Nginx+ AMI(Amazon Machine Image)<sup>15</sup> für alle als Webserver verwendeten EC2 Instanzen zu setzen. Diese wären dann auch je nach Traffic automatisch replizier- und abbaubar.

### **10.2.3 Statische Inhalte**

Die Verteilung der statischen Inhalte würde man durch Amazon S3<sup>16</sup> durchführen, da dies ein Service ist, der speziell auf diesen Einsatzzweck zugeschnitten ist.

### **10.2.4 Database-Backend**

Man würde die vorhandene Amazon RDS Instanz so einrichten, dass ein zumindest manuell eingeleiteter automatischer Failover in einer Master/Slave-Architektur möglich wäre<sup>17</sup>.

---

<sup>14</sup>Informationen unter <https://aws.amazon.com/elasticbeanstalk/>

<sup>15</sup>Informationen unter <https://www.nginx.com/resources/admin-guide/setting-nginx-plus-environment-amazon-ec2/>

<sup>16</sup>Informationen unter <https://aws.amazon.com/s3/>

<sup>17</sup>Informationen unter <http://dba.stackexchange.com/questions/62894/how-to-configure-postgresql-hotstandby-automatic-failover>

### **10.2.5 Cache-Backend**

Als Cache-Backend wäre es sinnvoll auf Amazon ElastiCache<sup>18</sup> zu setzen. Hier werden alle Cache-Daten, wie Sessions und häufig verwendete Querysets, entweder über Redis<sup>19</sup> oder Memcached<sup>20</sup> verwaltet.

### **10.2.6 Load-Balancer**

Es wäre auch sinnvoll dem ganzen System einen Load-Balancing Service<sup>21</sup> vorzuschalten, um die Verteilung des Traffics intelligent managen zu können.

## **10.3 Admin**

Das Admin Panel ist zur vereinfachten und übersichtlichen Verwaltung der Datenbasis da. Hierbei werden den jeweiligen Nutzern nur die für sie freigeschalteten Objekte angezeigt. So können alle am Advisor interessierten Parteien ihre Szenarien und Produkte übersichtlich verwalten und aber auch neue erstellen. Somit ist sichergestellt, dass das System auch für technisch weniger versierte Nutzer einfach und schnell zu benutzen ist.

### **10.3.1 Startseite Verwaltung**

Nach erfolgreicher Anmeldung wird dem Nutzer eine Übersicht aller ihm zur Veränderung freigegebenen Objekte präsentiert siehe Abbildung 18.

### **10.3.2 Änderungen vornehmen**

Ausgehend von dieser Ansicht kann der Nutzer die unterschiedlichen Entitäten und Relationen verwalten. Dafür muss er lediglich das zu Verändernde Element auswählen. Anschließend werden alle Entitäten dieses Typs listenartig dargestellt siehe Abbildung 19. Hier wird dem Nutzer eine Liste aller

---

<sup>18</sup>Informationen unter <https://aws.amazon.com/elasticache/>

<sup>19</sup>Informationen unter <http://redis.io/>

<sup>20</sup>Informationen unter <https://memcached.org/>

<sup>21</sup>Informationen unter <https://aws.amazon.com/elasticloadbalancing/>

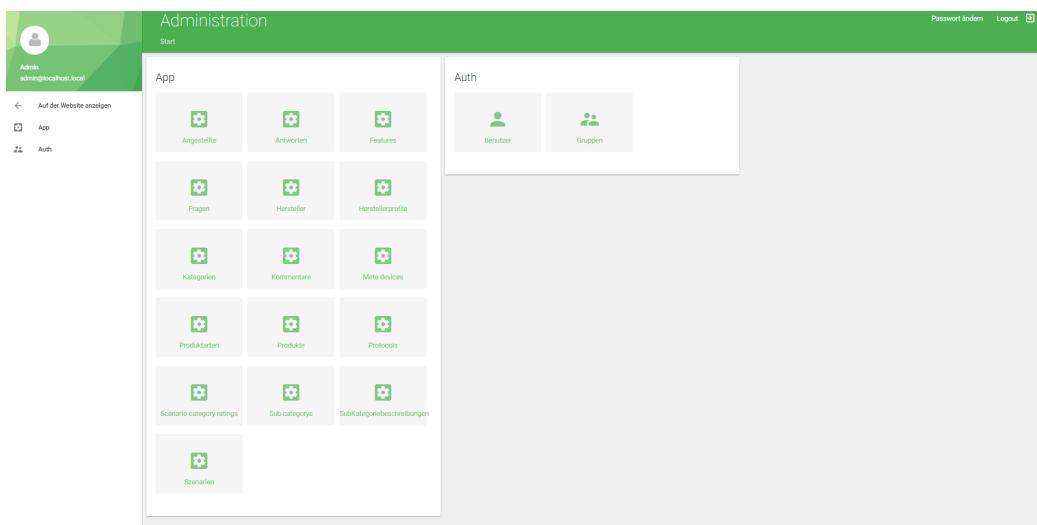


Abbildung 18: Willkommensseite

Entitäten dieses Typs präsentiert. Je nach Definition der jeweiligen Ansicht werden auch unterschiedliche Attribute mit dargestellt. In dieser Übersicht können mehrere Elemente über die Checkboxen markiert werden, um diese dann direkt zu löschen. Andererseits können auch neue Elemente über das rote Plus Symbol in der unteren rechten Ecke erstellt werden. Hier werden je nach zu erstellendem Objekt auch unterschiedliche Attribute zur Eintragung abgefragt.

The screenshot shows an admin dashboard with a sidebar and a main content area.

**Sidebar (Left):**

- User profile: Admin, admin@localhost.local
- Navigation links:
  - Auf der Website anzeigen
  - App (selected)
  - Angestellte
  - Antworten
  - Features (selected)
  - Fragen
  - Hersteller
  - Herstellerprofile
  - Kategorien
  - Kommentare
  - Meta devices
  - Produktarten
  - Produkte
  - Protocols
  - Scenario category ratings
  - Sub categories
  - SubKategoriebeschreibungen
  - Szenarien
  - Auth

**Main Content Area:**

## Features

Start > Features

AKTION
0 von 100 ausgewählt
<b>AUSFÜHREN</b>

The list of features is as follows:

- Feature
- asdasdasd
- controls remotely\_AN
- controls remotely\_AUS
- controls remotely\_Regler
- controls remotely\_Schloss
- notifies\_Anruf tätigen
- notifies\_Audio tätigen
- notifies\_sendEmail
- notifies\_sendPushNotif
- notifies\_sendSMS
- notifies\_visuelles
- operates\_Backofen
- operates\_Fenster
- operates\_Heizung
- operates\_Herd
- operates\_Hifi
- operates\_Hometrainer

A red button with a plus sign (+) is located at the bottom right of the feature list.

Abbildung 19: Admin Listenübersicht einer Entität/Relation

# 11 Infrastruktur Frontend

Im Folgenden werden einige Technologien erläutert und ggf. die Vorteile und deren begründeten Einsatz beschrieben. Auch wird kurz auf genutzte, externe Bibliotheken eingegangen.

## 11.1 Angular 2

Angular 2 ist ein Javascript Framework, dass von Google entwickelt und im Oktober 2016 veröffentlicht wurde. Es eignet sich hervorragend um Single-Page-Applikation zu bauen. Wie viele andere JS-Frameworks ist es nach dem MVC Schema aufgebaut, bei welchem die Software in drei Komponenten eingeteilt ist:

- Model: enthält die zu darstellenden Daten
- View: ist für die Darstellung der Daten für den Nutzer und Entgegennahme von Benutzerinteraktionen zuständig
- Controller: enthält die Programmlogik

Dieses Pattern erleichtert eine Erweiterung oder Änderung des Codes für die Weiterführung des Projektes und ermöglicht eine bessere Weiterverwendung von bestehendem Code.

Angular 2 setzt diese Trennung mittels Folgenden Architekturkomponenten um:

- Component: entspricht dabei dem Controller
- Service: entspricht dem Model
- Html-Pages: entspricht der View (welche spezifische Angular II Attribute enthält)

Die Entscheidung für dieses Framework lag nicht nur in der überzeugenden Umsetzung des genannten Architekturmusters, sondern es gab eine breite Übereinstimmung praktische Erfahrung mit diesem Framework sammeln zu wollen. Zu guter Letzt nutzt es Typescript (siehe 11.3 )als Programmiersprache, statt, wie viele Webframeworks, Javascript.

## 11.2 npm

Der Node Package Manager (npm) ist ein Paketmanager für Javascript. Es bietet die Möglichkeit den Entwicklern untereinander Code auszutauschen, wiederzuverwenden und geteilten Code einfach zu aktualisieren. Im Folgenden werden die wichtigsten Pakete, die in DONAALDA verwendet werden, kurz erläutert.

- RxJS (Version: 5.1.0): RxJs ist eine JavaScript-Bibliothek und bietet die Möglichkeit zur reaktiven Programmierung. Dies bedeutet, dass Prozesse und Funktionen asynchron und evenbasiert ablaufen können. Daten-Sequenzen werden dabei als Datenströme aus Dateien oder Web-Anfragen dargestellt. Wichtig für die Entwicklung unseres Systems waren dabei vor allem die Funktionen für die asynchronen Serverabfragen.
- Hammerjs (Version: 2.0.8): Hammerjs ist eine Bibliothek die, dafür sorgt, dass Gesten auf einem Touchdisplay richtig erkannt und umgesetzt werden (drehen, zoomen etc.). Dieses ist vor allem bei mobilen Geräten, wie Tablets oder Smartphones wichtig.
- @angular: Diese Bibliothek enthält alle, für Angular II wichtigen, Ressourcen und Funktionen.

## 11.3 Typescript

Typescript (Version: 2.2.0) ist eine Programmiersprache, welche von Microsoft entwickelt wurde und nach Javascript kompiliert. Dies bedeutet, dass in den Typescript-Code auch Javascriptcode eingefügt werden kann und bisherige Javascriptbibliotheken auch in Typescript Verwendung finden können.

Die Vorteile, die Typescript mit sich bringt, sind die erweiterten Technologien, welche Javascript nicht bietet. So können Klassen und Interfaces genutzt werden, sowie Konzepte der Vererbung und statische Typisierung. Dies erleichtert das Programmieren im Vergleich zu herkömmlichen Javascriptprogrammen und bietet die Möglichkeit objektorientiert zu programmieren

## **11.4 JQuery**

JQuery (Version: 3.1.1) ist eine Javascriptbibliothek, welche weitere Funktionen (bspw. Eventhandling, Animation und Serveranfragen) zur Veränderung des HTML-Dokumentes zur Verfügung stellt. Dieses Bibliothek funktioniert browserübergreifend. Viele frei zugängliche Javascript Bibliotheken nutzen JQuery für ihre Animationen oder Funktionalitäten. Dies ist beispielsweise bei Bootstrap der Fall. Damit deren Funktionalität gewährleistet ist, hat die JQuery Bibliothek Einzug in DONAALDA gefunden.

## **11.5 Bootstrap**

Bootstrap (Version: 3.3.6) ist ein freies, open-source CSS-Framework, welches Designvorlagen für Formulare, Buttons, Navigations- und andere Oberflächenelemente bietet. Dafür nutzt es neben CSS aber auch Javascript und bietet die Möglichkeit optionale Javascript Erweiterungen benutzen zu können. Der große Vorteil liegt in der Unterstützung des responsive Designs. Das bedeutet, dass sich das Frontend dynamisch, je nach Gerätetyp und Auflösung, anpassen kann. Somit ist eine benutzerfreundliche Umgebung auf allen Geräten garantiert und es muss nicht für jede Auflösung eine neue Ansicht erstellen.

## **11.6 Material Kit**

Material-Kit ist ein UI (User Interface) Kit, welches Bootstrap Oberflächenelemente verwendet, diese aber design-technisch verändert und an das sogenannte Material Design angepasst hat. Dieses Design ist für seinen Minimalismus bekannt und verwendet Animationen und Schatten um Tiefeneffekte und Ebenen zu erzeugen. Damit kann dem Nutzer ein Interaktionsfeedback, einen Überblick über weitere Informationen oder Hinweise zu aktiven bzw. inaktiven Bereichen gegeben werden.

In diesem Kit sind Elemente, in Bezug auf Farbe, Größe und anderen Designaspekten, so aufeinander abgestimmt, dass man während des Programmierens sich keine weitere Gedanken über mögliche unstimmige Designs machen muss. Neben den bestehenden Oberflächenelementen aus Bootstrap werden auch weitere Elemente hinzugefügt, welches ein einfacheres Gestalten und Anordnen der Elemente ermöglicht.

## 12 Aufbau

Die Web-App ist in zwei Teile aufgeteilt. Zum Einen soll der Kunde auf der Landing-Page über den Advisor und einige Ansätze im Smart Home informiert werden. Er bekommt einen Überblick über die Funktionalitäten des Advisors. Der zweite Teil ist der Advisor selbst. Jeder der vier Schritte entspricht einer Ansicht im Advisor. Über Grundfragen (1) soll der Kunde eine Präferenz für sein Smart Home abgeben. Nach der Auswertung der Antworten erhält der Kunde eine Übersicht über passende Szenarien (2). Diese Szenarien, die bestimmte Anwendungsfälle im Smart Home beschreiben, geben eine Übersicht, welche Möglichkeiten die Anschaffung und Vernetzung von Smart Home Produkten bietet. Zu jedem Szenario werden nun Produkte zugeordnet, die die Funktionalitäten des Szenarios erfüllen können. Hierbei werden im Backend alle hinterlegten Produkte miteinander verglichen und das optimale Produktset auf die passenden Funktionalitäten abgestimmt. Der Benutzer kann sich dann für Szenarien die ihn interessieren entscheiden und diese in einen Warenkorb legen. Danach kann der Kunde sich über die abgestimmten Produkte informieren (3). Neben den Produktspezifikationen und Informationen zu der Produktauswahl hat er die Möglichkeit die Produktauswahl zu modifizieren. Schließlich erhält der Nutzer eine Übersicht über alle Produkte und kann sich die Liste der Produkte ausdrucken um sie zu erwerben (4).

## 13 Komponenten

Der Advisor hat folgende Komponentenstruktur

**Index Page** 13.1 Eine Art Landing Page, die den Nutzer einen Start in den Beratungsprozess leitet

**Onboarding** 13.2 Grundfragen zur Ermittlung von Nutzerpräferenzen

**Szenarioauswahl** 13.3 die Frontend-Schnittstelle zur Darstellung und Filtern der Ergebnisse passend zu den Nutzerpräferenzen

**Übersicht** 13.5

DONAALDA's Beratungsprozess ist grob in 4 Phasen aufgeteilt, deren Interaktionsabläufe ausführlich in 5 beschrieben wird. Die Beschreibung hier dient lediglich der Rechtfertigung des Designs im Bezug auf User Experience und User Interaction.

### 13.1 Index-Seite

Die Indexseite ist die dem Nutzer als Erstes präsentierte Seite vor dem eigentlichen Beginn des Beratungsprozesses, auf welcher der Nutzer grundlegende Informationen über die ihn zu erwartende Schritte bezüglich Smart Home Konfiguration präsentiert und erklärt bekommt. Aufgebaut wurde diese Seite als Single Page Application mit Hilfe von Angular2 TypeScript, HTML, Bootstrap und CSS.

Die Idee ist, den Benutzer beim Start der Application in eine packende Reise durch die Smart Home Welt eintauchen zu lassen. Deswegen gleich am Anfang wurde ein Startappel an den Nutzer formuliert mit dem Advisor Start-Button. Allgemein besteht die Indexseite aus vier Teilen. Der erste Teil informiert den Benutzer über vier Advisor-Schritte (Bild 1), die ihn auf dem Weg zu seinem Smart Home erwarten. Diese sind:

- Grundfragen - Szenarioauswahl - Produktauswahl - Übersicht

## Dein Smart Home in nur vier Schritten

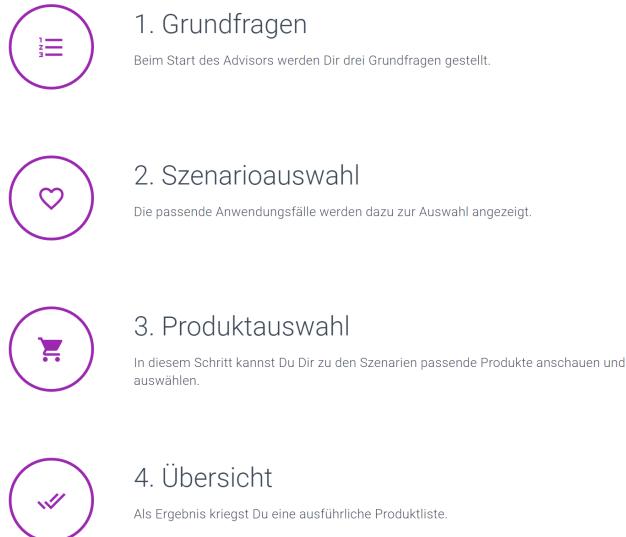


Abbildung 20: Darstellung der vier Beratungsphasen

Der nächste Teil präsentiert DONAALDA als personalisierten Berater (Bild 2) und beschreibt besondere Vorteile für den Nutzer in Form von „Info“- und „Wie“-Angaben. Anzumerken ist, dass für diese beiden Teile der Indexseite die gleiche Art von Icons wie im ersten Advisor-Schritt „Onboarding“ benutzt wurden, um das Design einheitlich zu halten.

Ein wichtiger Aspekt der Seite war auch die Partner und Hersteller anhand deren Logos zu repräsentieren. Damit wurden die Partner geehrt und den Nutzern bekannt gemacht welche Hersteller sie als Vorschlag vom Advisor bekommen. Weil als großer Vorteil von DONAALDA, vor allem, die Vielfalt von den Gerätemarken und deren Gesamtkonfiguration gesehen wird. Ein Zwischenteil der Seite ist das Bild mit einem Parallax-Effekt, das auch den Benutzer dazu motiviert, den Advisor sofort zu starten, nachdem er schon etwas besser informiert ist. Als letzter Teil der Seite beinhaltet eine kurze Information über die Auftraggeber des Projektes und ihre Repräsentation mit Logos und Links zu deren Webseiten.

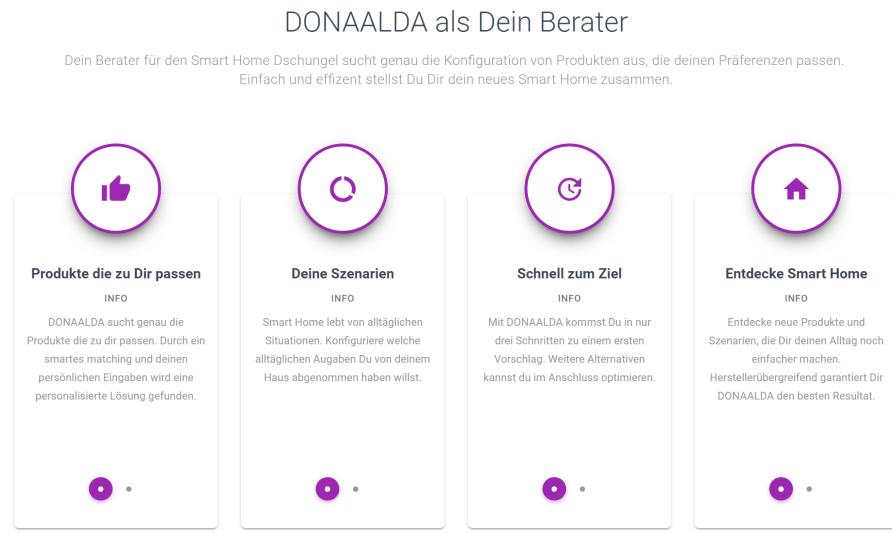


Abbildung 21: Vorteile von DONAALDA als Berater

## 13.2 Onboarding

Das Onboarding besteht aus 4 internen Schritten, den 3 Grundfragen bzgl. Szenariopräferenz, Produktpräferenz & Baumaßnahmenpräferenz und der Übersicht der 3 Fragen inklusive Antworten. Für diese 4 internen Schritte haben wir uns für einen Wizard entschieden, in welchem man sequentiell jeweils einen Schritt absolviert. Es wird in einer Art Schrittstrahl stets angezeigt, in welchem Schritt man sich befindet und wie viele es insgesamt gibt. Die Schritte werden sowohl farblich als auch mit einem Icon hervorgehoben, wobei sich absolvierte Schritte, noch kommende Schritte und der Schritt, in dem man sich gerade befindet, voneinander unterscheiden. Alle Fragen und vom Nutzer einstellbare Optionen haben erklärende Tooltips, die sowohl für Desktop-, Handy- und Tabletnutzer nutzen können. Buttons sind stets sichtbar, können aber bei notwendigen Antworten eventuell gesperrt sein. Die Übersicht ist für den User notwendig, um ähnlich wie bei Bestellungen nochmals seine Angaben in einer kompakten Zusammenfassung zu prüfen. Bei Nachkorrektur kann zum entsprechenden Schritt gesprungen und nachgebesert werden.

## 13.3 Szenarioauswahl

In der Szenarioauswahl war uns die Übersichtlichkeit besonders wichtig, da in dieser Phase die Unterkomponenten im Vergleich zu den anderen Phasen eine hohe Komplexität haben. Links wird die Hausansicht und rechts die restlichen Unterkomponenten angezeigt.

### 13.3.1 Szenarioliste

Die Szenarioliste nimmt die rechte Seite der Szenarioauswahl ein, wenn die Filter nicht ausgeklappt sind. In dieser Unterkomponente werden bis zu 6 Szenarien gleichzeitig gezeigt. Jedes dieser Szenarien gibt dem Nutzer mit einem Bild und Titel einen ersten Eindruck. Es ist möglich, das Szenario entweder direkt in den Warenkorb hinzuzufügen oder mit Aktivieren eines Infobuttons wird ein zusätzliches Panel aufgeklappt, das mehr Informationen von dem Szenario bietet. Unter Anderem sind es eine Kurzbeschreibung, die Übereinstimmung des Szenarios mit den vom Nutzer angegebenen Szenario-präferenzen, eine ausführliche Beschreibung und die enthaltenen Produkttypen, die notwendig sind, um das Szenario zu implementieren. Bei der Auswahl eines Szenarios wird das Szenario in den Szenario-Warenkorb gelegt und verschwindet aus der Liste der vorgeschlagenen Szenarios. Neue Vorschläge werden berechnet, eine genauere Beschreibung ist in ref Interaktivitätsablauf zu lesen.

### 13.3.2 Filterfunktion

Die Filterfunktion ist zu Beginn des Szenarioauswahl nicht ausgeklappt. Hier werden wie in ref Interaktivitätsablauf beschrieben Filter für Anfragen an das Backend ausgewählt und neue Vorschläge können angefragt werden.

### 13.3.3 Hausansicht

Die Hausansicht bietet dem Nutzer eine visuelle und interaktive Repräsentation seines zukünftigen Smart Homes. Dies wird mithilfe eines SVG umgesetzt, welches man mithilfe von DOM-Manipulation interaktiv wirken lassen kann. Produkttypen der ausgewählten Szenarien werden farblich hervorgehoben und zusätzlich unterhalb des Hauses noch einmal explizit aufgelistet.

## 13.4 Produktauswahl

Nachdem der Kunde seine Szenarien ausgewählt hat, kommt er in den dritten Schritt des Advisors, die Produktauswahl.

Hier sieht er zum ersten Mal die Produkte die der Advisor anhand der Grundfragen und der Funktionalitäten aus den gewählten Szenarien ausgewählt hat, in einer ansprechenden Übersicht. Neben dem Produktbild, einer Kurzbeschreibung und dem Preis, kann der Kunde sich weitere Spezifikationen der Produkte in einem Detailbereich anzeigen lassen.

Ziel des Produktauswahlschrittes ist es, dem Kunden mehrere Möglichkeiten der Modifikation der Produktauswahl anzubieten. Allerdings ist das initiale Produktset schon das optimale Produktset aus Sicht der Präferenzen der Benutzers. Nach Änderungen der Produkten werden die Abhängigkeiten der gesamten Produktauswahl neu berechnet. Es wird weiterhin die Funktionsabdeckung zu den Szenarien gewährleistet sein.

Berücksichtigt wird außerdem, welche Gateways notwendig sind damit alle Produkte eingebunden werden können.

- Jedes Produkt kann durch andere Produkte ausgetauscht werden, so lange es die gleiche Funktionalität des zu ersetzenen Produkts bietet (Beispiel: Bosch Waschmaschine durch Miele Waschtrockner).
- Jedes Produkt kann gesperrt werden, sodass es beim Neuberechnen der Produktauswahl nicht aussortiert wird.

Während der Produktauswahl bzw. dem Ändern der Produktauswahl können so gewollte Nebeneffekte beobachtet werden. Diese sind Bestandteil der Zuordnung der Produkte.

Wird ein Produkt ausgetauscht, werden die Abhängigkeiten zu den anderen, zuvor vorgeschlagenen Produkten geprüft. Dabei können 0..n Produkte ebenfalls ausgetauscht werden. Zudem kann ein anderes Gateway oder mehrere Gateways nun vorgeschlagen werden, da sich die Kompatibilität eventuell geändert hat.

Auch vermeintlich unpassende Produkte können in der Produktauswahl erscheinen, da die Produkte immer aufgrund der bereitgestellten Funktionalitäten vorgeschlagen werden und ein Produkt die Funktionalität unterstützt. Ein Beispiel dafür: Ein Szenario zum Ausschalten von Geräten und dem Vorschlag eines Kühlschranks. Ein ausgeschalteter Kühlschrank macht wenig Sinn. Allerdings hat der vorgeschlagene Kühlschrank eine Funktion für Stand-By um bei Zeiten geringer Benutzung die Kühlleistung herabzusetzen.

## 13.5 Übersicht

Als letzte Seite bekommt der Nutzer noch einmal eine Gesamtübersicht über alle zuvor getroffenen Entscheidungen (Szenarioauswahl und Produktauswahl) und die Möglichkeit sich eine Übersicht davon auszudrucken.

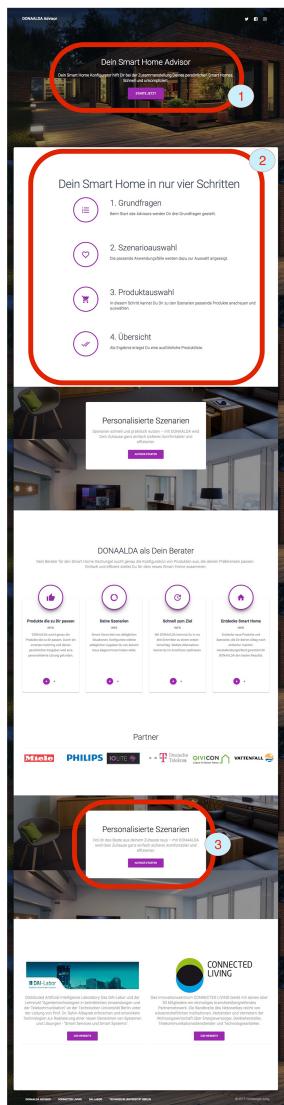
Der Nutzer erhält untereinander aufgeregnet eine Übersicht aller benötigten Produkte, die all seine gewünschten Szenarien umsetzen können. Dabei werden ihm erneut Informationen, wie Preis, Hersteller, Seriennummer, Energieverbrauch, Produktbeschreibung und Bilder gegeben um bei Bedarf weitere Recherche über dieses Produkt betreiben zu können. Darüber hinaus wird noch einmal geklärt zu welchen Szenarien dieses Produkt gehört. Denn falls dem Nutzer ein Gerät zu teuer ist und es nicht kaufen möchte kann er nachvollziehen, welche Szenarien dann nicht umgesetzt werden können.

Auf der rechten Seite befindet sich eine Kurzübersicht aller Geräte mit dem Namen, Hersteller und Preis und der Gesamtpreis aller Geräte summiert. Diese Kurzübersicht ist vor allem für Nutzer die sich einen Kostenüberblick über alle benötigten Komponenten in ihrem Smart-Home verschafft wollen.

### **13.5.1 Druckansicht**

Die Druckansicht ist über einen Button, weit oben auf der Seite erreichbar. Dieser öffnet einen neuen Tab, bei welchem unnötige Designelemente entfernt werden und eine druckfreundliche HTML-Seite mit den aus 13.5 genannten Informationen (Hersteller, Preis etc.) angezeigt wird, die über die Druckfunktion des Browsers gedruckt werden kann.

## 14 Anleitung - Frontend



1) Wireframe:  
der Button zum Advisor starten hebt sich durch Farbe und Position hervor

2) Visuelle Erklärung:  
durch Unterscheidung von Headings und Text kann der Nutzer die Information schnell überfliegen und das relevante rauslesen.

Für weiteren Details eignen sich die Abschnitte drunter.

3) Navigationsdesign:  
Um wieder den Nutzer visuell auf die wesentliche Funktion des Advisors zu lenken

### Fragen zum Szenario Matching

The screenshot shows a navigation bar at the top with tabs: Kategorien, Produkte, Baumaßnahmen, and Übersicht. Below it is a question: "Wie wichtig sind Dir die einzelnen Kategorien?". There are four circular icons representing categories: Gesundheit, Energie, Sicherheit, and Komfort. Each icon has a numerical scale from 1 to 5 below it. A tooltip "Was wichtig und Dein Szenario mit einem Schwerpunkt auf Sicherheit?" is shown next to the Sicherheit icon. A "WEITER" button is at the bottom right.

- 1) Navigationsleiste  
Informiert, wo sich der Nutzer im Beratungsprozess befindet.
- 2) Antwortmöglichkeiten auf einer Frage
- 3) Likert Skala zur Angabe der Präferenz der jeweiligen Kategorie. Die Skalen sind von einander unabhängig verschiebbar. Beim Verschieben erscheint die genaue Präferenzzahl 1.
- 4) beschreibende Ikonen (nicht-klickbar) für die jeweilige Kategorie mit Tooltips 5).

### Fragen zum Produkt Matching

The screenshot shows a navigation bar with tabs: Kategorien, **Produkte**, Baumaßnahmen, and Übersicht. Below it is a question: "Was ist Dir bei den Produkten am Wichtigsten?". There are three circular icons representing product features: Effizienz, Erweiterbarkeit, and Preis. A tooltip "Bei der Suche wird auf effiziente Produkte und Produkte mit Preis gelegt" is shown next to the Preis icon. A "WEITER" button is at the bottom right.

- 1) klickbare Ikonen zur Auswahl hier soll man sich die am meisten zutreffende Auswahl klicken. Das hilft dem Backend, die Produkte zu matchen
- 2) Um in das nächste Bildschirm zu kommen, braucht man die aktuelle Frage zu beantworten.
- 3) NavBar zeigt den aktuellen Stand. Eine beantwortete Frage wird mit einem Haken versehen. Die aktuelle Frage ist mit den "..." hervorgehoben.

### Übersicht

The screenshot shows a summary screen with three completed sections: 1) "Wie wichtig sind Dir die einzelnen Kategorien?", showing preferences for Gesundheit (6), Energie (10), Sicherheit (2), and Komfort (2). 2) "Was ist Dir bei den Produkten am Wichtigsten?", showing a checkmark for "ERWEITERBARKEIT". 3) "Willst Du Baumaßnahmen bei Dir zu Hause durchführen?", showing a checkmark for "JA". A "ZU DEN Szenarien" button is at the bottom right.

- 1) Eine abgeschlossene Befragung

## Szenarioauswahl

The screenshot shows the 'DONAALDA Advisor' software interface, specifically the '2. Szenarioauswahl' (Scenario Selection) screen. The interface is divided into several sections:

- Top Bar:** Shows the title 'DONAALDA Advisor' and the current step '1. Grundfragen > 2. Szenarioauswahl'.
- Left Column:** Displays a floor plan of a house labeled 'Den Smart Home'. Below it is a photograph of a modern wooden house at night with illuminated exterior lights.
- Middle Column:** A large section titled 'Wähle Deine Szenarien aus' (Select your scenarios). It lists two scenarios with icons and prices:
  - 'Warnung vor Feuer' (Fire warning) at €252.88
  - 'Kindersicherung Unterwegs' (Child safety on the go) at €5.064.69
- Right Column:** A detailed description of the 'Warnung vor Feuer' scenario:
  - Beschreibung:** A narrative about a kitchen fire and how the system alerts the fire department and sends a notification.
  - Produkttypen:** Icons for Smart Home, Rauchmelder (Smoke detector), and Bewegung (Motion).
  - Unterkategorien:** Icons for 'Warnung vor Wasser und Feuer' (Water and fire warning) and 'Rauchmelder'.
  - Product List:** A grid of products:
    - Steady Standby (€547.98)
    - Checken von Verfallsdaten (€2.188.99)
    - Schutz vor Einbruch (€696.88)
    - Benachrichtigung für Wäsche (€1.566.69)

- 1) Navigationsleiste mit aktuellem Schritt
- 2) Drop-Down Menü zeigt die ausgewählten Szenarien
- 3) Drop-Down Menü zeigt die Produkte aus allen Szenarien
- 4) Szenariofilter
- 5) Szenario und die entsprechenden Produkte (der erste Auswahlvorschlag) hinzufügen
- 6) Ein Szenario, das gerade betrachtet wird
- 7) Szenariobeschreibung schließen
- 8) Bei der Auswahl eines Szenarios werden die entsprechenden Produktkategorien hervorgehoben
- 9) Die hervorgehobenen Kategorien werden hier aufgelistet
- 10) Weiter zur anzeigen der Produkte
- 11) weitere Szenarien laden
- 12) Szenariobeschreibung

## Komponenten in der Szenarioauswahl



Beispiel für eine markierte Produktauswahl

Filtern nach Produkttypen			
<input type="checkbox"/> Alarmsirene	<input type="checkbox"/> Bewegungsmelder	<input type="checkbox"/> Blutdruckmessgerät	<input type="checkbox"/> CO2-Sensor
<input type="checkbox"/> Dunstabzugshaube	<input type="checkbox"/> Dusche	<input type="checkbox"/> Fitnessgerät	<input type="checkbox"/> Geschirrspüler
<input type="checkbox"/> Heizungsteuerung	<input type="checkbox"/> HiFi	<input type="checkbox"/> Home Bot	<input type="checkbox"/> Kaffeemaschine
<input type="checkbox"/> Kamera	<input type="checkbox"/> Kochfeld	<input type="checkbox"/> Kontaktssensor	<input type="checkbox"/> Kühlschrank
<input type="checkbox"/> Lampe & Bridge	<input type="checkbox"/> Leuchtmittel (vernetzbar)	<input type="checkbox"/> Lichtsteuerung & Dimmer	<input checked="" type="checkbox"/> Luftfeuchtigkeitssensor
<input type="checkbox"/> Ofen	<input checked="" type="checkbox"/> Photovoltaikanlage	<input type="checkbox"/> Radio	<input type="checkbox"/> Rauchmelder
<input type="checkbox"/> Rolladenaktuator	<input type="checkbox"/> Router	<input type="checkbox"/> Sauerstoffmessgerät	<input type="checkbox"/> Schaltaktor
<input type="checkbox"/> Smarter Wecker	<input type="checkbox"/> Smart-Fernseher	<input type="checkbox"/> SmartHome Basisgerät	<input type="checkbox"/> SmartHome Gateway
<input type="checkbox"/> Smartphone / Tablet	<input type="checkbox"/> Smart Plug	<input type="checkbox"/> Smart Safe	<input checked="" type="checkbox"/> Smart TV
<input type="checkbox"/> Toaster	<input type="checkbox"/> Solaranlage	<input type="checkbox"/> Stromzähler	<input type="checkbox"/> Temperatursensor
<input checked="" type="checkbox"/> Waage	<input type="checkbox"/> Trockner	<input type="checkbox"/> Türklingel	<input type="checkbox"/> Uhr
	<input type="checkbox"/> Waschmaschine	<input type="checkbox"/> Wassersensor	<input type="checkbox"/> Wetterstation

Produktfilter

## Produktauswahl

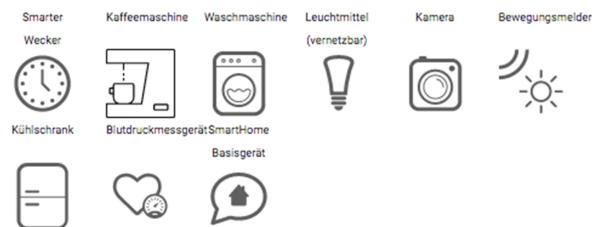
Diese Produkte wurden ausgewählt. Bist Du mit einem Produkt nicht zufrieden, kann Du es durch ein anderes ersetzen, das den selben Funktionsumfang bietet und weiterhin mit Deinen Szenarien kompatibel ist

<b>1</b>	<b>2</b>	<b>3</b>
Beddi Beddi Smart Alarm Wecker Smarter Wecker €99.99	Fibaro Fibaro Smoke Sensor Rauchmelder €52.89	Aeotec by Aeon Labs Aeon Labs MultiSensor 6-in-1 Bewegungsmelder €44.90
IOLITE IOLITE Gateway SmartHome Basisgerät €199.99	Smarter Wifi Kaffeemaschine mit Mahlwerk Kaffeemaschine €347.99	Bosch WAWH8690 Waschmaschine mit intelligenter Dosierautomatik €449.99

- 1) die erste Produktvorschläge, die anhand der Szenarien gematched sind.
- 2) ein Produkt, der die Szenarien entspricht und mit allen anderen Produkten kompatibel ist.
- 3) wenn man diesen Produkt für die finale Auswahl mitbehalten möchte, kann man es "schließen"

## Symbole für Produktkategorien

Produkttypen, die in Deinen Szenarien vorkommen:



# 15 Tests

Zur Qualitätssicherung wurden diverse automatisierte und manuelle Tests verwendet. Im Folgenden wird auf die eingesetzten Testverfahren seitens Backend und Frontend eingegangen und beschrieben welche Teile der Software durch die jeweiligen Tests abgedeckt werden.

## 15.1 Backend

Während der Entwicklung war es uns wichtig, dass die Frontend und Backend Gruppen so unabhängig wie möglich voneinander arbeiten können. Ein wichtiges Werkzeug um dieses Ziel zu erreichen war die Swagger API Beschreibung (siehe Sektion 8.3), die durch das bereitstellen von Beispielaufrufen viele unnötige Diskussionen in den Gesamtgruppentreffen verhindert hat.

Um Sicherzustellen, dass die API Beschreibung immer aktuell und verwendbar ist, haben wir einen integration Test geschrieben, der jede in Swagger definierte Schnittstelle mit den jeweiligen Beispieldaten anfragt und überprüft, ob der Server mit einem Erfolgs Statuscode antwortet. Dadurch konnten viele potentielle Fehler gefunden und behoben werden bevor die Frontendgruppe je auf sie gestoßen ist. Leider erlaubt es Swagger nicht mehr als ein set von Beispieldaten zu deklarieren<sup>22</sup> weshalb manuelles Testen hin und wieder nötig war.

Beim Beheben von Programmfehlern wurden zusätzlich zwei Unit Tests geschrieben um zu verhindern, dass die Fehler im späteren Entwicklungsverlauf wieder auftreten.

---

<sup>22</sup>Die „Beispieldaten“ sind eigentlich gar nicht in Swagger vorgesehen. Allerdings erlauben viele Tools, die Swagger objekte verarbeiten die „Default“ Werte als Beispielanfrage abzusenden.

## 15.2 Frontend

Im Front-End wurde durchgängig auf die Funktionalität von DONAALDA getestet. REST HTTP Requests tests wurden meistens über die Konsole und/oder Postman ausgeführt. Weitere Unit-Tests waren durchs Testen in unterschiedlichen Browsern und auf unterschiedlichen Betriebssystemen mit unterschiedlichen Auflösungen.

Für die zwei letzten Meilensteine haben wir DONAALDA 6 externe Nutzer im Altersbereich von 20-65 Jahre mit unterschiedlichen Hintergrund über SmartHomes gezeigt. Das war insofern hilfreich, da diese entweder bestimmte Funktionalitäten verlangten oder welche nicht verstanden haben bzw. irritiert waren. Um einen gemeinsamen Nenner zu finden wurden die gemeinsame Beobachtungen der Nutzer zu weiteren Issues weitergeführt. Wichtig war sowohl die gesamte Seite zu testen als auch bestimmte Features separat, da bei einer Allgemeinbewertung ein Nutzer tendiert emotional und nicht sachlich zu bewerten.

Ein Beispiel für ein verwendetes Testprotokoll:

## Usability Testprotokolle

Protokoll für Usability Test					
Datum/Uhrzeit:	04.01.2017	Protokoll-Nr.:	06		
Prüfer/in:	Morteza Norozzi	Beobachter:	Oskar Schlösinger		
Testgegenstand/ Testelemente :	Filterfunktion				
Teststart:					
Testablauf:	-in Chrome: -nachdem Onboarding HttpResponse zurückkommt -Filtermaske einblenden -aus der Liste relevante Ergebnisse auswählen -gleiches Prozedere nach einer Seitenaktualisierung probieren -gleiches Prozedere nach einer zurück – vorwärts probieren				
Aufgetretene Fehler/ Probleme:	-irrelevante Suchergebnisse werden angezeigt	Mögliche Ursachen:	-Implementierung: es wurde eine falsche Methode benutzt		
Testergebnis:	<input type="radio"/> akzeptiert <input checked="" type="radio"/> nicht akzeptiert <input type="radio"/> Test abgebrochen	<input type="radio"/> ohne Einschränkungen <input type="radio"/> mit Einschränkungen			
Bemerkungen :					
Screenshot:	<b>Filtern nach Unterkategorien</b> <p> <input type="checkbox"/> Aufwachanreiz      <input type="checkbox"/> Aufzeichnung des Aktivitätsgrads      <input type="checkbox"/> Aufzeichnung von Vitaldaten      <input type="checkbox"/> Ausschaltung Elektrogeräte  <input type="checkbox"/> Automatisierung von Routinetätigkeiten      <input type="checkbox"/> Besuch      <input type="checkbox"/> Automatische Lichtsteuerung      <input type="checkbox"/> Rolladensysteme  <input type="checkbox"/> Kinderschutz      <input type="checkbox"/> Küchenassistent      <input type="checkbox"/> Energieverbrauch aufzeigen      <input type="checkbox"/> First-Aid Kit            Warnung vor Wasser und Feuer      <input type="checkbox"/> Schlafanreiz      <input type="checkbox"/> Intelligenter Wärmestrahl      <input type="checkbox"/> Schutz vor Einbruch         </p> <p style="text-align: right;"><b>NEU FILTERN...</b></p>				

### 15.3 Bugs

Auch wenn die Grundfunktionalität des Advisors gewährleistet ist, existieren noch einige kleinere Fehler. Diese behindern zwar die Funktionalität nicht, führen jedoch zu einigen Unstimmigkeiten. Genau verbleiben die folgenden Fehler:

- Wenn in der Szenariophase ein Szenario aus dem Warenkorb entfernt wird, das nicht das zuletzt hinzugefügte ist, sind die Angezeigten Preise und Produkttypen der verbleibenden Szenarien im Warenkorb inakkurat. Der Grund dafür ist ein Kommunikationsproblem zwischen der Backend und Frontend Gruppe. Das Backend schickt in der Szenarioauswahlphase Preise und Produkttypen für die gesamte SmartHome Konfiguration zurück. Das Frontend interpretiert die Daten jedoch als Szenariospezifisch. Das führt dazu, dass das Löschen eines Szenarios aus dem Warenkorb alle Produkttypen, die beim hinzufügen davon im Haus angezeigt wurden wieder entfernt werden, selbst wenn sie noch von anderen Szenarien verwendet werden.
- Wenn von der Szenarioauswahl zur Onboardingphase zurückgewechselt wird, wird der Szenariowarenkorb geleert. Zusätzlich wird dies nicht im User Interface widergespiegelt weshalb dies nicht nur den spezifizierten Anforderungen nicht entspricht sondern auch für den Nutzer verwirrend ist.

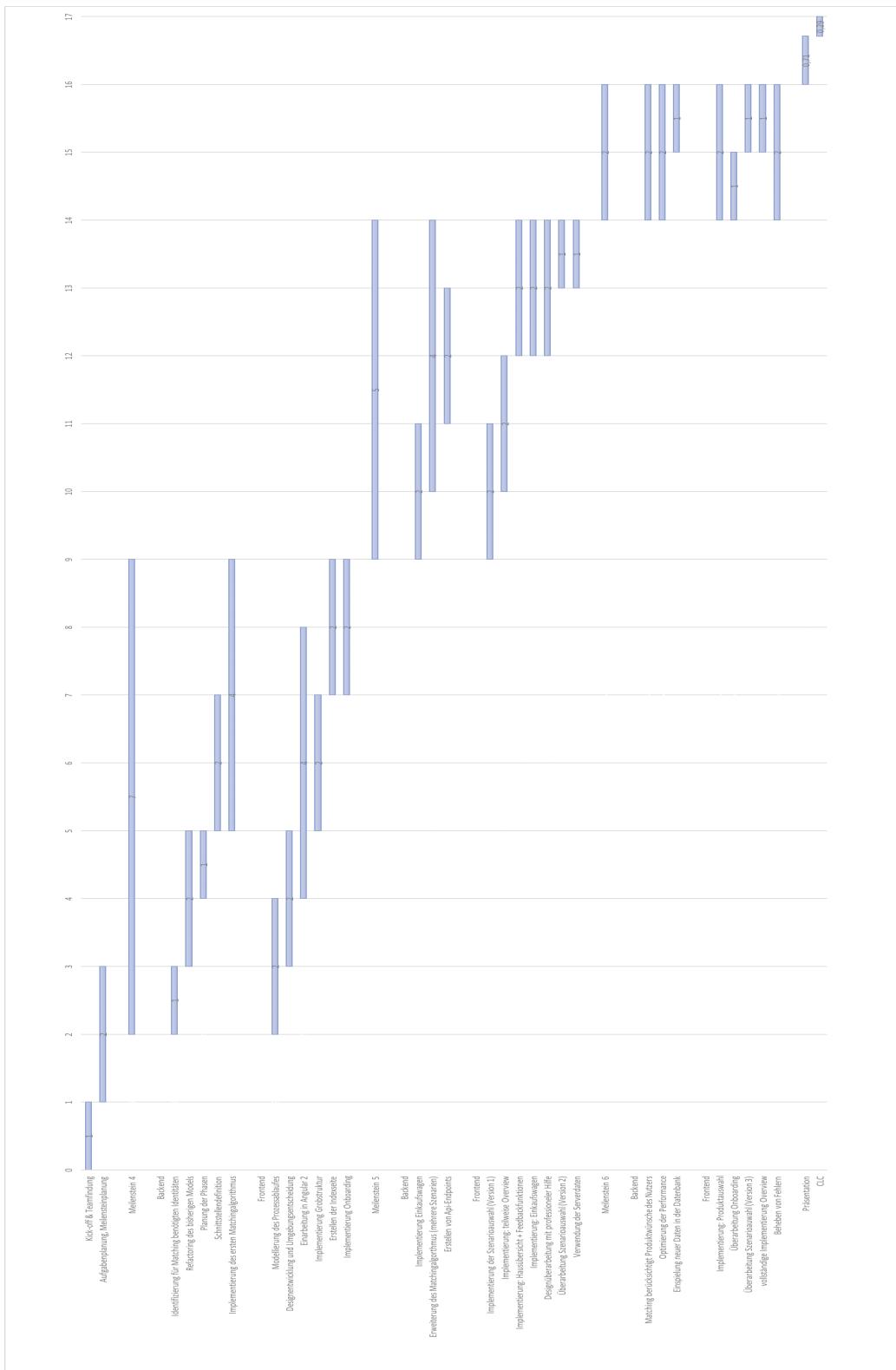
Ein weiterer möglicher Ansatzpunkt für zukünftige Verbesserungen ist das Layout der Seite. Aktuell ist dieses für Desktop Rechner mit einer Auflösung von  $1920 \times 1080$  Pixeln optimiert und verhält sich zum Teil merkwürdig auf Touchscreens und anderen Auflösungen. Ideal wäre es, wenn sich das Layout dynamisch an die Bildschirmgröße anpassen würde.

## **16 Ablaufplan**

Das Projekt erstreckte sich über das Wintersemester 2016/2017. Aufgrund der begrenzten Zeit und der fehlenden Expertise hat sich das Team gegen Scrum entscheiden. Der Aufwand, Anforderungen in Form von User Stories zu definieren und diese abzuarbeiten, gab der Zeitrahmen nicht her. Eine klassische Projektplanung mit Projektstrukturplan und festen Arbeitspaketen war aus Zeitgründen ebenfalls keine Option. Wir haben uns daher für eine Iterative Projektplanung in Anlehnung an die agile Projektplanung entschieden. Somit waren feste wöchentliche Treffen der Gesamtgruppe geplant. Alle zwei Wochen erfolgten feste Abstimmungstermine mit den Auftraggebern als Review und für die weitere Planung. Zwei Meilensteine sorgten für die Vorstellung des ersten und zweiten Prototypens im Abstand von vier Wochen. Zur Endpräsentation wurde dann der fertige Advisor dem Auftraggeber vorgestellt. Zudem konnte der Advisor Mitte Februar 2017 auf der jährlich stattfindenden Connected Living ConnFerence einem Fachpublikum erstmals vorgestellt werden.

Um die Herausforderungen und Ziele zu erreichen, hat sich das Team in ein Frontend- und ein Backendteam aufgeteilt. Die Teams haben sich mehrmals in der Woche getroffen. Schrittstellenspezifikationen und Fragen zur Gesamtzielsetzung wurden dann regelmäßig in dem wöchentlichen Gesamtmeeting diskutiert.

### **16.1 Zeitplan**



## **16.2 4. Meilenstein**

Der vierte Meilenstein war eine Einführungsphase für das Projekt, wo die ersten Ideensammlungen bezüglich des Frontenddesigns und Aufbau des Matchingprozesses gesammelt wurden. Die ersten Prototypen wurden stets mit den Auftraggebern ausdiskutiert und mit in die nächsten Schritte übernommen. Dadurch entstand auch detaillierte Planung der weiteren Implementierungsphasen wie z.B. Onboarding und Ablaufstruktur des Advisors. Eine Herausforderung für das Frontend-Team bereitete die Einarbeitung in die Angular2, finden der relevanten Dokumentation und einsetzbaren Patterns.

Abließend waren die gesetzten Ziele im Sinne des Erstellen vom ersten Prototypes des Advisors, sowie die Umsetzungsphasen des Matching-Algorithmus erfüllt.

## **16.3 5. Meilenstein**

Ziel des fünften Meilensteins war die Umsetzung des ersten Designprototyps für das Szenarioauswahlverfahren, Hausdarstellung, Produktauswahl- und Overview-Phasen im Frontend, sowie Implementierung vom Multiszenario Matching als Erweiterung von schon umgesetzten Matching-Algorithmus im Backend. Parallel erfolgte die Verbesserung des Onboarding-Designs und Testen der ersten Szenariendarstellungen anhand der ausgewählten Präferenzen in der Onboarding-Phase. Das Backend-Team arbeitete an der Umsetzung von Shoppingbasket Funktionalitäten, die später im Frontend mithilfe von TypeScript entstanden ist.

Für den erfolgreichen Abschluss dieses Meilensteins wurde dem Frontend-Team auch professionelle Hilfe seitens des Auftraggebers zur Verfügung gestellt. Basierend auf dem erhaltenen Feedback wurden auch das Design der einzelnen Phasen überarbeitet.

## **16.4 6. Meilenstein**

Im letzten Projektmeilenstein wurden grundlegende Designverbesserungen im Frontend durchgeführt. Das Haus wurde parallel interaktiver und farbiger für den User gemacht. Dabei mussten viele SVG-Grafiken an das Haus angepasst bzw. neu erstellt werden. Das Backend-Team beschäftigte sich mit dem Ersetzen von Produkten.

Abschließend wurde Feedback von den Auftraggebern in der Endpräsentation eingeholt und die letzten Änderungen vor der CLC-Konferenz, sowie Fehlerbehebungen und Bugfixes durchgeführt. Die noch offene Projektissues und bestehende Bugs, die leider aus zeitlichen Gründen nicht mehr zu verbessern waren, sind in „Bugs“ 15.3 beschrieben.

## 16.5 Projektabschluss

Nach einer internen Vorstellung wurde das Projekt auf der Connected Living ConnFerence 2017 am 15.-16.2.2017 vorgestellt und hatte das Interesse von mehreren Teilnehmer geweckt.

## 16.6 Ausblick

Der Advisor, der im Rahmen des PAS-Projekts nun entstanden ist, hat beim Fachpublikum auf der Connected Living ConnFerence 2017 einen äußerst positiven Eindruck hinterlassen und aufgezeigt, dass die Vision von CL, Smart Home Lösungen dem Kunden näher zu bringen, einen dringenden Bedarf darstellt.

Die Fragmentierung der Produkte auf dem Markt und die stets sich ändernden technischen Gegebenheiten führen beim Verbraucher zur Verunsicherung. Der Advisor hilft diese Hürde zu überspringen.

Der Advisors wird somit Bestandteil von Connected Living und dessen Angebots. Dazu wird das DAI-Labor die Weiterentwicklung des Advisor fachlich durchführen und an weitere Anforderungen von CL anpassen.

## 17 Anhang

### 17.1 Wichtige URLs

- **Frontend Repository** <https://gitlab.tu-berlin.de/sebastian.ahrndt/de.dailab.aal2016.tornados>
  - Web-App
  - Dokumentation
  - Protokolle
  - Präsentation
- **Backend Repository** <https://gitlab.tu-berlin.de/niautanor/aal2-backend>
- **Latest Deployment** <http://donaalda.baguette.management/>

### 17.2 Glossar

**Component / Komponente** Im Allgemeinen sind Components eine Kombination aus Template und Controller. Eine Angular Anwendung besteht hauptsächlich aus Komponenten. Die Definition einer Komponente erfolgt dabei ähnlich zu den anderen Bestandteilen einer AngularJS-App. Durch den Aufruf der Funktion `.component` wird eine neue Komponente erstellt und kann verwendet werden. Sie erwartet zwei Argumente:

- Name der Komponente (String) - eindeutige Zeichenkette
- Konfiguration (Object) - verknüpft Template mit Controller

**Service / Dienst** Services enthalten die Geschäftslogik und binden externe Ressourcen – etwa REST-Webservices – ein. Services werden als Singleton instanziert.

Services können selbst programmiert werden oder von Drittanbietern übernommen werden. Das AngularJS-Framework stellt aber bereits zahlreiche Services (erkennbar am \$-Präfix) zur Verfügung. Dazu zählen beispielsweise `$http` und `$resource`, die zum Durchführen von AJAX-Anfragen dienen. Beide greifen intern auf das XMLHttpRequest-Objekt zu und unterscheiden sich im Abstraktionsgrad. Während `$http` beliebige HTTP-Anfragen durchführen kann, ist `$resource` auf REST-Services spezialisiert.

**Onboarding / Grundfragen** Die ersten Fragen, die dem Nutzer gestellt werden. Diese helfen ihm, Präferenzen zu ermitteln.

**Category / Kategorie** es sind vier Kategorien definiert, denen Szenarien zugewiesen werden können. *Energie, Komfort, Sicherheit und Gesundheit.*

Jedes Szenario hat eine Bewertung von 1 bis 10 für jede der vier Kategorien, die unabhängig ausgewählt werden kan. Dementsprechend beeinflusst die Bewertung "10" z.B bei der Kategorie "Gesundheit" die anderen nicht. Die Bewertungen werden während des Onboardings abgegeben und in der Szenarioauswahl berücksichtigt.

**Subcategory / Unterkategorie** Jede Kategorie ist in Unterkategorien unterteilt.

Bsp.: *Feuer* wäre eine Unterkategorie von *Sicherheit*. Die Nutzung von Unterkategorien ist soweit nur im Backend implementiert mit dem Hintergedanken, dem Nutzer eine effizientere Szenarioauswahl anzubieten.

**Scenario / Szenario** Ein Szenario entspricht einer genauen Anwendung der Funktionalität in einem SmartHome bzw. um einen bestimmten Zweck zu erfüllen.

Bsp.: „Warnung vor Feuer“ würde einen Szenario beschreiben, „Warnung“ dagegen nicht, da es zu generisch wäre.

Ein Szenario hat einen Namen, eine Beschreibung, Bewertungen für jede der vier Kategorien (siehe oben) und eine Menge von Meta-Devices, die zur Implementierung der Funktionalität eines Szenarios nötig sind

**Feature / Merkmal** Ein Merkmal ist eine genaue Funktionalität eines Produkts oder Meta-Device, die fürs Zusammenpassen (Matching) der beiden Elemente benutzt wird. Ein Produkt lässt sich als Implementierung eines Meta-Devices betrachten, sofern es den gleichen Typ hat (Broker oder Endpoint) und die Merkmale des Produktes sich als eine Obermenge wie ein Meta-Device darstellen lassen. Gerätetypen (z.B. "Ist eine Waschmaschine") sind keine Merkmale.

Da Einzelmerkmale unendlich vielfältig sein können, hilft es, wenn wir uns an Merkmalskategorien halten und die Beschreibung eines einzelnen Merkmals durch die Zusammenführung mehrerer Merkmale aus folgenden fünf Merkmalskategorien beschreiben: "operates", "notifies", "records", "senses" und "controls".

**Meta-Device** Ein Meta-Device repräsentiert ein Gerät mit bestimmter Menge von Features (siehe oben). Diese dürfen unterschiedlich spezifisch sein, sollten aber an den Benennungsrichtlinien orientieren.

Meta-Devices können nur Merkmale spezifizieren, die in den implementierten Produkten enthalten sind. "Negativmerkmale" wie "leuchtet nicht grün" sind nicht möglich.

**Produkt** Ein Produkt ist ein physisches Gerät, das eine bestimmte Menge von Merkmalen implementiert. Dementsprechend implementiert das selbe Produkt auch jenes Meta-Device, der eine Untermenge der im Produkt definierten Merkmale hat.

Produkte können spezifizieren, dass deren Installation oder Montage eine Fachkraft benötigt wie z.B. eine Verkabelung durch die Wand. Entsprechend dieses Vermerks wird ein solches Produkt bei Angabe des Nutzers in den Grundfragen in Betracht gezogen und das Produkt erscheint oder erscheint nicht in der Endgültigen Auswahl.

**Broker und Endpoint** Meta-Devices und Produkte sind klassifiziert als entweder Broker oder Endpoint.

**Endpoints** Sammeln Informationen (z.B. Lichtsensor) oder verarbeitet diese (z.B. Rolladenaktor).

**Broker** Ermöglichen die Kommunikation zwischen zwei Produkten (z.B. SmartHome Bedienungsgerät wie IOLITE).

Ein Broker kann außerdem wie ein Endpoint agieren, falls es Information anbieten kann oder die Umgebung verändern kann. (Bsp. ein SmartHome Bedienungsgerät kann die Wettervorhersage über das Internet empfangen und direkt ohne Temperatursensor den Rolladenaktuator steuern)

**Protokoll** Produkte kommunizieren miteinander über Protokolle. Protokolle können von Geräten im Leader oder im Follower Modus implementiert werden. Wir sagen, dass ein Gerät ein Protokoll im Follower Modus unterstützt, wenn es alle Informationen, die zum auslesen und kontrollieren aller seiner Features notwendig sind über dieses Protokoll von einem Gerät, das dieses Protokoll im Leader Modus unterstützt senden bzw. empfangen kann. Das bedeutet, dass diverse Standards wie Bluetooth oder UPnP, auch wenn sie oft so genannt werden bei uns keine Protokolle sind.

Protokolle wären stattdessen z.B. "UPnP-Samsung-Smart-TV". Implementierung dieses Protokoll in einer SmartHome Basisstation würde bedeuten, dass diese Basisstation alle Features des entsprechenden Samsung Smart-TV kontrollieren kann.

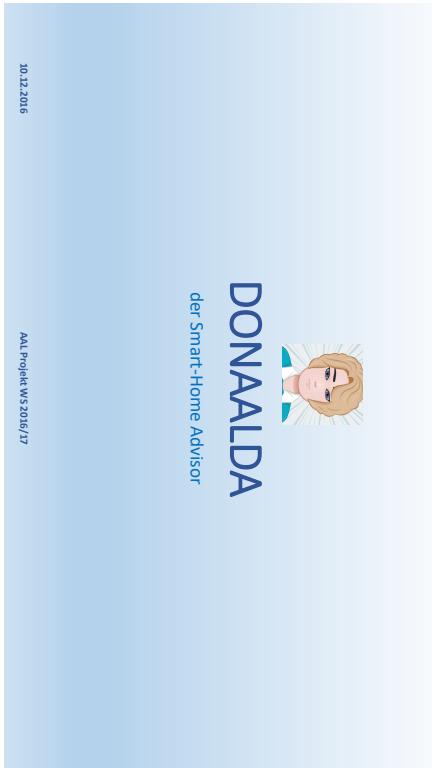
**Produkttyp** Produkte haben Produkttypen um den Nutzer zu erlauben, Szenarien danach zu filtern.

Bsp.: Ein Nutzer möchte nach dem Produkttyp "Waschmaschine" filtern, um zu schauen, welche Extra-Funktionalität eine Smart-Waschmaschine von Vorteil hat. Da Szenarien nicht direkt auf Produkten festgelegt werden, erscheint ein Szenario nach dem Filtern, wenn dieses mindestens ein Produkt enthält, welches mindestens durch ein Produkt mit implementiert wird, der diesen Produkttyp hat.

## **17.3 Präsentationsfolien**

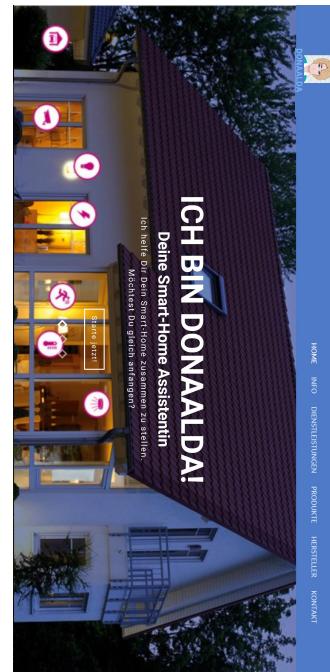
### **17.3.1 4. Meilenstein**

## Index Page



10.12.2016

AAL-Projekt WS 2016/17



2

## Auswahl der Produktpräferenzen

Grundfragen für Dein Smart Home



1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

## Szenariendarstellung

<Haus Component>	<Category Component>	<Preference Component>	<Renovation Component>
<Scenario Filter Component>	<Filter-Subkategorien>		
<Scenario Selected Component>			
<Scenario List Component>			

6

Button für das  
Anzeigen der  
weiteren Szenarien

Onboarding Component-Aufbau

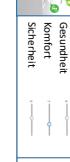
<b>&lt;Orientierungskomponenten&gt;</b>
[Orientierungskomponenten Teil 1] ... 1. Orientierung ..... 2. Orientierung ..... 3. Orientierung ..... 4. Orientierung
<b>&lt;Orientierungskomponenten&gt;</b>
[Orientierungskomponenten Teil 2] ... 1. Orientierung ..... 2. Orientierung ..... 3. Orientierung ..... 4. Orientierung
<b>&lt;Orientierungskomponenten&gt;</b>
[Orientierungskomponenten Teil 3] ... 1. Orientierung ..... 2. Orientierung ..... 3. Orientierung ..... 4. Orientierung
<b>&lt;Orientierungskomponenten&gt;</b>
[Orientierungskomponenten Teil 4] ... 1. Orientierung ..... 2. Orientierung ..... 3. Orientierung ..... 4. Orientierung

5

## Szenarienauswahl (Beispiel)

The screenshot shows the EnergyGuru app's scenario selection screen. On the left, there are three cards with images and labels: 'Lichtsteuerung' (light control) with a smartphone displaying a light switch icon; 'Heizungssteuerung' (heating control) with a hand adjusting a digital thermostat; and 'Fenstersteuerung' (window control) with a person looking out a window. A large central button labeled 'Mein' (My) has an arrow pointing to it from the bottom left. To the right, a box contains the text 'Szenarien werden vom User ausgewählt' (Scenarios are selected by the user). Above this box is a section titled 'Szenarienfilter' (Scenario filter) with a grid of icons representing different filters: Energie (Energy), Gesundheit (Health), Komfort (Comfort), Scherheit (Safety), Preis (Price), Erweiterbarkeit (Extensibility), Professionalität (Professionalism), Do-it-yourself (DIY), and Renovierung (Renovation). Each filter has a slider or checkbox next to it, with some being checked.

88

Szenarienfilter	
	Kategorie Unterstützung
	Kategorie Netzwerkgewinnung
	Kategorie Fernsteuerung
	Kategorie Sicherheit
	Präferenzen Energyeffizienz Energieunabhängigkeit Professional Do-it-yourself Preis
	Renovierung Energyeffizienz Energieunabhängigkeit Professional Do-it-yourself Preis

7

## Szenarienauswahl (Beispiel)

**Szenarienfilter**

Kategorie	Präferenzen	Renovierung
Energie	Energieeffizienz	Professional
Gesundheit	Erweiterbarkeit	Do-it-yourself
Komfort	Preis	
Sicherheit		

**Umtreuung**

**Heizungssteuerung**

**Fenstersteuerung**

**Sicherheitskontrolle**

**Mehr**

10

## Szenariodetails als Modalfenster

**<Titel>**

**<Subtitel>**

**Präferenzen**

Kategorie	Präferenzen	Renovierung
Energie	Energieeffizienz	Professional
Gesundheit	Erweiterbarkeit	Do-it-yourself
Komfort	Preis	
Sicherheit		

**<Beschreibung?>**

**<Gerätytypen>**

**Schließen**

**In den Warenkorb**

12

## Szenarienauswahl (Beispiel)

**Szenarienfilter**

**Umtreuung**

**Heizungssteuerung**

**Fenstersteuerung**

**Sicherheitskontrolle**

**Mehr**

9

## Szenarienauswahl (Beispiel)

**Szenarienfilter**

**Umtreuung**

**Heizungssteuerung**

**Fenstersteuerung**

**Sicherheitskontrolle**

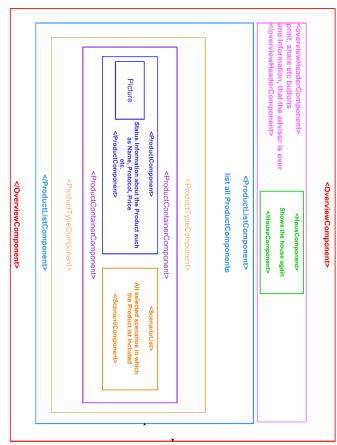
**Air-Conditioning**

**Mehr**

Wertete, zum Auswahl  
passende Szenarien werden  
angezeigt  
(immer 3x2 Darstellung)

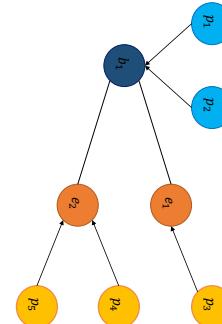
11

## Product-Overview (Component-Aufbau)



## Szenario – MetaDevice Alternativen

- Konstellation aller möglichen Produkte zu den MetaDevices
- Metabroker kann durch zwei Alternativen implementiert werden ( $p_1$  and  $p_2$ )
- MetaEndpoint 1 ( $e_1$ ) kann nur von Produkt 3 implementiert werden ( $p_3$ )
- MetaEndpoint 2 ( $e_2$ ) kann von Produkt 4 ( $p_4$ ) und Produkt 5 ( $p_5$ ) implementiert werden



14

## Szenariodetails als Modalfenster (Beispiel)

Lichtsteuerung	
Stromverbrauch immer im Blick	
<b>Kategorie</b>	<b>Präferenz</b>
Energie	Energieeffizienz
Gesundheit	EcoVirtuellerkeit
Kontakt	Preis
Sicherheit	Professional Do-it-yourself

**Seufz berichtet, hell aufgetroffen; Lebhafte nach Wunsch.**  
Überall im Haus die passende Beleuchtung - je nach Tageszeit, Ablenkungen und Vorlieben. Entdecken Sie Ihr systembasierte Lösungen von Philips für Ihr Smart Home. Wir fühlen Sie sich am wohlsten. Wurden Sie Ihr Wohnzimmer so in klassisch wie helles Licht tauschen oder bevorzugen Sie einzelne Leuchten eine warmen Farbe? Oder Sie eine eingesparte Arbeitsgrafe zum lesen oder eine gezielte Sicherheitsbeleuchtung im nächsten Hauflur schaffen wollen. Mit Philips Geräten gestalten Sie ganz einfach ein passendes Lichtsystem.

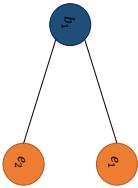
**Lösungen für Ihre Smart Home**

**Schließen** **In den Warenkorb**

13

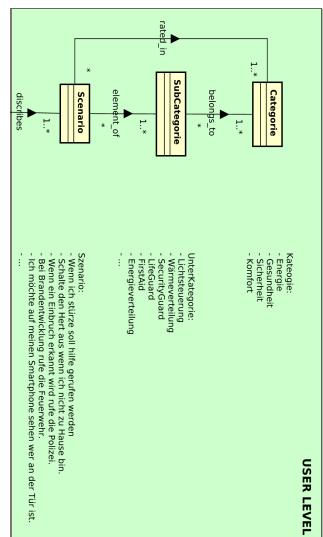
## Szenario – MetaDevice Konstellation

- Beispielhaftes Szenario
  - Ein MetaBroker ( $b_1$ )
  - Zwei MetaEndpoints ( $e_1$  and  $e_2$ )



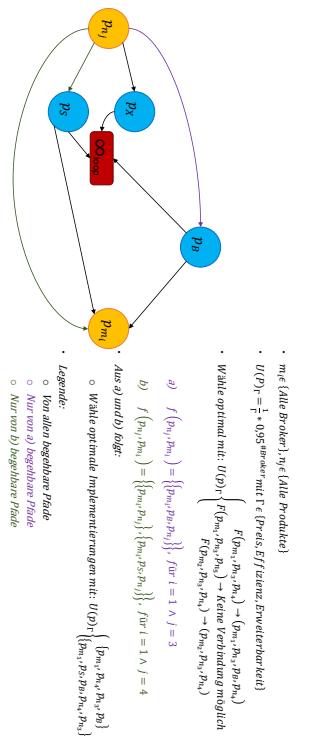
15

## [Back-Up] Kurz zur Definition



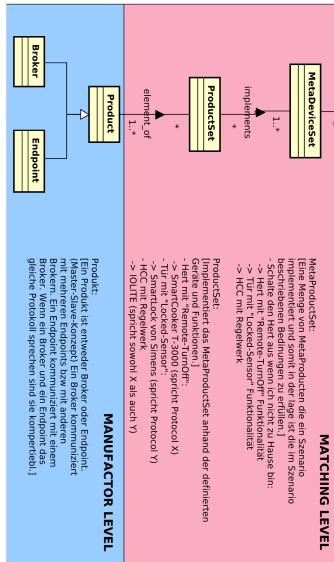
18

## Matching – optimale Lösung



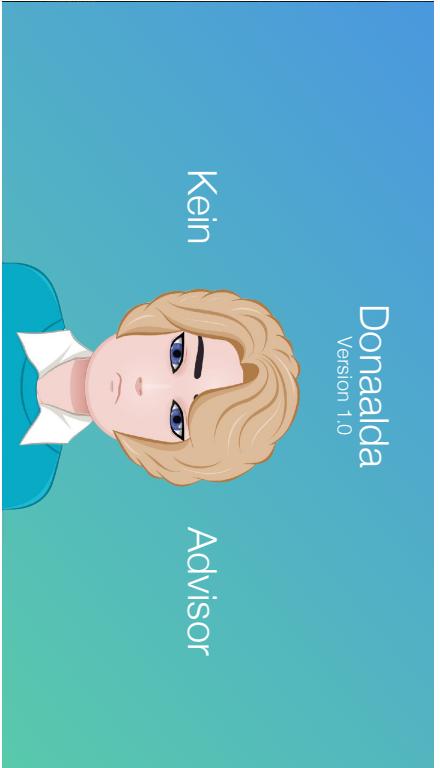
17

## [Back-Up] MetaDevice & Matching Level

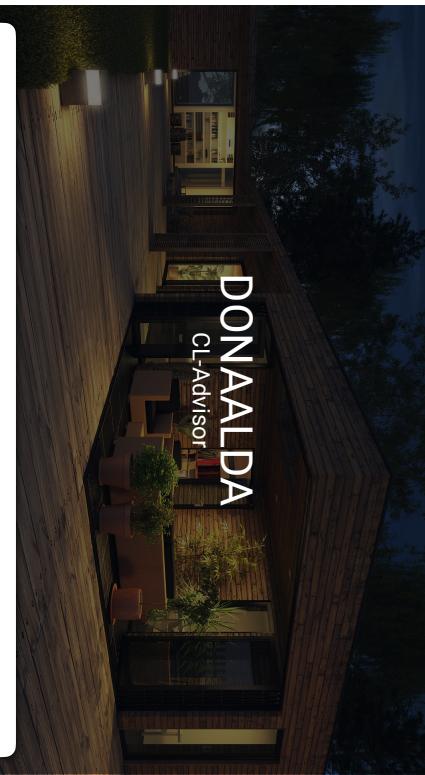


19

### **17.3.2 5. Meilenstein**



Donaalda  
Version 1.0

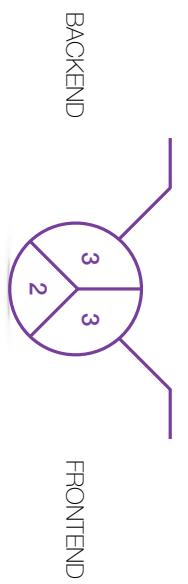


## TRANSFORMATION



- ★ Ein richtiger Advisor (Konfigurator)
- ★ Bedeutung der Szenarien überdacht
- ★ Produktzugehörigkeit überdacht
- ★ Flexibel, anpassungsfähig und dynamisch
- ★ Smart Home Erlebniswelten
- ★ Usability for Dummies
- ★ Advanced Matching
- ★ Single Page Application
- ★ Frontend / Backend Trennung
- ★ REST-Framework

## HERAUSFORDERUNGEN



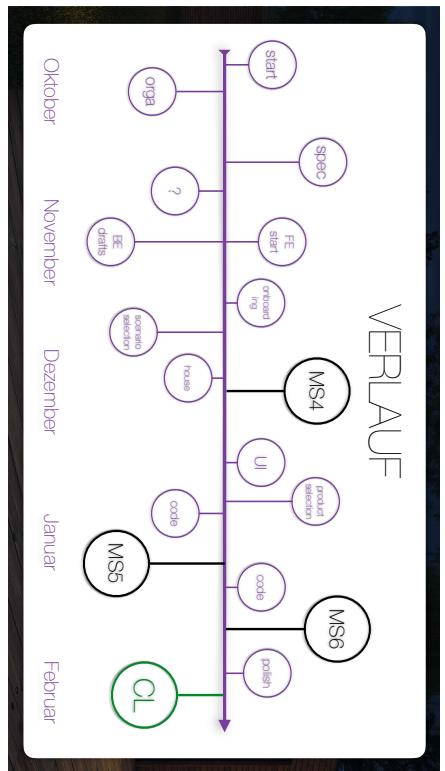
## ORGÄ

- ★ Aufteilung in Frontend & Backend
- ★ Ein wöchentliches Meeting mit der ganzen Gruppe
- ★ Mehrfache wöchentliche Meetings in den Gruppen
- ★ Spec / Kein Scrum / Aber Agil / Prototyping
- ★ Tools

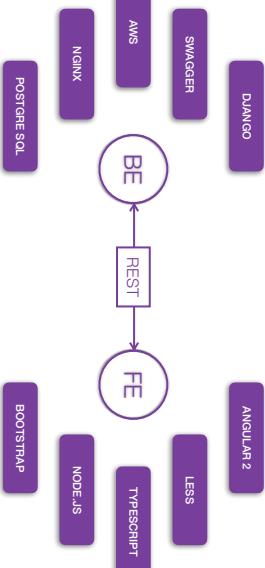
*„Der Connected Living Advisor führt den Benutzer in vier einfach Schritten zu seinem Smart Home, indem er gewünschte Funktionalitäten mit kompatiblen Produkten verbindet und dem Benutzer anbietet.“*

## HERAUSFORDERUNGEN

- ★ Multi Scenario Matching
  - ★ Scenario merging
  - ★ Komplexität
  - ★ Laufzeit
  - ★ Database requests
  - ★ Static Swagger
  - ★ Fixture dump database
- 



## TECHNOLOGIE



MATCHING  
Marvin



## SZENARIO-AUFBAU

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

## SZENARIO-AUFBAU

Waschen  
Abstraktes Gerät

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

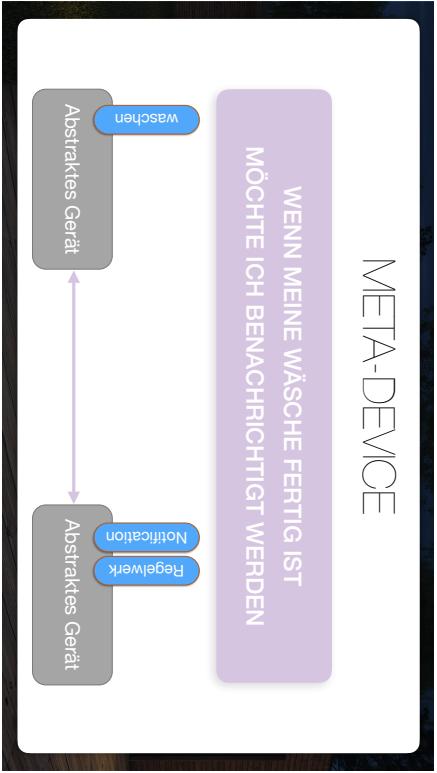
Waschen  
Abstraktes Gerät

Notifikation  
Abstraktes Gerät

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

## META-DEVICE

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

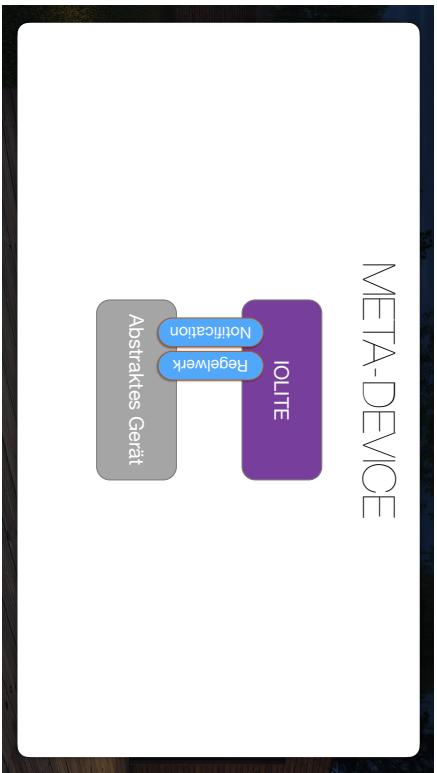


## IMPLEMENTIERUNG

Meta-Device

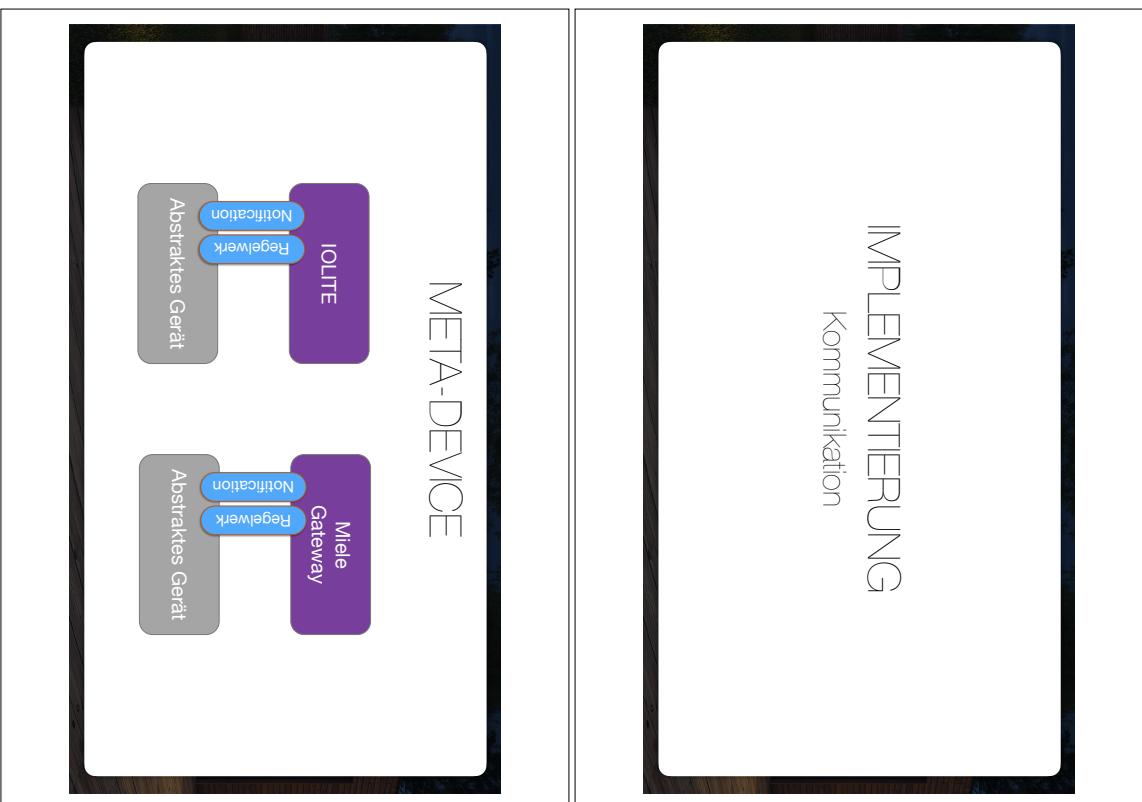


## META-DEVICE



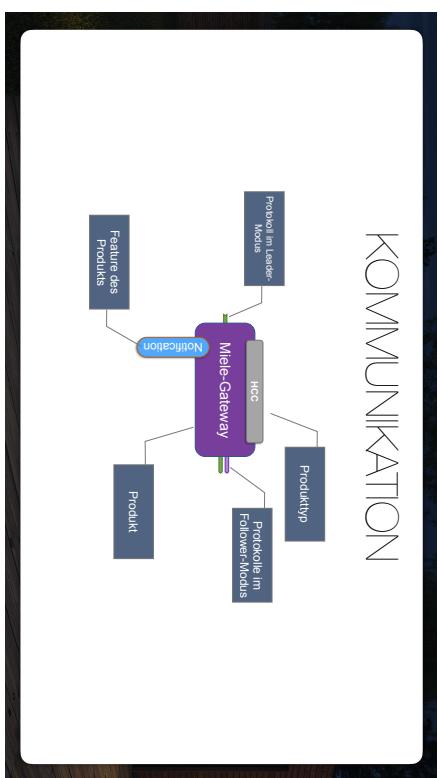
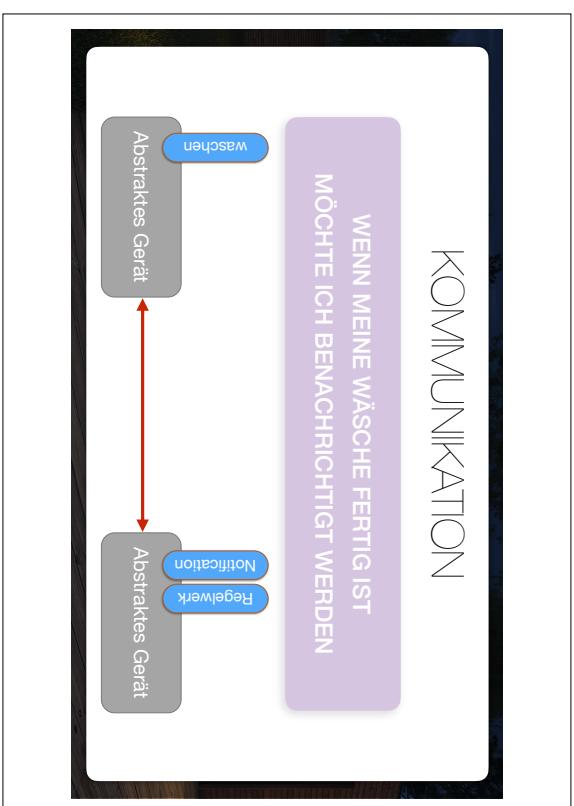
## IMPLEMENTIERUNG

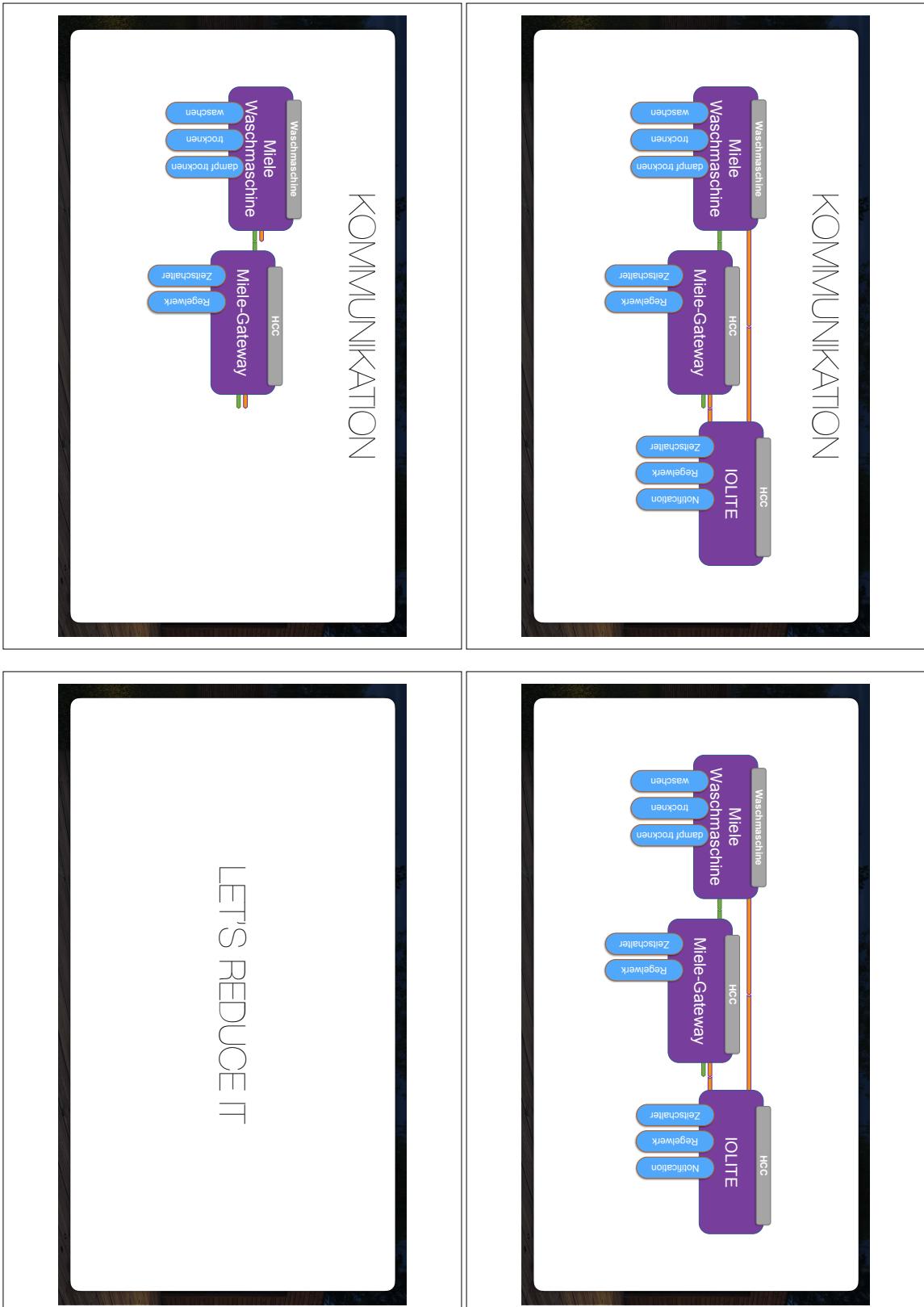
### Kommunikation

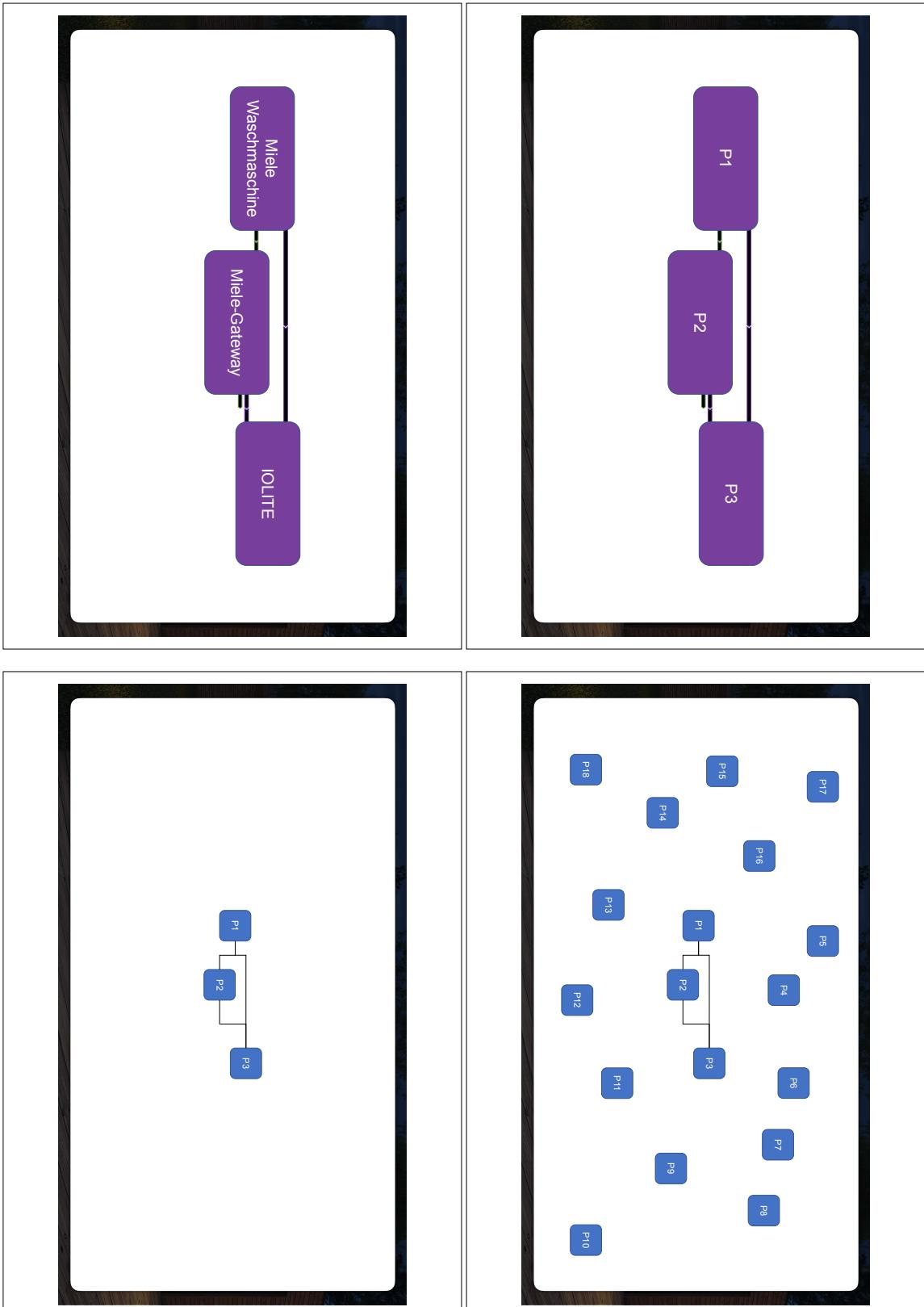


## KOMMUNIKATION

### KOMMUNIKATION

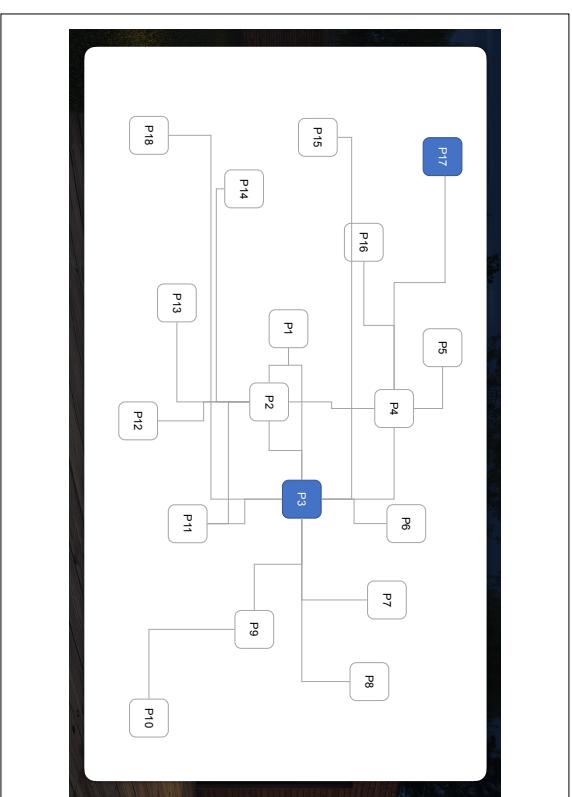
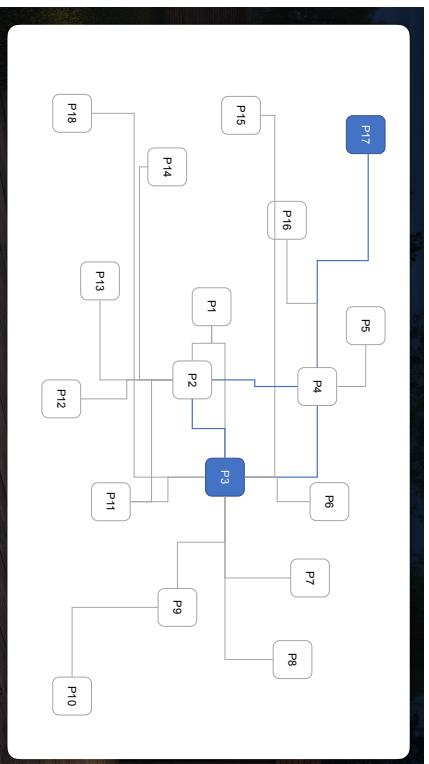
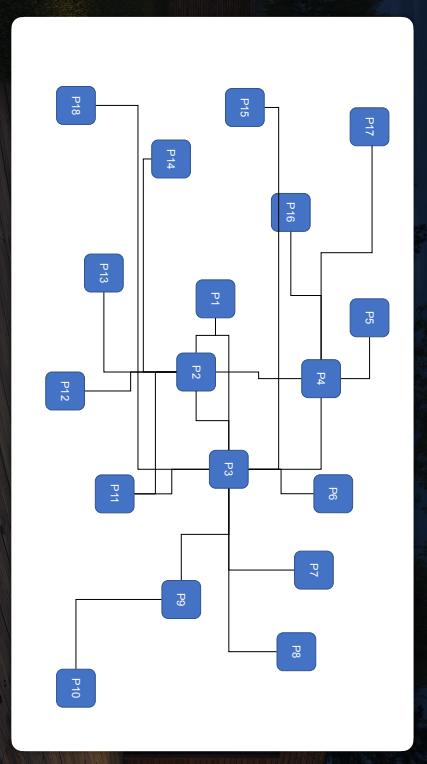


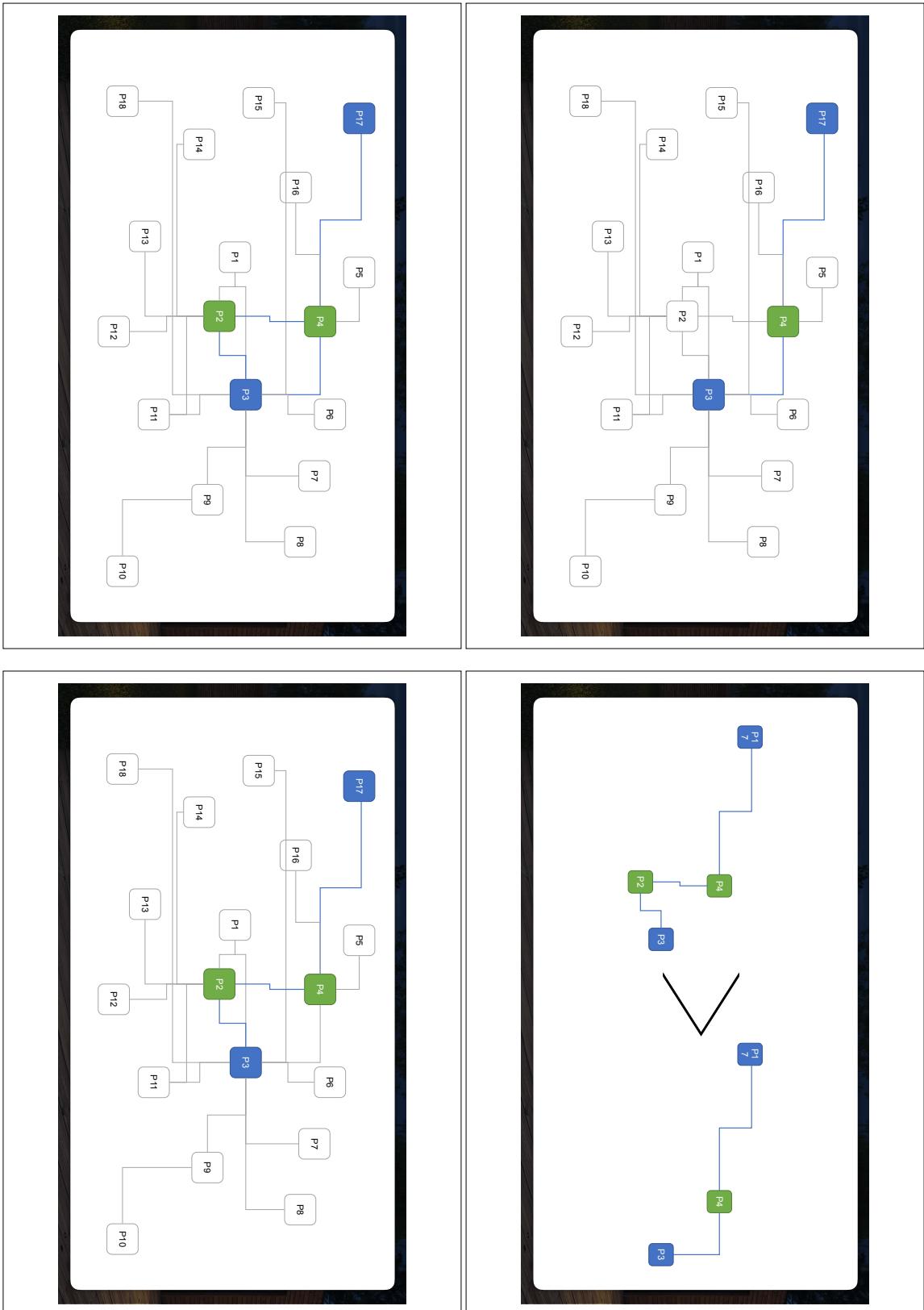




## REKURSIVE TIEFENSUCHE

Finde alle Wege wie ein Produkt mit einem anderen reden kann







## PRODUKTMATCHING

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

P17  
P4  
P3

P1  
P3

## PRODUKTMATCHING

P17  
P4  
P3  
P1  
P3

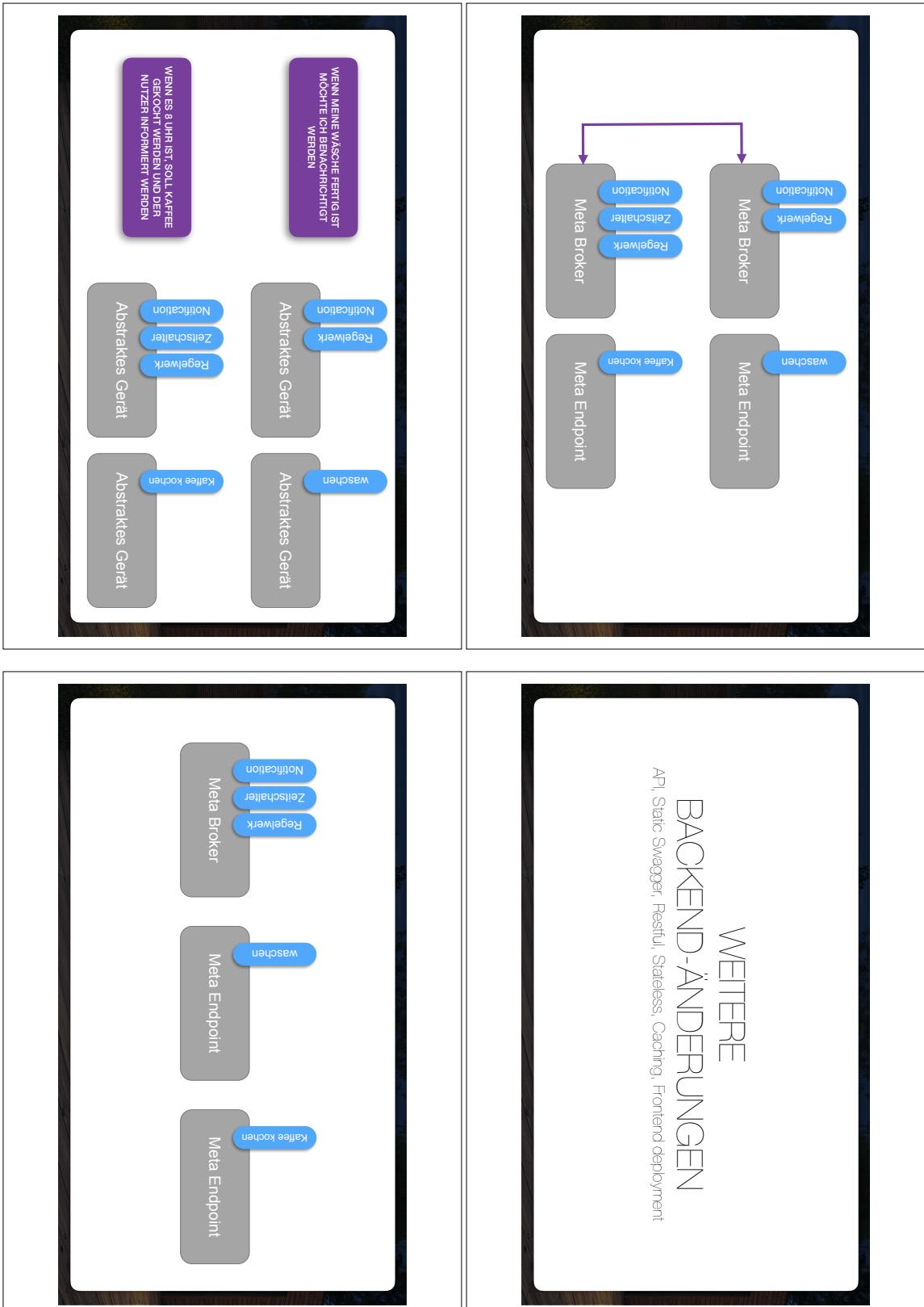
P1  
P3  
=

Meile  
Waschmaschine

IOLITE

## SZENARIO MERGING

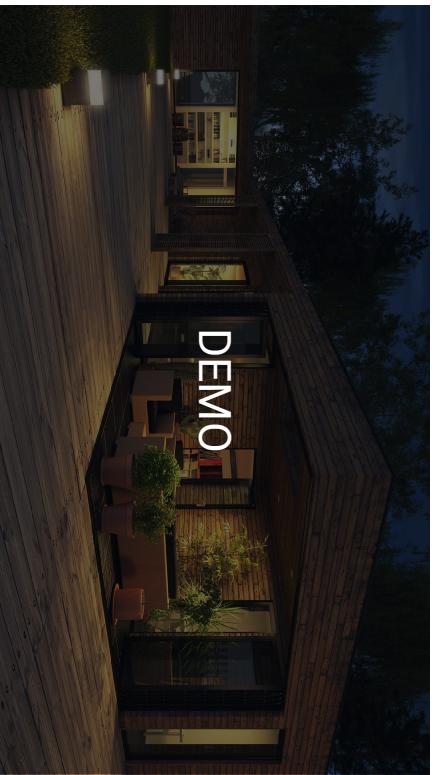
Merging von zwei Szenarien



## BACKEND-ÄNDERUNGEN

Restful – Stateless – Caching

- ★ Django SessionFramework wird nicht mehr verwendet
- ★ Jede Anfrage sendet alle benötigten Informationen zur Verarbeitung
- ★ Caching nun über method decorator anstatt von expliziten Aufrufen
- ★ Erhebliche Performance Verbesserungen
- ★ Gleiche Anfragen von unterschiedlichen Sessions greifen nun auf den selben Cache zu



## BACKEND-ÄNDERUNGEN

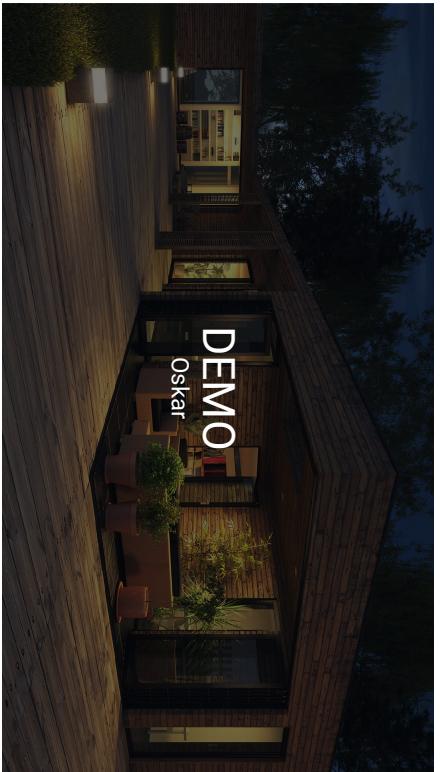
Frontend deployment

- ★ Neues git repository auf dem web server für das Frontend-Team
- ★ Post-receive hook als deployment script
  - ★ Kopiert gepushten git work tree in ein separates Verzeichnis
  - ★ Instaliert neu benötigte node modules
  - ★ Kompiliert die typescript Dateien
- ★ Schlichtberechnetes ProduktSet mit den zugehörigen Szenarios
- ★ Swagger UI ist nun statisch
- ★ Beispiele korrekt im UI von Endpoints mit verschiedenen Serializern



## VIER SCHritte

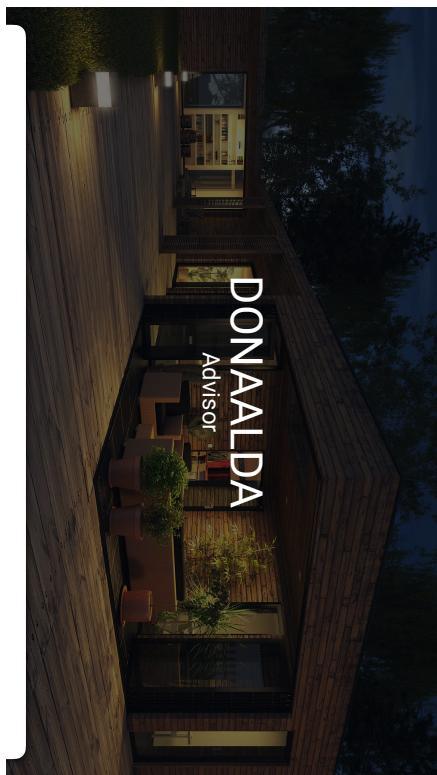
- 1 Grundfragen  
Bei Start des Advisors werden drei Grundfragen gestellt
- 2 Szenarioauswahl  
Anhand der Antworten aus Schritt 1 werden passende Szenarien gezeigt
- 3 Produktauswahl  
Die passenden Produkte zu den Szenarien inkl. Austauschen / Präferieren
- 4 Den Smart Home  
Die Übersicht aller passenden Produkte



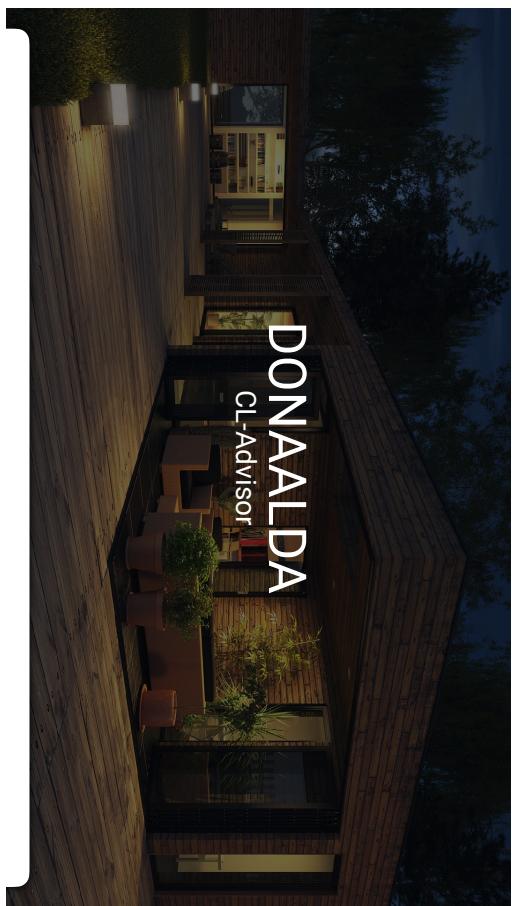
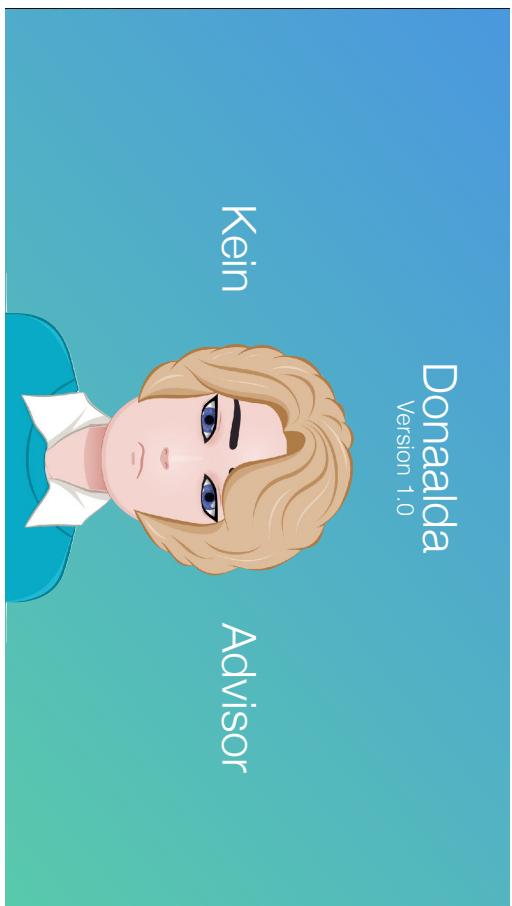
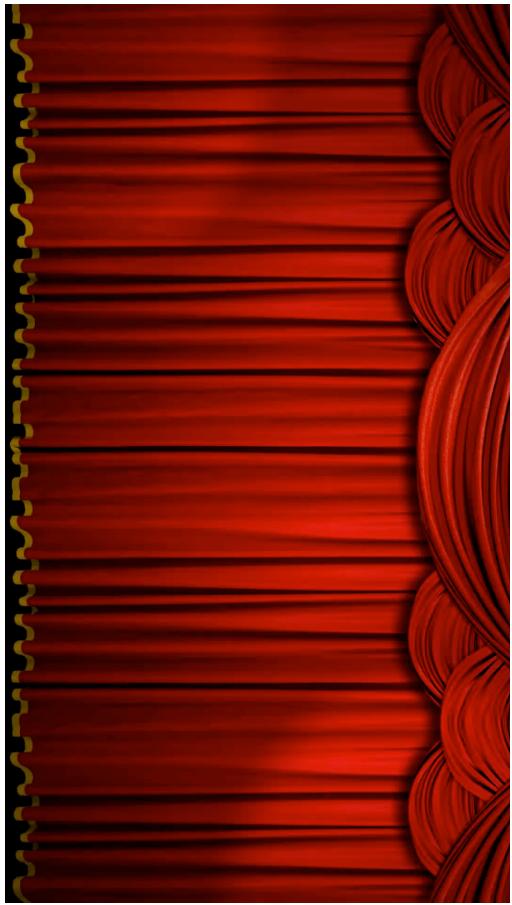
DONAALDA  
Advisor

CONNECTED  
LIVING  
ConnFerence 2017

We will be ready for



### **17.3.3 6. Meilenstein - Interne Präsentation**



## ORGÄ

„Der Connected Living Advisor führt den Benutzer in vier einfachen Schritten zu seinem Smart Home, indem er gewünschte Funktionalitäten mit kompatiblen Produkten verbindet und dem Benutzer anbietet.“

- ★ Aufteilung in Frontend & Backend
- ★ Ein wöchentliches Meeting mit der ganzen Gruppe
- ★ Mehrfache wöchentliche Meetings in den Gruppen
- ★ Spec / Kein echtes Scrum / Aber Agil / Prototyping
- ★ Tools (Git, Slack)

## TRANSFORMATION



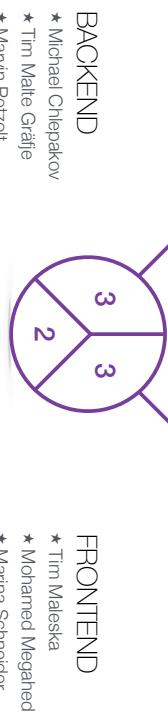
- ★ Ein richtiger Advisor (Konfigurator)
- ★ Bedeutung der Szenarien überdacht
- ★ Produktzugehörigkeit überdacht
- ★ Flexibel und erweiterbar
- ★ Smart Home Erlebniswelten
- ★ REST-Framework
- ★ Usability für Unerfahrene
- ★ Erweitertes Matching
- ★ Single Page Application
- ★ Frontend / Backend Trennung

## WAS MACHT DER ADVISOR ?

## HERAUSFORDERUNGEN

- ★ Multi Scenario Matching
- ★ Scenario merging
- ★ Komplexität
- ★ Laufzeit
- ★ Database requests
- ★ Product Selection
- ★ Fixture dump database
- ★ Angular 2 + Type Script 2
- ★ Später Final Release
- ★ Viele Veränderungen
- ★ Stetige Lernkurve
- ★ Design
- ★ Multiple Framework Issues
- ★ Ressourcen + Visualisierung

## HERAUSFORDERUNGEN



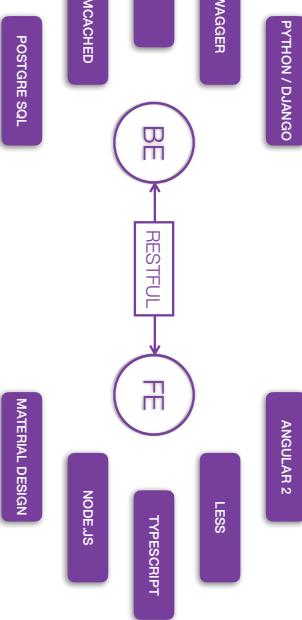
### FRONTEND

- ★ Michael Chlepkov
- ★ Tim Malte Gräfe
- ★ Marvin Petzolt

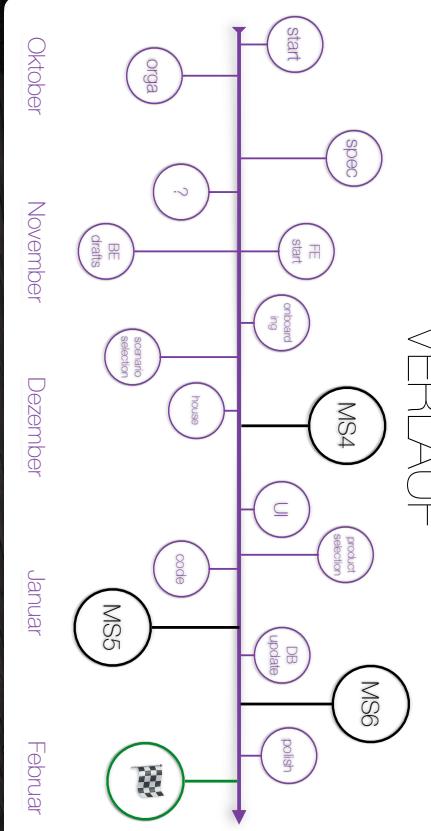
### FLEX

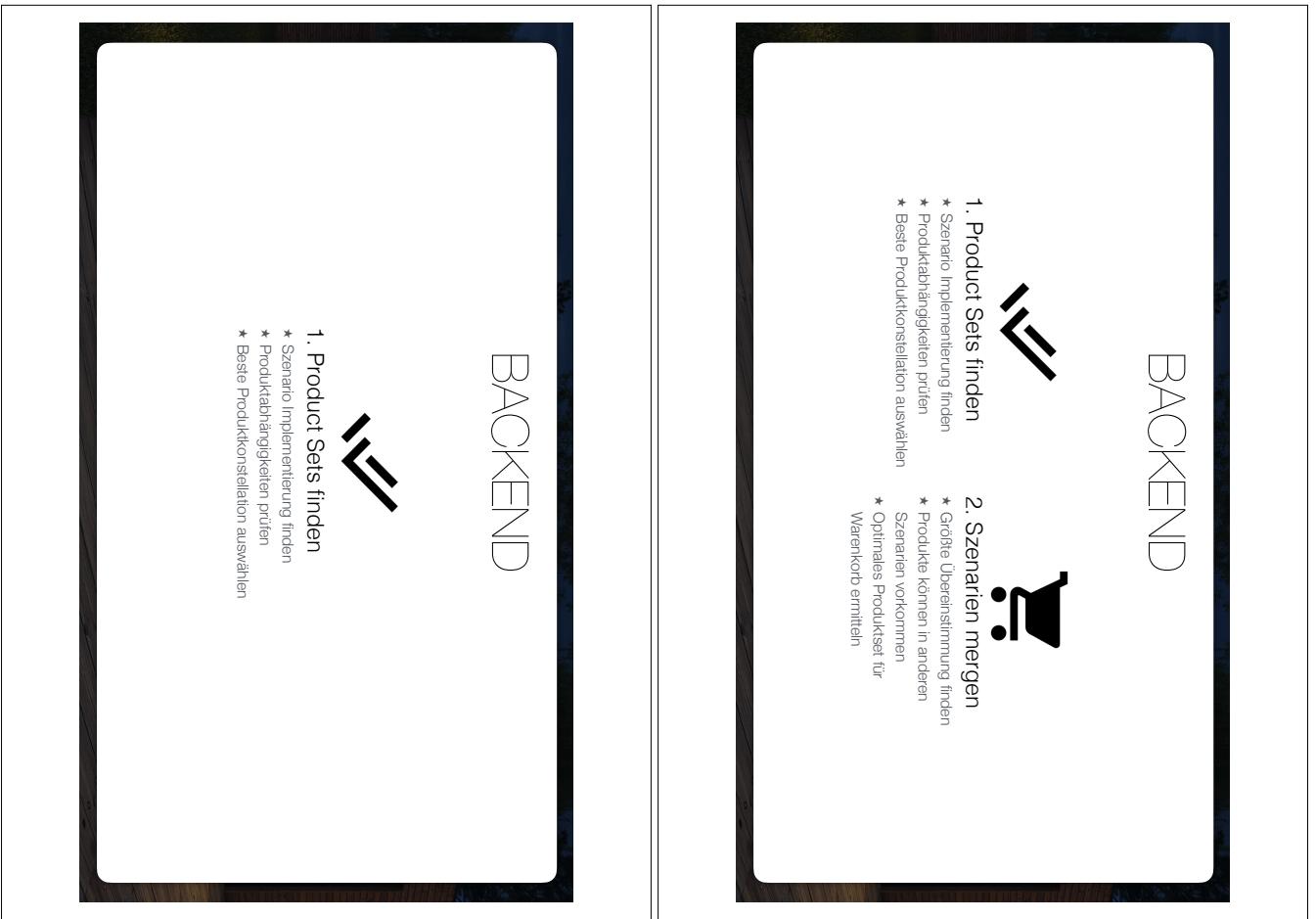
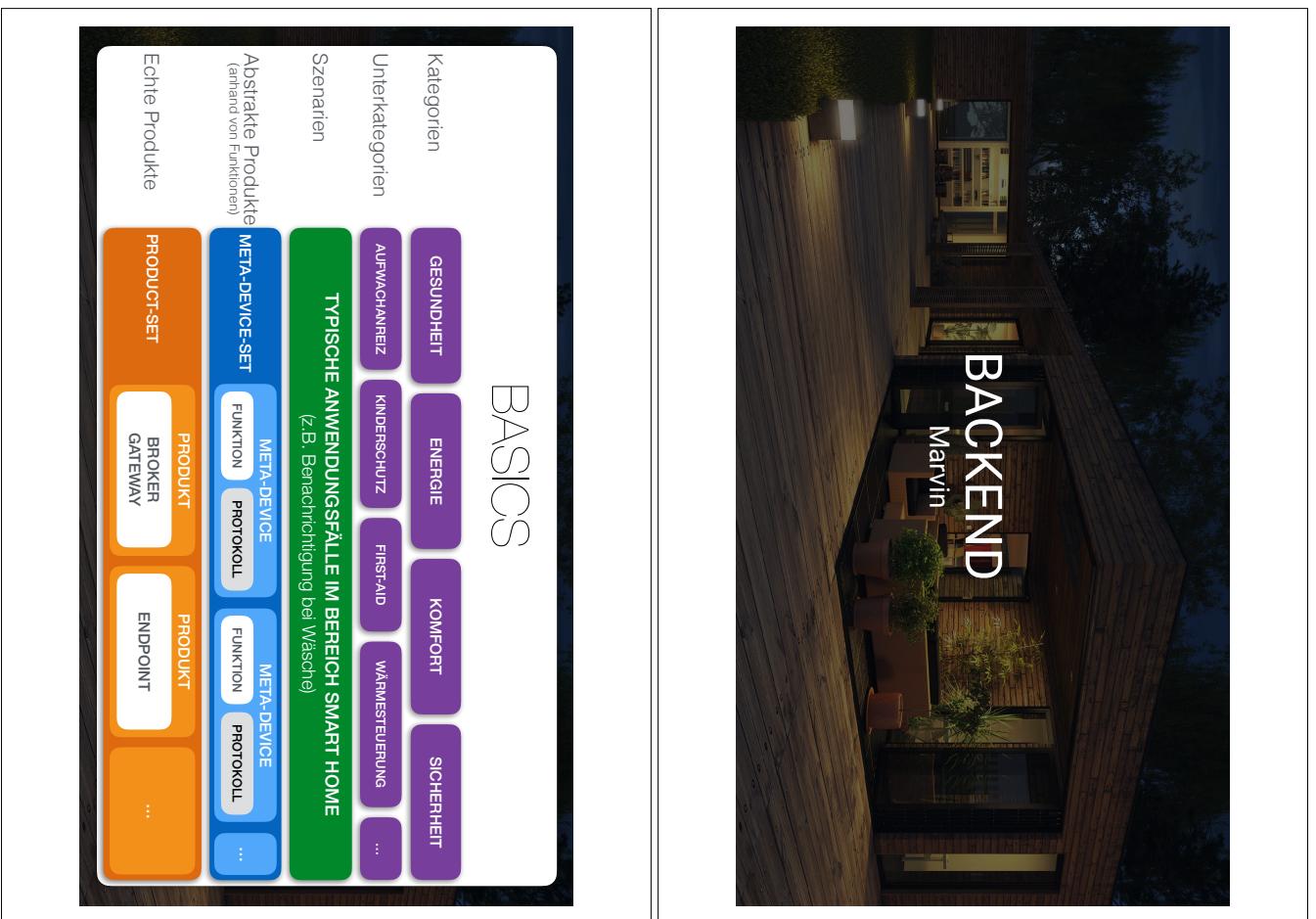
- ★ Oskar Schrössinger
- ★ Timo Runk

## TECHNOLOGIE



## VERLAUF





## SZENARIENAUFBAU

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

Abstraktes Gerät

Waschen

## BACKEND



WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN



### 1. Product Sets finden

- \* Szenario Implementierung finden
- \* Produktabhängigkeiten prüfen
- \* Beste Produktkonstellation auswählen

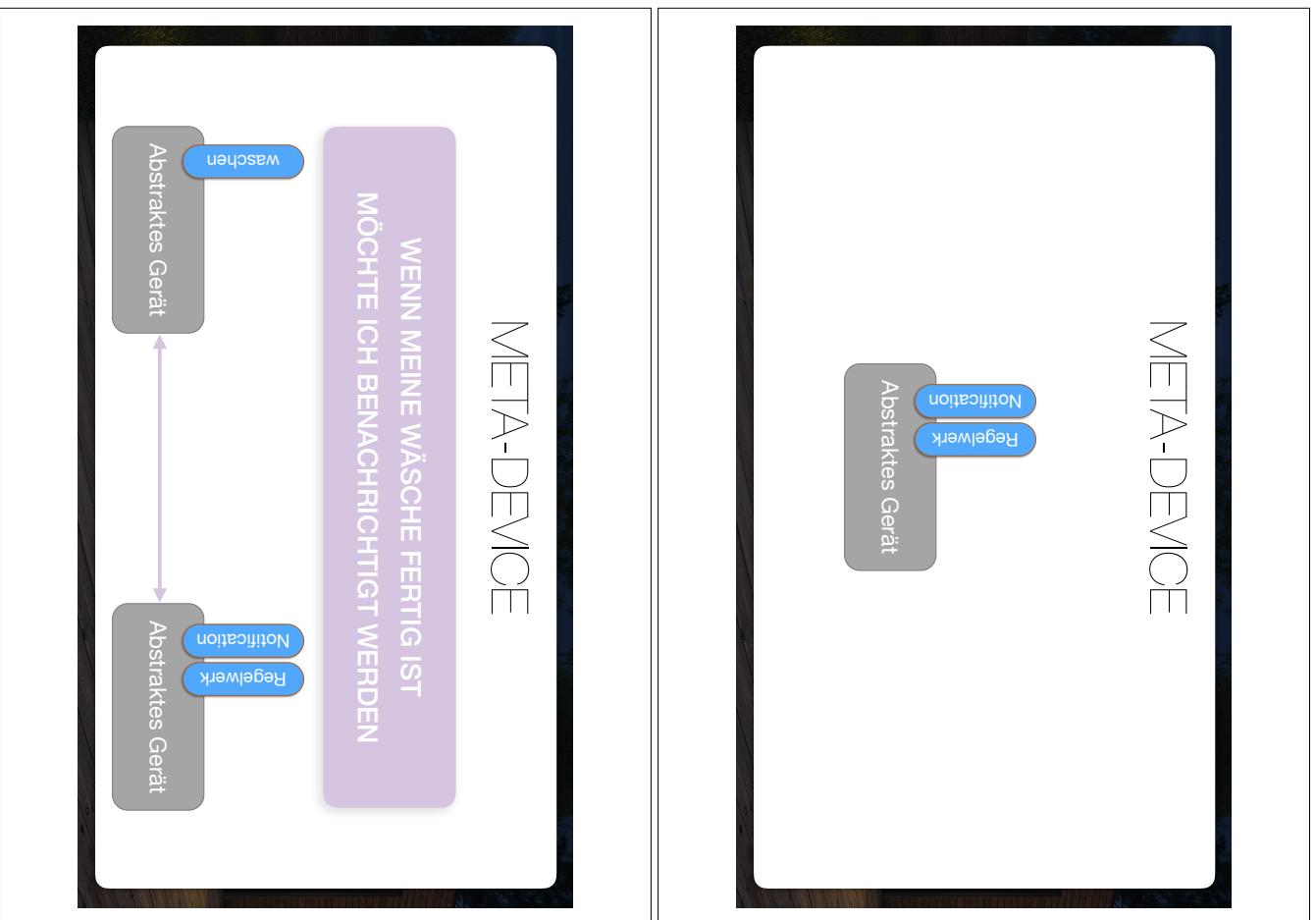
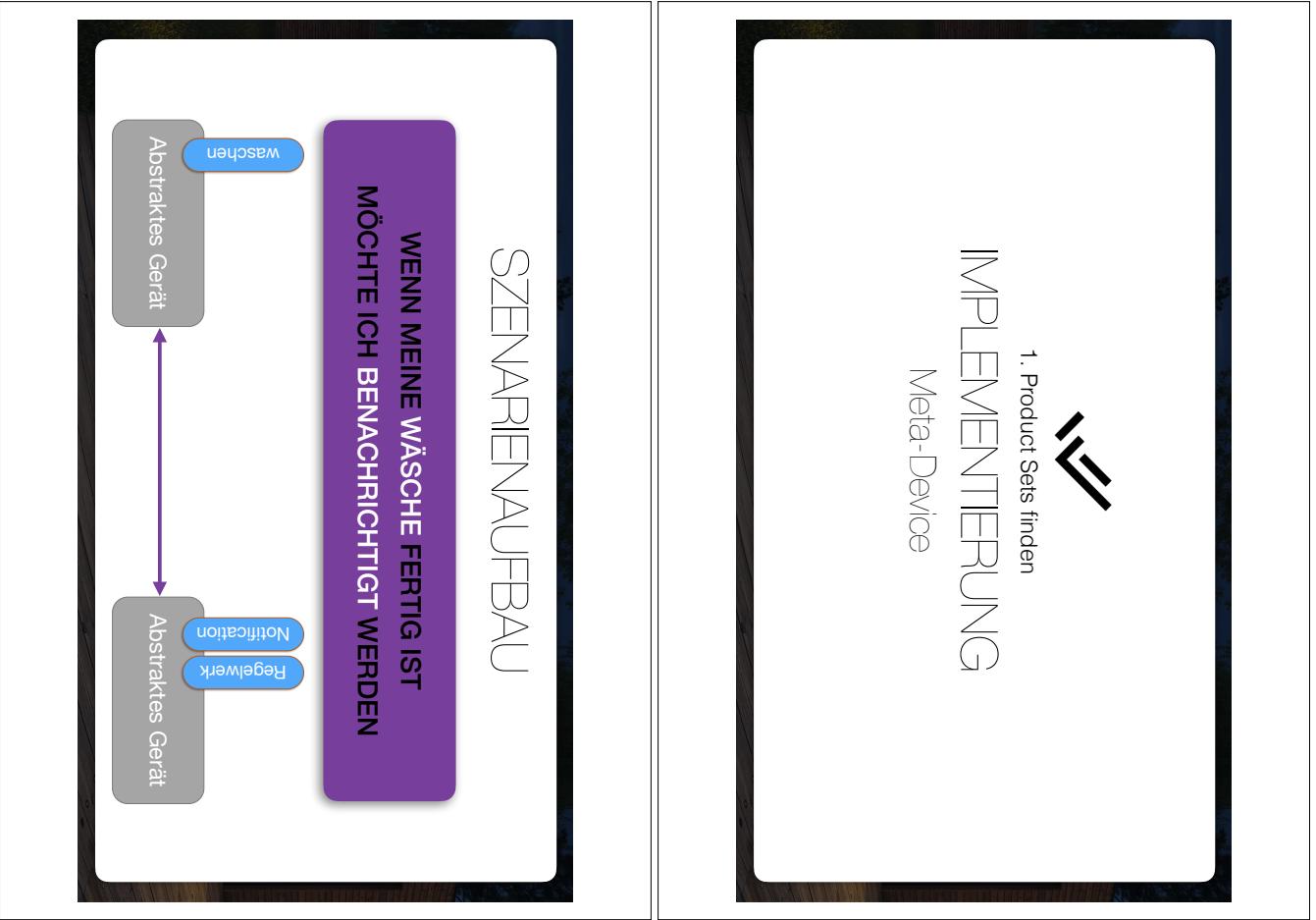
### 2. Szenarien mergen

- \* Größte Übereinstimmung finden
- \* Produkte kommen in anderen Szenarien vorkommen
- \* Optimales Produktset für Warenkorb ermitteln

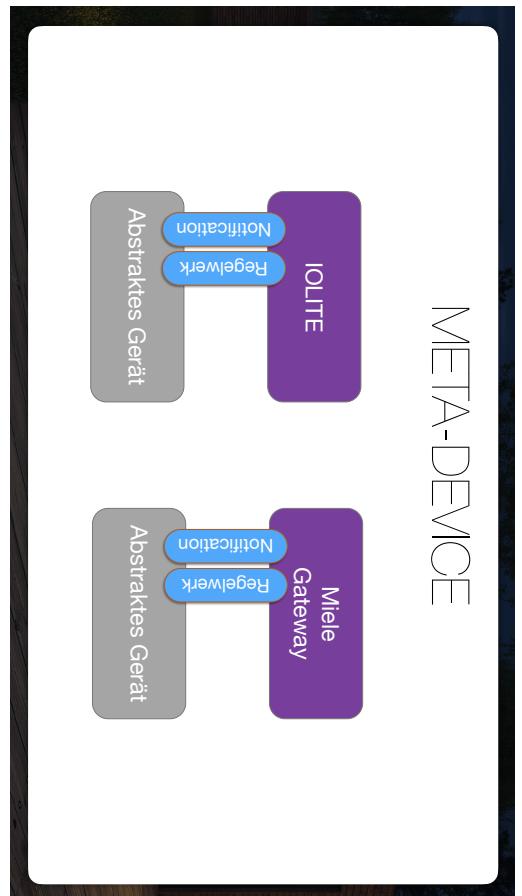
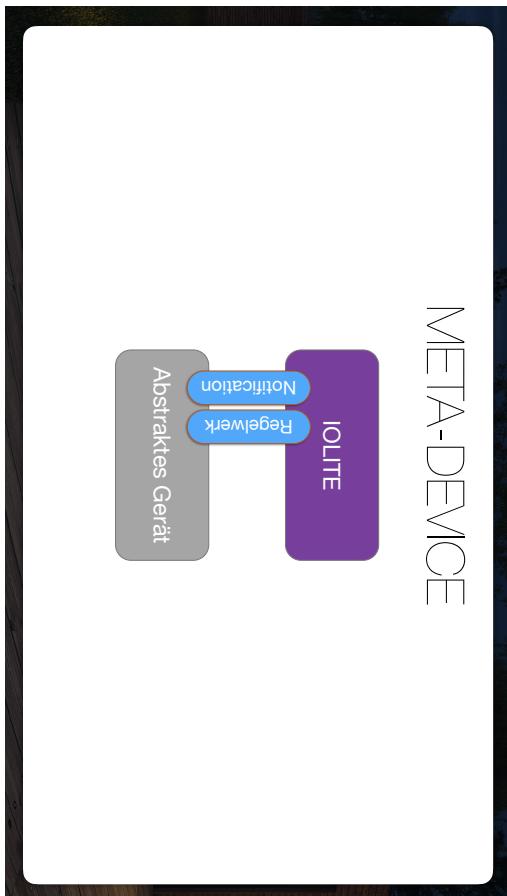
### 3. Produkte austauschen

- \* Funktional identische Produkte zum Austausch anbieten
- \* Produktabhängigkeiten prüfen
- \* Neues optimales Produktset für Warenkorb ermitteln

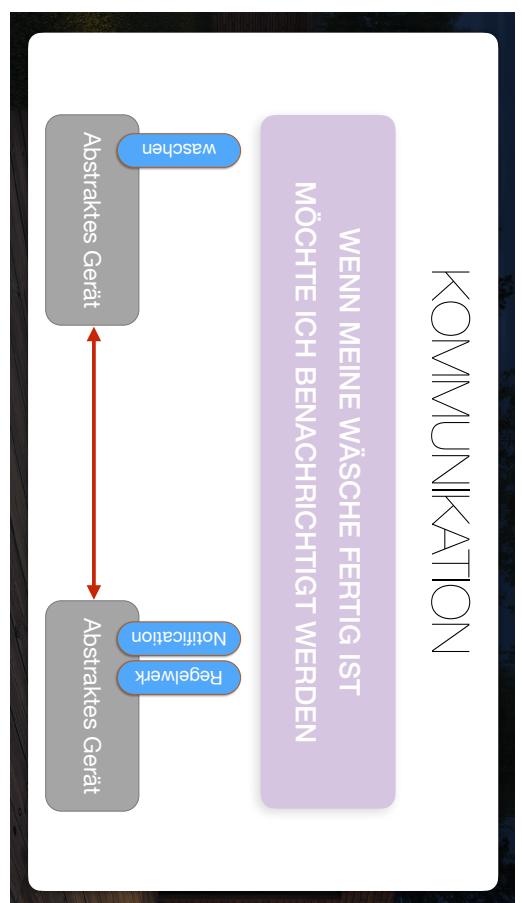




## META-DEVICE



## KOMMUNIKATION

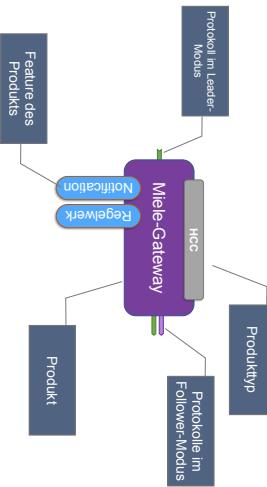


IMPLEMENTIERUNG  
Kommunikation

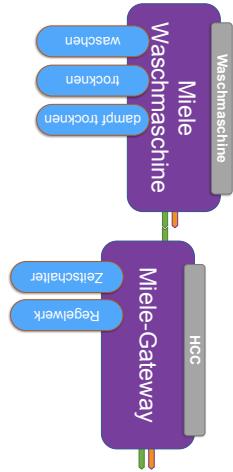


1. Product Sets finden

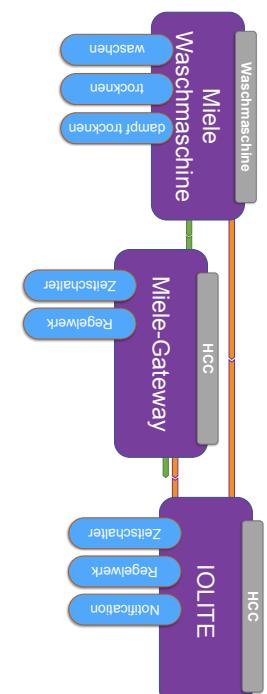
# KOMMUNIKATION



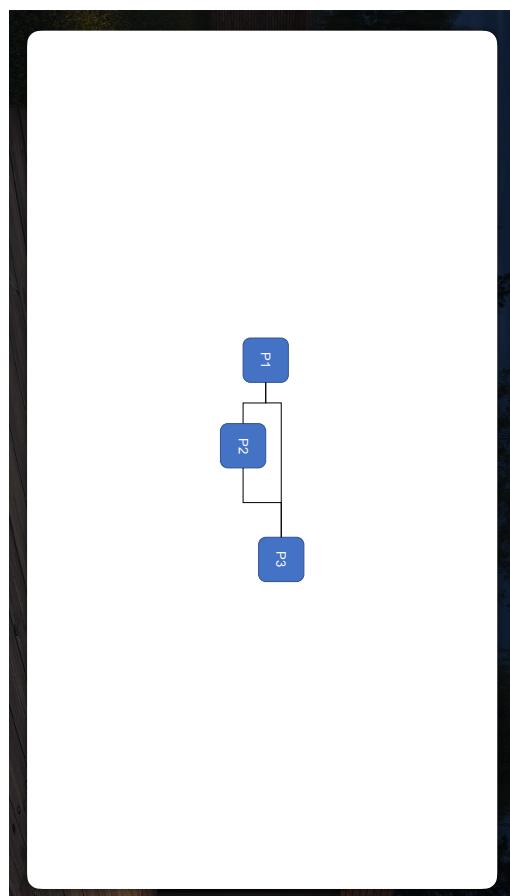
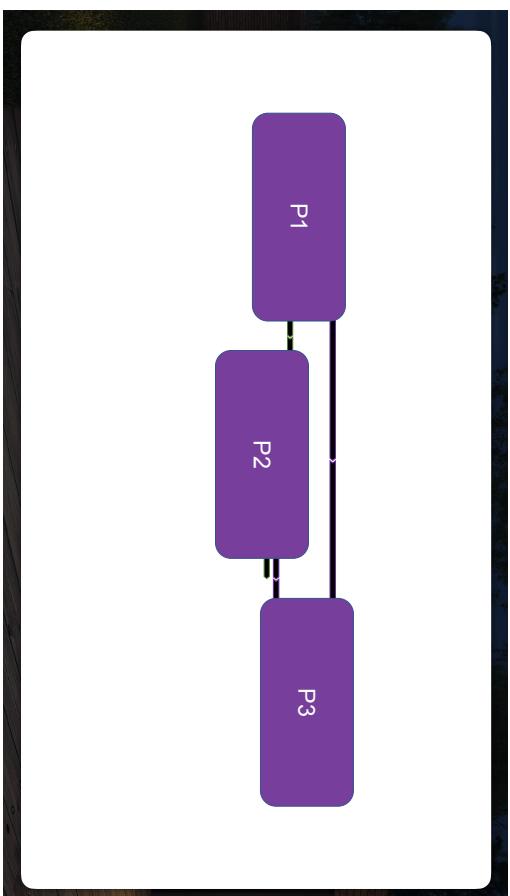
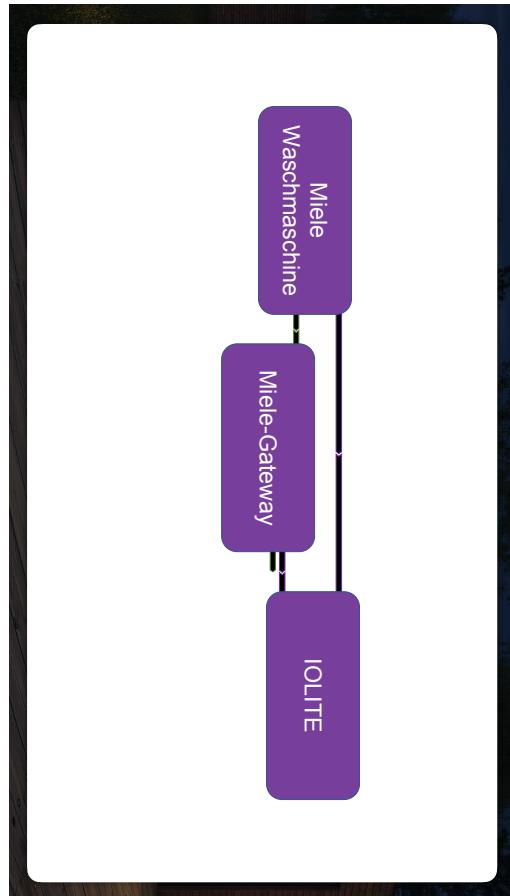
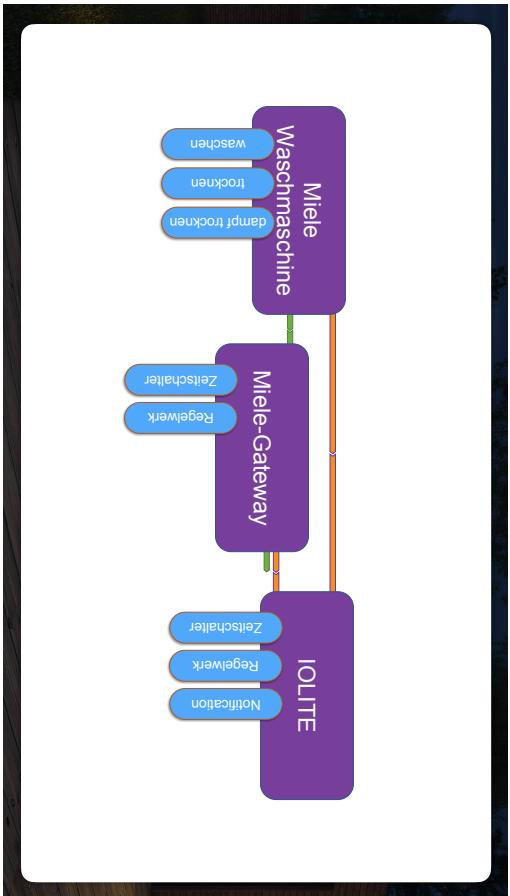
# KOMMUNIKATION

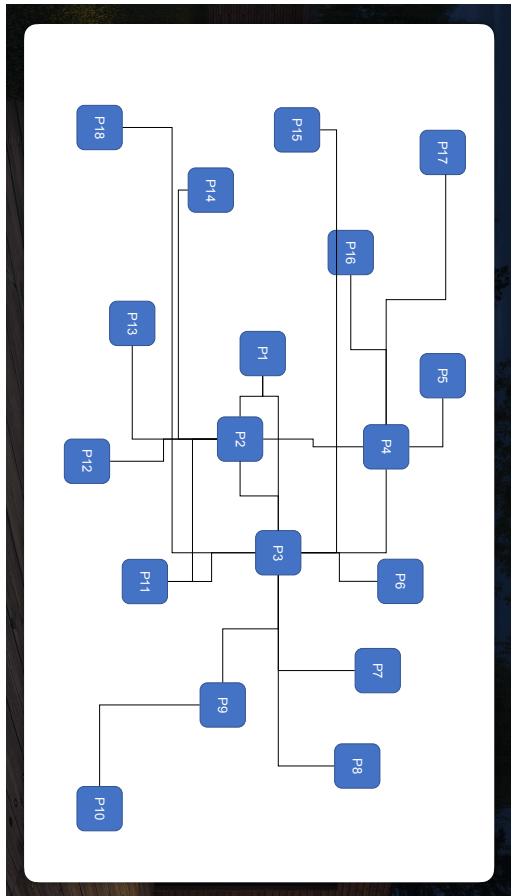
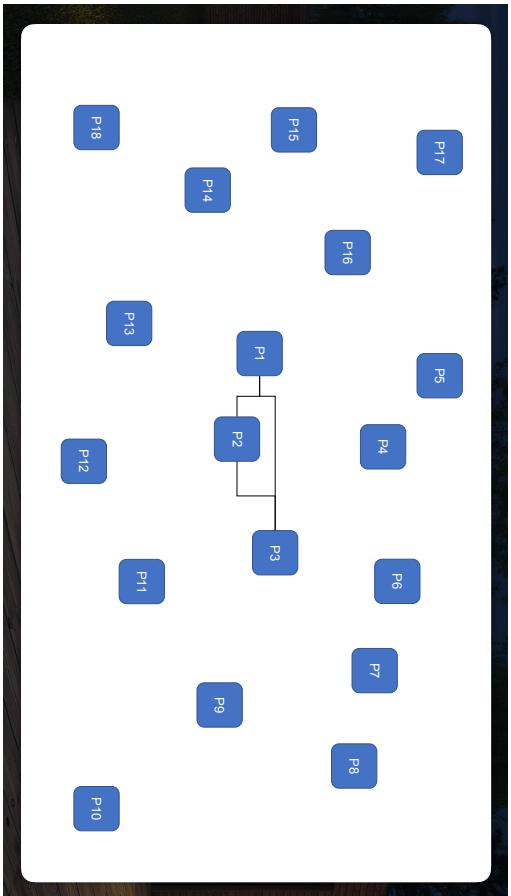


# KOMMUNIKATION



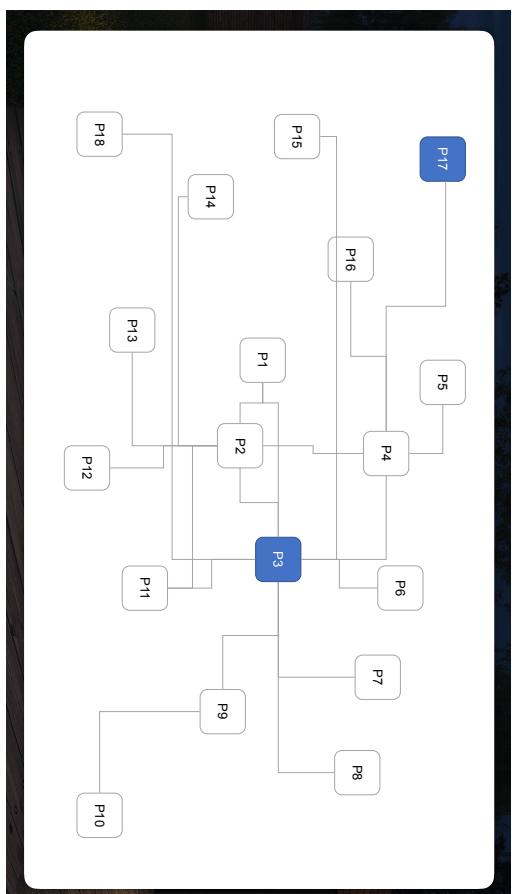
LET'S REDUCE IT

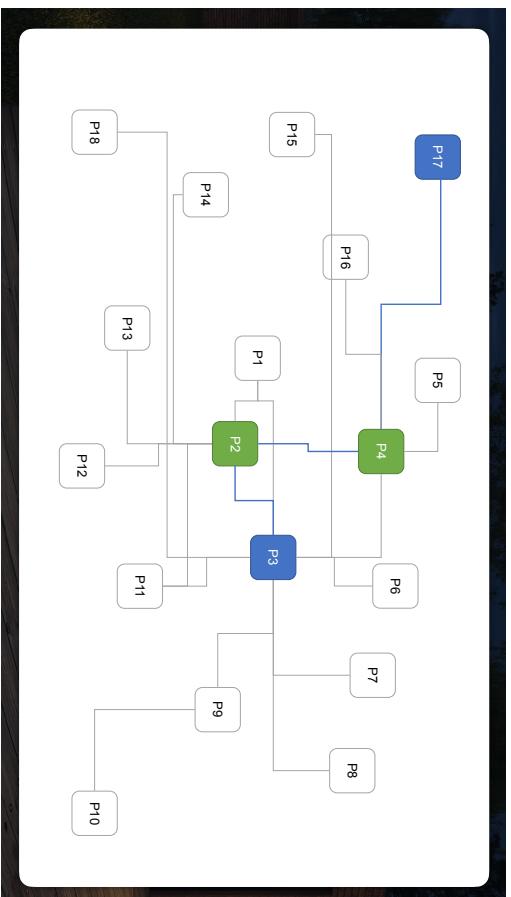
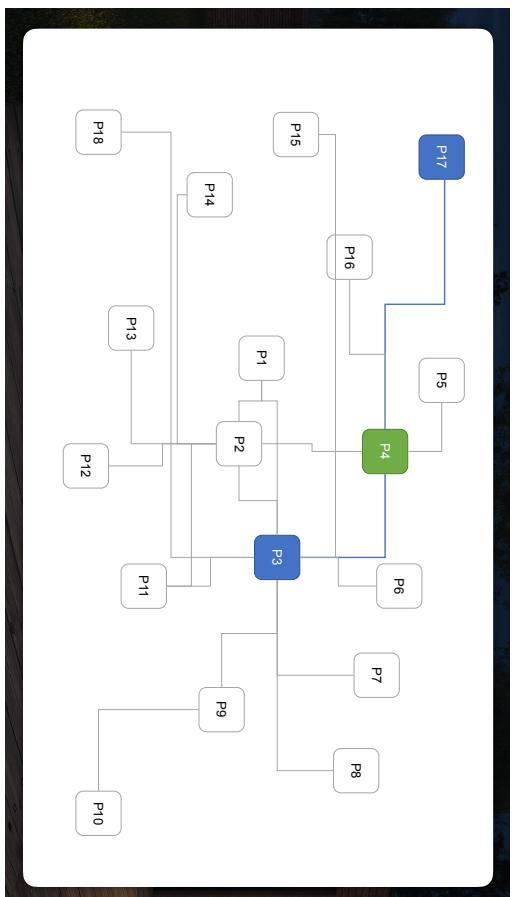
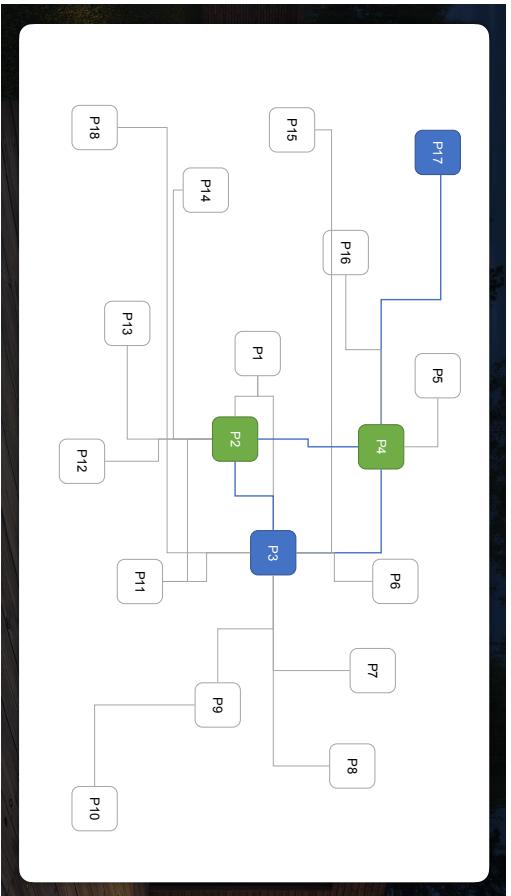
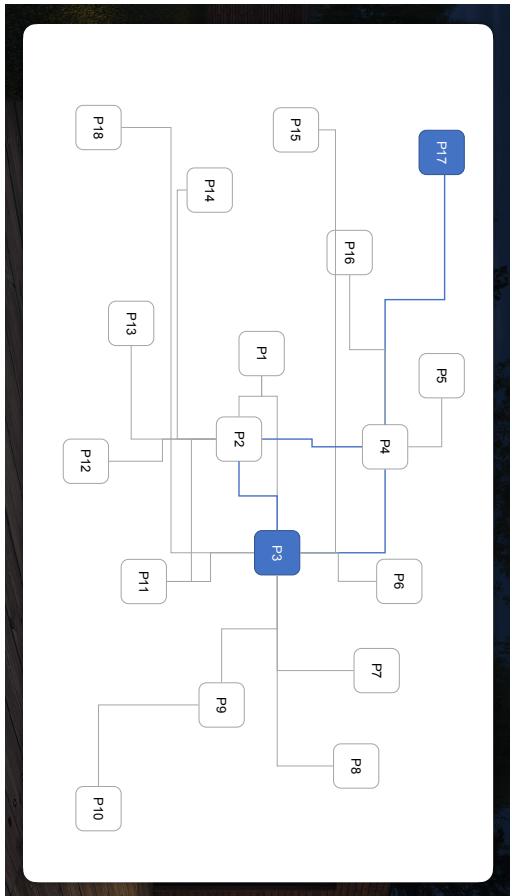


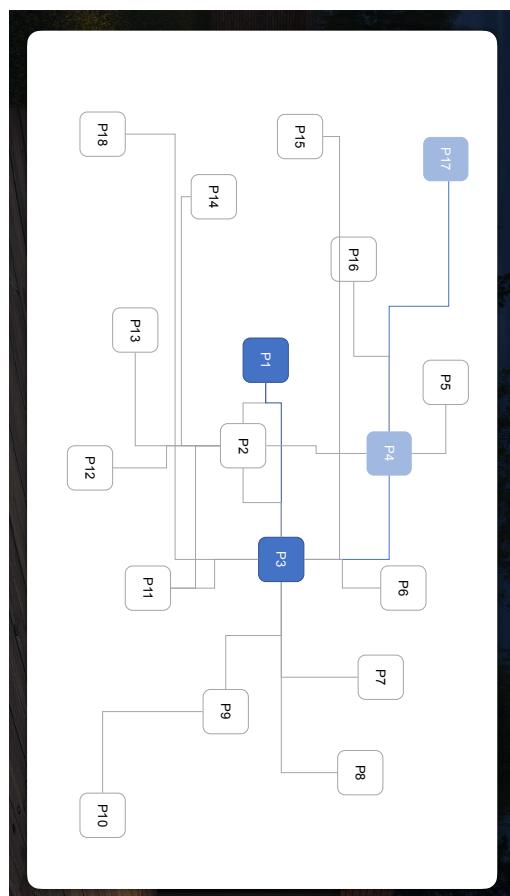
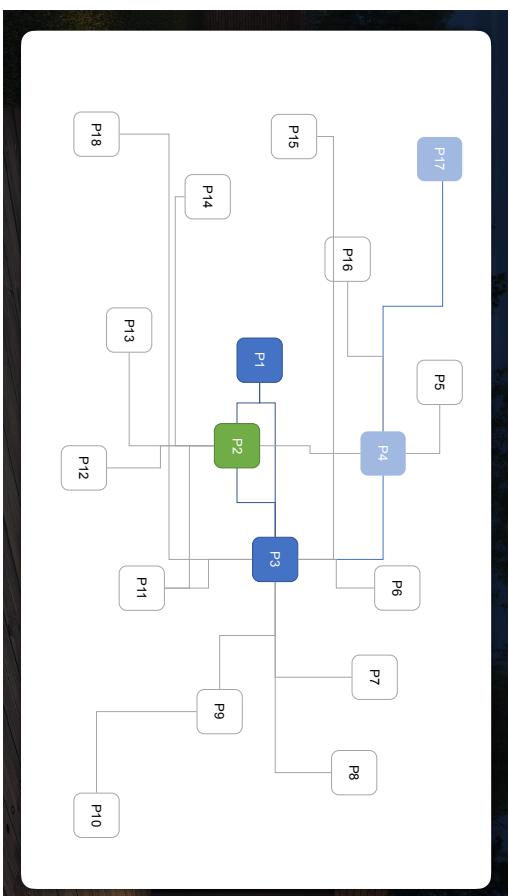
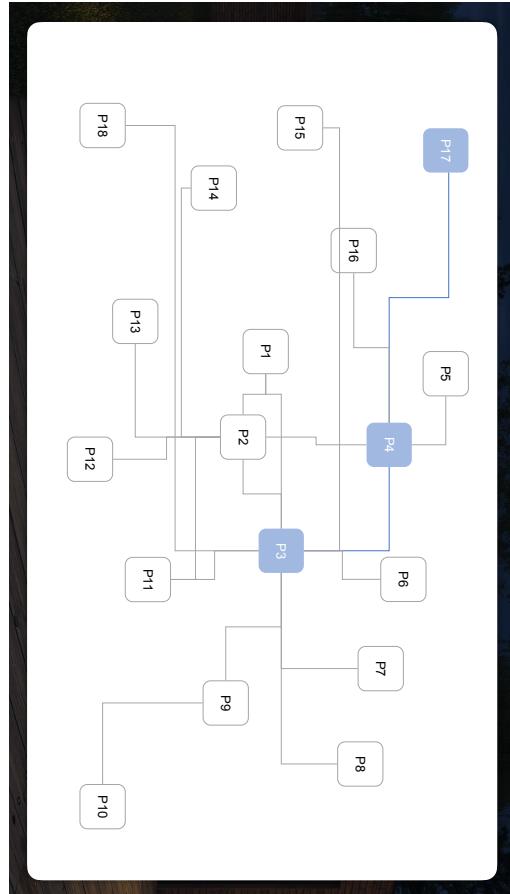
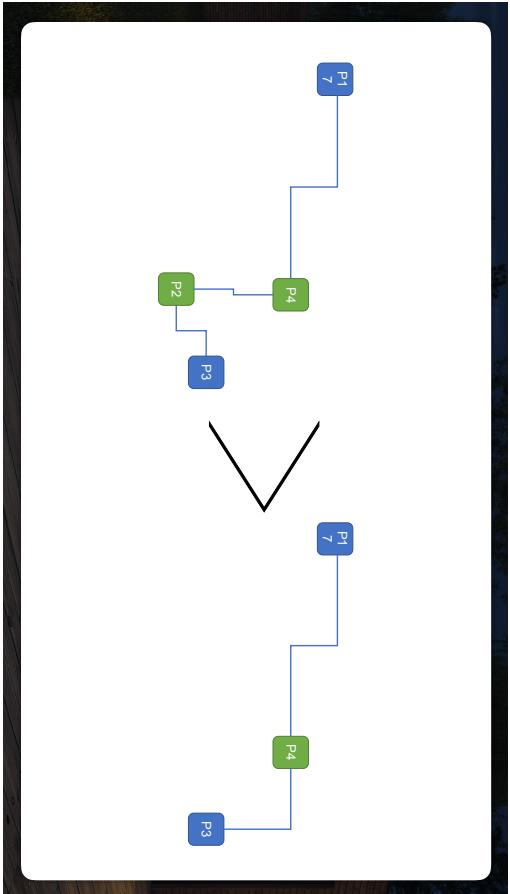


# REKURSIVE TIEFENSUCHE

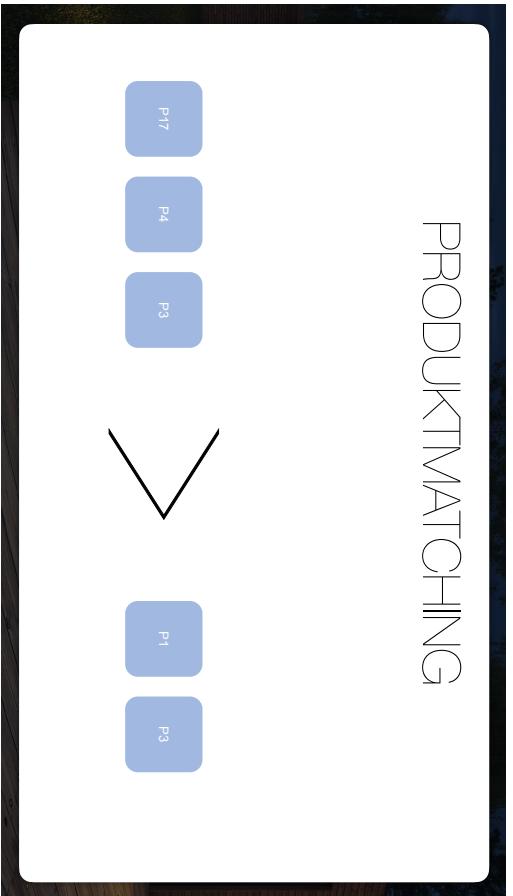
Finde alle Wege wie ein Produkt mit einem anderen reden kann







## PRODUKTMATCHING

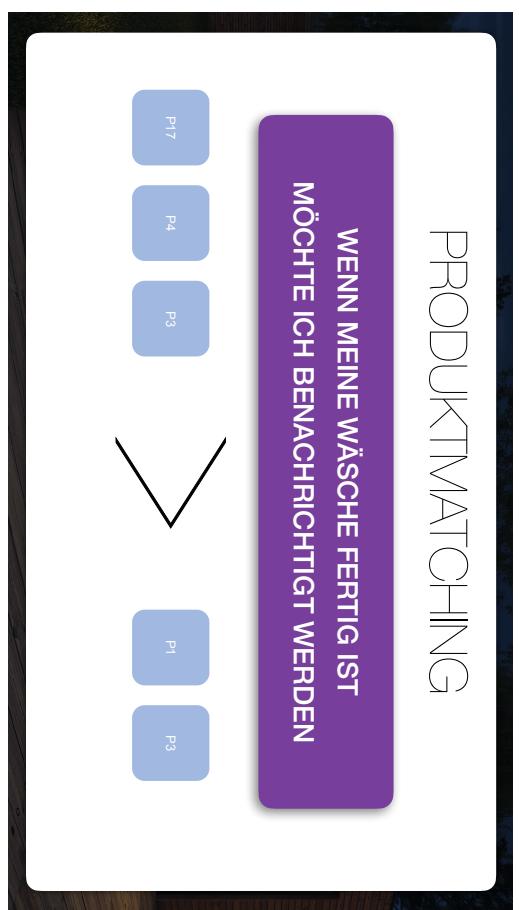


WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN



## PRODUKTMATCHING

WENN MEINE WÄSCHE FERTIG IST  
MÖCHTE ICH BENACHRICHTIGT WERDEN

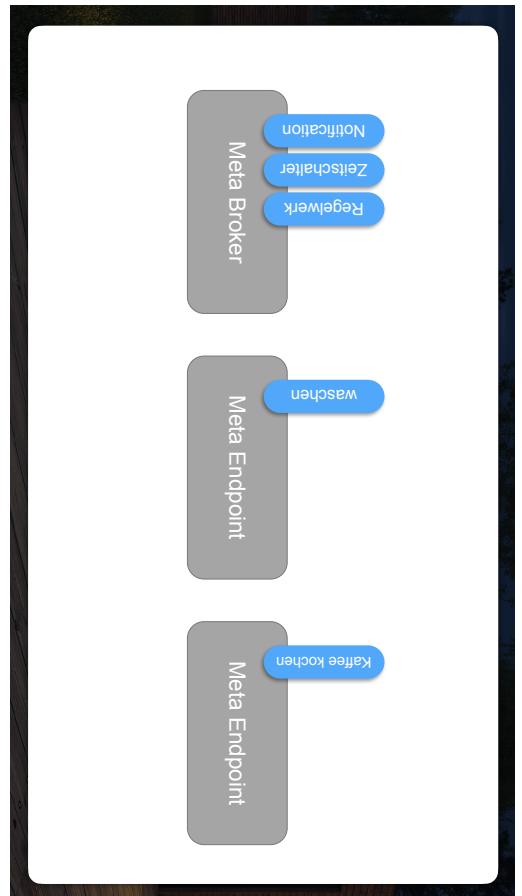
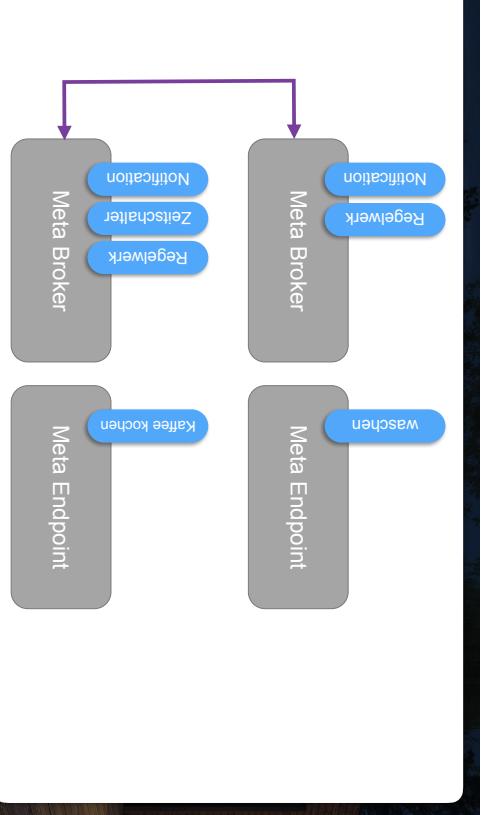
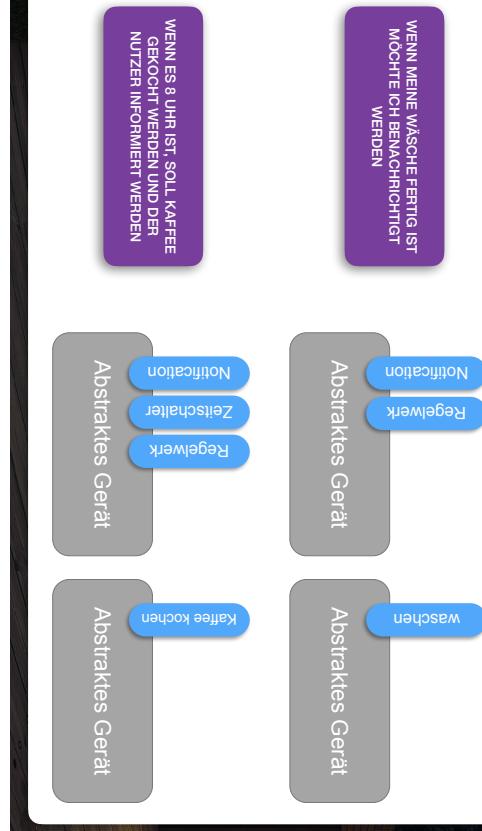


## SZENARIO MERGING

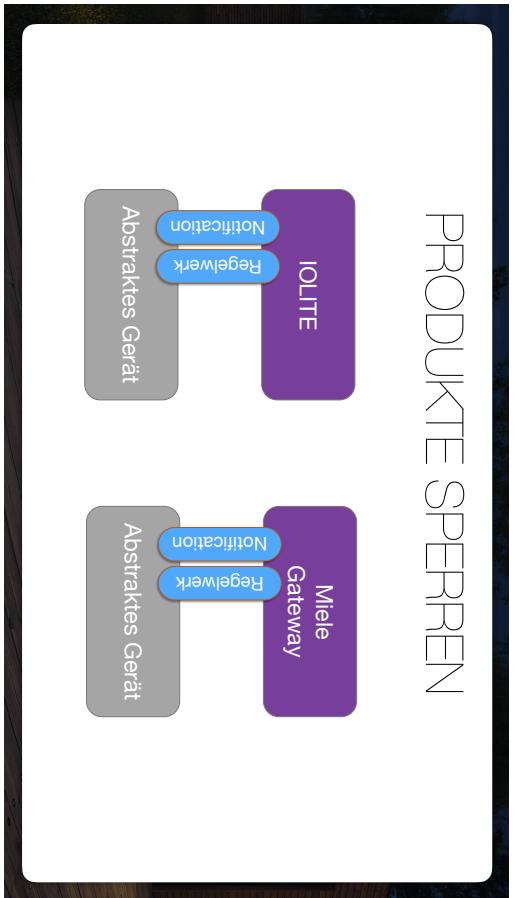
Merging von zwei Szenarien



2. Szenarien mergen



## PRODUKTE SPERREN

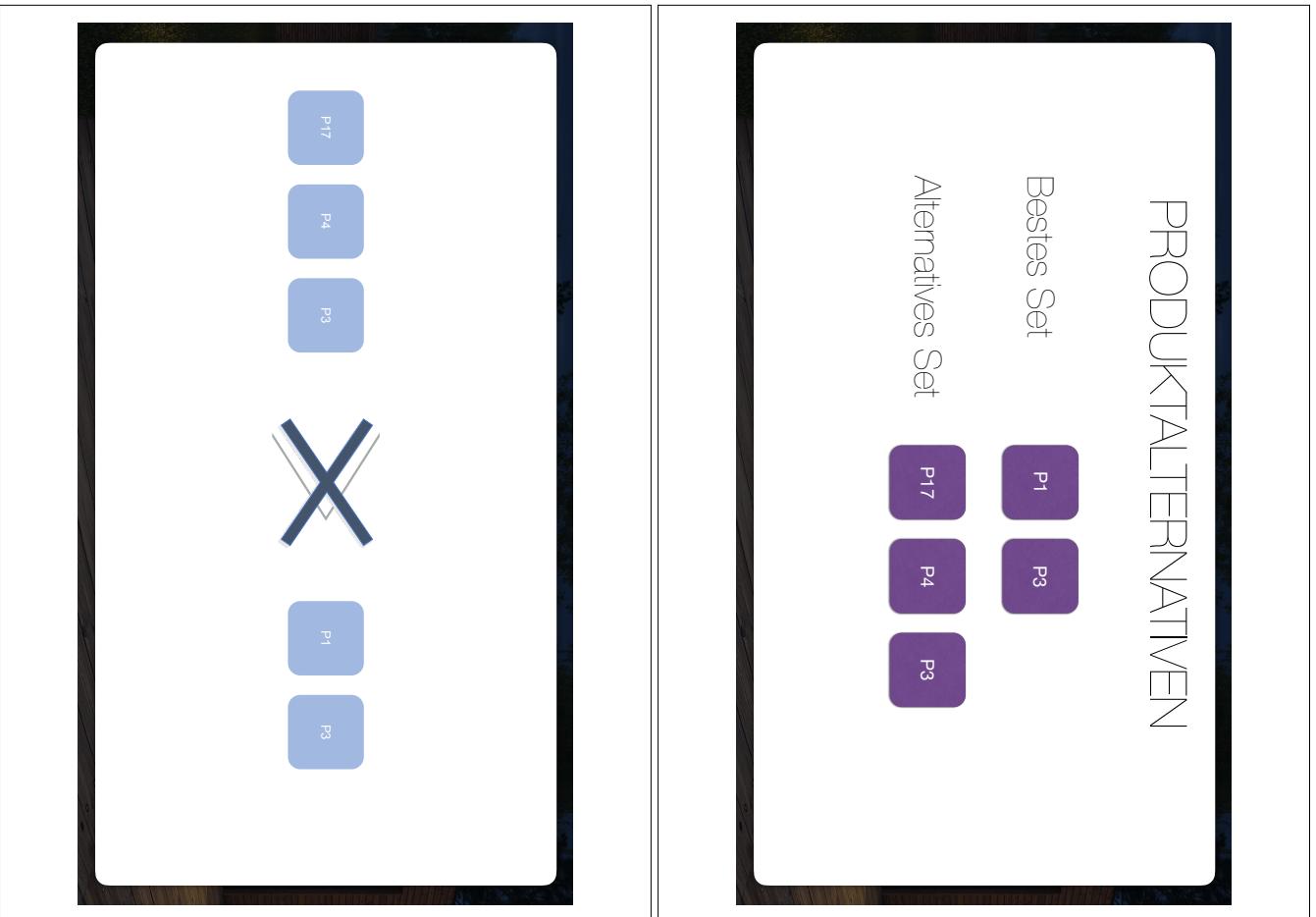
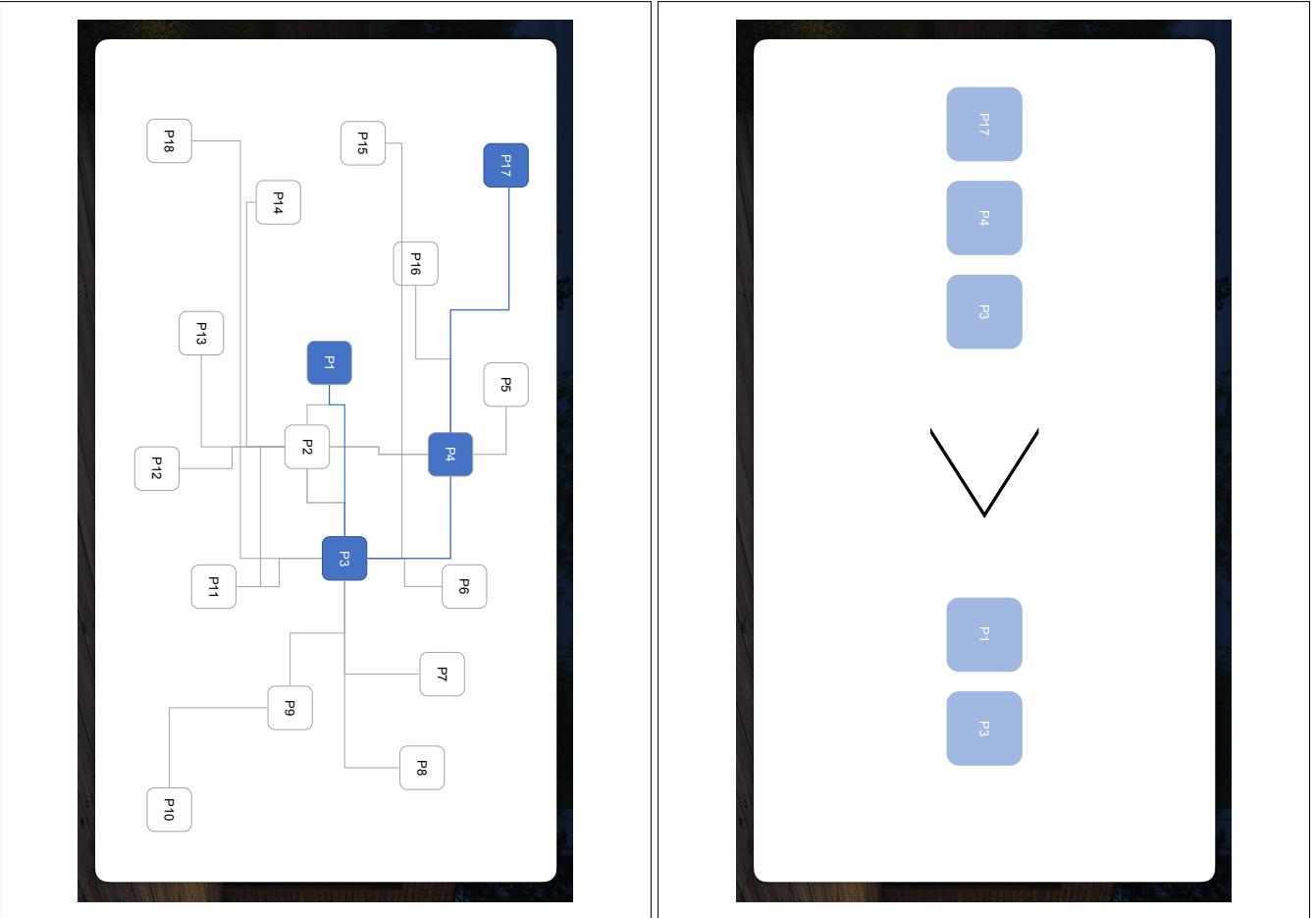


## PRODUKTALTERNATIVEN



## PRODUKTE SPERREN





## BACKEND Zusammenfassung

### ★ Matching

★ Komplexität ist NP und in  $O(n^m)$  mit m Anzahl des Meta-Devices im Warenkorb, n Anzahl Produkte in DB

★ Caching reduziert die Ladezeiten signifikant; deterministisch

★ Performance-Maximierung durch minimieren von Datenbank-Requests

★ Finden eines optimalen Produktssets von Szenarien

★ Bestes Ergebnis vs. Ladezeiten

### ★ Erweiterbarkeit

★ Räumliche Variablen

★ Genaue Analyse von Protokollen

BACKEND  
ZUSAMMENFASSUNG

## VER SCHritte

### Grundfragen

Beim Start des Advisors werden drei Grundfragen gestellt

1

2

3

4

### Szenarioauswahl

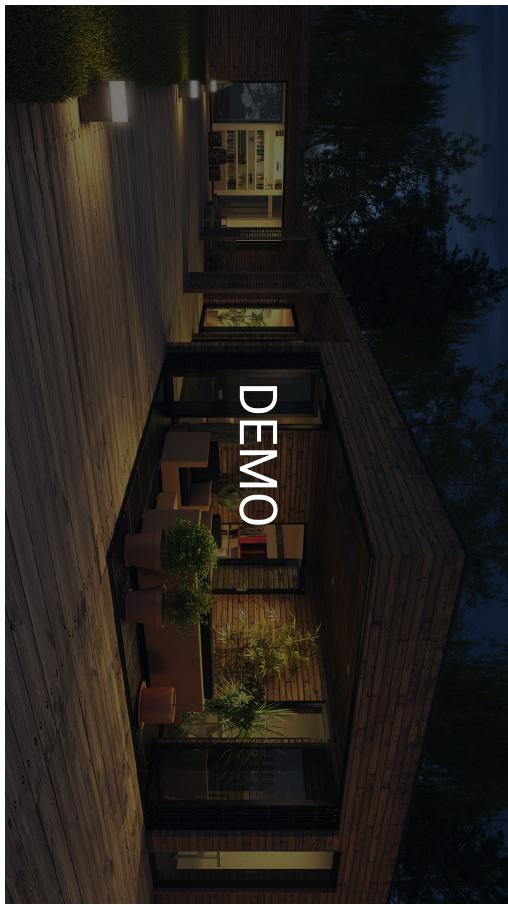
Anhand der Antworten aus Schritt 1 werden passende Szenarien gezeigt

### Produktauswahl

Die passenden Produkte zu den Szenarien inkl. Austauschen / Präferieren

### Dein Smart-Home

Die Übersicht aller passenden Produkte



DEMO

