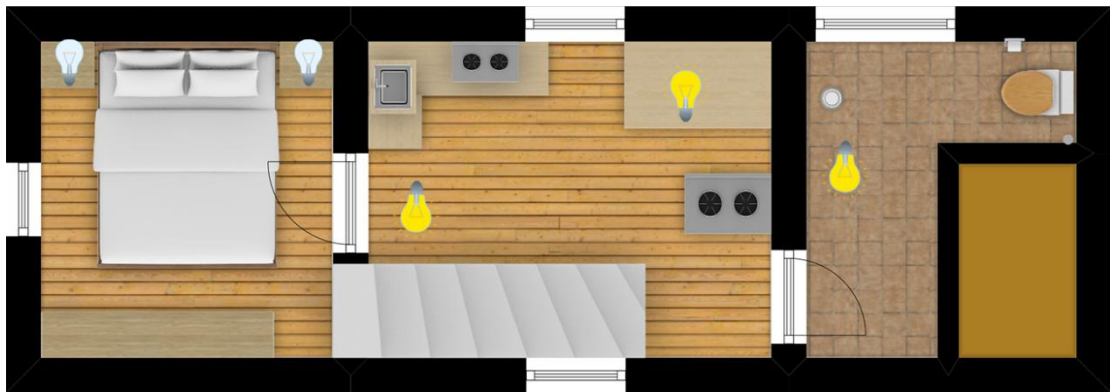


Eindopdracht mySmartBulb



Opdracht: Building a smart Bulb

Daniëlle Mertens

1 september 2020

In opdracht van NOVI Hogeschool

Inhoudsopgave

Inleiding.....	1
1 Vooronderzoek en requirements.....	2
1.1 mySmartBulb beschrijving.....	2
1.2 Functionele Requirements.....	2
1.3 Niet-functionele requirements.....	2
1.4 Technische haalbaarheid.....	3
1.5 Economische haalbaarheid.....	3
1.6 Operationeel en organisatorische haalbaarheid.....	3
1.7 Risico en beveiligingsanalyse.....	3
2 Technisch en functioneel ontwerp.....	5
2.1 Technisch ontwerp.....	5
2.2 Eisen en wensen: MosCow.....	6
2.3 Class diagram.....	7
2.4 User Case Diagram.....	8
2.5 Sequence Diagram.....	9
3 UX / UI	10
3.1 Resultaten persoonlijke interviews.....	10
3.2 Persona.....	10
3.3 User stories.....	11
3.4 Customer journey map.....	11
4 Prototyping.....	12
4.1 Paper prototyping.....	12
4.2 Schetsen.....	13
4.3 Usability testing.....	14
5 Installatie Guide	15
5.1 Benodigde programma's:.....	15
5.2 Waar staat de code?.....	15
5.3 Logins.....	16
6 Verantwoordingsdocument.....	17
6.1 Homepage.....	17
6.2 Dashboard.....	19
6.3 Login module.....	19
6.4 Chat functie.....	21
6.5 CRUD RESTFul API.....	22

Inleiding

De lampen die bij je thuis aangaan wanneer de zon ondergaat. De verwarming die je niet meer zelf moet programmeren, maar die zich aanpast aan jouw levensritme. De gordijnen die automatisch opengaan wanneer het licht wordt buiten. Met domotica kun je heel veel verschillende systemen aansturen.

Ik heb voor deze eindopdracht gekozen om domotica te gebruiken voor het aansturen van lampen. Het leek me een leuke opdracht om uit te voeren, en het biedt voor de toekomst een hoop mogelijkheden om deze applicatie verder uit te kunnen breiden. Het lijkt me ook heel leuk om deze applicatie uit te breiden speciaal voor mensen met een beperking, dat je echt iemand kan helpen hiermee.

Voordat ik deze applicatie heb kunnen bouwen heb ik eerst een analyse gedaan. Deze wordt in dit hoofdstuk 1 uitgewerkt. Hieruit volgt een technisch ontwerp: hoofdstuk 2. In hoofdstuk 3 en 4 ga behandel ik gebruikerservaringen en prototyping.

Ten slotte leg ik uit hoe je de applicatie kunt installeren en geef ik uitleg over de keuzes die ik wel of niet gemaakt heb en waarom.

Daniëlle

1 Vooronderzoek en requirements

Voor mijn eindopdracht ga ik een webapplicatie maken op basis van een Smart Home applicatie, waarmee ik lichten kan aansturen door middel van een app en sensors.

1.1 mySmartBulb beschrijving

De mySmartBulb app is een applicatie waarmee je lampen individueel aan- en uit kunt zetten en waarmee je lampen kunt dimmen. De lampen kunnen aangestuurd worden, door op de app de lampjes te klikken of door de slider te gebruiken om een lamp te dimmen. In dit hoofdstuk ga ik informatie verzamelen en achterhalen wat de behoefte is van deze applicatie.

1.2 Functionele Requirements

Op een interface wordt een display getoond, waar gedetecteerd en/of gemeten parameters getoond worden. Deze waarden ga ik visualiseren door middel van knoppen en een slider, gemaakt in HTML5 / Javascript.

Dit zijn de requirements die ik in de eerste instantie in gedachte had:

1. Alle lichten kunnen aan of uit per kamer of verdieping
2. Als de deurbel gaat, gaat automatisch het licht in de gang fllikkeren
3. Lampen gaan aan of uit bij zonsopgang of zonsondergang
4. Als een kamer meer dan 1 uur leeg is dan gaan de lampen uit
5. Bij bedtijd gaan de lampen naast het bed op de goede lees stand
6. Met een knop alles kunnen uitzetten
7. Snack licht: zodat je de weg naar de ijskast kunt vinden
8. Als natuurlijke licht minder dan 50% is dan lichtsterkte is 50%, als natuurlijke licht minder dan 40% is, lichten gaan naar 60% etc.
9. Als de grond te droog wordt, gaan de sproeiers on de tuin aan
10. Automatisch de tijd van de sproeiers kunnen instellen, maar niet nooit sproeien als de zon volop schijnt.

1.3 Niet-functionele requirements

- De app moet via Internet te benaderen zijn
- De Informatie moet juist en volledig zijn
- Je moet kunnen zien wat de status van de lampen is op ieder moment
- Betrouwbaarheid: fout bestendig
- Usability (begrijpelijk, gebruikersgemak, aantrekkelijk om te zien)
- Snelheid
- Makkelijk te onderhouden en over te dragen

1.4 Technische haalbaarheid

De technieken die gebruikt gaan worden bestaan al of kunnen worden gemaakt. Ik heb gebruik gemaakt van Java, JavaScript en HTML, dus de lichten gaan uiteindelijk niet echt real time branden. Ik heb een simulatie gemaakt, zodat de applicatie wel kan laten zien wat er precies gebeurt. Aangezien we tijdens deze Bootcamp al deze kennis verworven hebben, is deze applicatie technisch te realiseren.

1.5 Economische haalbaarheid

Economisch haalbaar betekent heel simpel dat de baten van een informatiesysteem meer zijn dan de kosten en de inspanning. De opbrengsten van deze applicatie zijn zowel op financieel vlak (inzicht in energiegebruik, zodat er zo zuinig mogelijk met de elektriciteit wordt omgegaan, op menselijk vlak (meer woongenot en ontspanning) en ecologisch vlak (de start voor een duurzaam huis).

De applicatie is in dit geval altijd economisch haalbaar, omdat het hier maar om 1 huishouden gaat. In een verdere fase als je alles in huis zou kunnen aansluiten op de applicatie en deze aanbieden aan andere huishoudens, zou je alsnog een economische haalbaarheidsanalyse kunnen doen.

1.6 Operationeel en organisatorische haalbaarheid

Operationeel zullen er wat zaken aangepast moeten worden. De applicatie gaat immers lampjes aansturen, waardoor de knoppen op een andere manier worden aangestuurd, als dit niet automatisch gebeurt. Ik zie verder geen problemen in deze nieuwe manier van werken. Wat betreft de kwaliteitseisen: snelheid, volume en betrouwbaarheid is het wel belangrijk deze van te voren te toetsen (is er voldoende internetverbinding, wordt alles goed aangestuurd en zit er geen lek in de applicatie).

1.7 Risico en beveiligingsanalyse

Domotica biedt niet automatisch meer veiligheid. Signaleren van gevaar is niet hetzelfde als voorkomen van gevaar (schijnveiligheid!).

Omdat het lastig te beschrijven is voor mijn applicatie, benoem ik voor risicoanalyse enkele algemene nadelen.

1) Kosten

Uiteraard kunnen sommige domotica systemen gepaard gaan met een stevig prijskaartje. De opties zijn eindeloos maar zijn niet altijd even goedkoop.

2) Gedecentraliseerd

Er zijn veel verschillende leveranciers, merken en soorten van domotica. En daar ligt ook het nadeel. Want deze toepassingen zullen niet altijd perfect met elkaar communiceren. Hierdoor zal je vaak verschillende applicaties moeten beheren om je volledige woning te bedienen.

3) Tijdrovend

Vaak heb je enorm veel mogelijkheden op het vlak van domotica waardoor je soms door de bomen het bos niet meer ziet. Het vraagt dan ook wel wat tijd om het maximum uit elke domotica toepassing te halen.

4) Veiligheid

Veiligheid is zowel een voordeel is als een nadeel. Als hackers in staat zijn om een slim apparaat te hacken, kunnen ze mogelijk de lichten en alarmen uitschakelen, en deuren openen. Daardoor zou een inbreker zonder inbraaksporen een huis kunnen verlaten. Verder kunnen hackers mogelijk toegang krijgen tot het netwerk van de huiseigenaar, wat ernstige gevolgen kan hebben. Ze kunnen apparaten ontregelen en bijvoorbeeld vanaf een afstand door jouw camera de woning bekijken.

5) Complex

De reden dat in Nederland nog weinig mensen een smart home hebben, komt mede door achterblijvende lanceringen van technologieën en de complexiteit van een smart home. Smart home fabrikanten werken hard aan het verminderen van de complexiteit en het verbeteren van de gebruikerservaring om het plezierig te maken voor alle mogelijke gebruikers.

6) Privacy

Naast beveiliging zijn veel smart home tegenstanders bezorgd over de privacy van gegevens. Producenten van smart home producten en systemen hebben consumentengegevens nodig om hun producten beter af te stemmen op de gebruiker. Vertrouwen en transparantie zijn van cruciaal belang voor fabrikanten om vertrouwen op te bouwen bij de gebruikers van hun slimme producten.

2 Technisch en functioneel ontwerp

In dit hoofdstuk ga ik op in het technisch en functioneel ontwerp van de bulb app. Ik heb een MoSCoW overzicht gemaakt, enkele schetsen toegevoegd en relevante diagrammen.

2.1 Technisch ontwerp

In het technisch ontwerp achterhaal ik wat de gebruiker wil bereiken met de functionaliteit. De eindgebruiker is op zoek naar:

- Gemak
- Meer controle over zijn of haar woning
- Energie en geld besparen
- Veiligheid

Wat moet het informatiesysteem doen voor de gebruiker?

Via de mySmartBulb app kun je vanaf je telefoon of pc je huis vanaf afstand bedienen en controleren. Ook kun je zien wat de actuele status van de lampjes is vanaf elke locatie (of je nu op je werk bent of thuis).

Wat doet het informatiesysteem?

Met de mySmartBulb app kun je lampen aansturen, je kunt ze aan - uitzetten, dimmen etc.

Welke functies krijgt het informatiesysteem?

- Lichten apart van elkaar aan- en uitzetten
- Als de deurbel gaat, gaat automatisch het licht in de gang flinkeren
- Lampen gaan aan of uit bij zonsopgang of zonsondergang
- Als een kamer meer dan 1 uur leeg is dan gaan de lampen uit
- Bij bedtijd gaan de lampen naast het bed op de goede lees stand
- Met een knop alles kunnen uitzetten
- Snack licht: zodat je de weg naar de ijskast kunt vinden
- Als natuurlijke licht minder dan 50% is dan lichtsterkte is 50%, als natuurlijke licht minder dan 40% is, lichten gaan naar 60% etc.
- Als de grond te droog wordt, gaan de sproeiers on de tuin aan
- Automatisch de tijd van de sproeiers kunnen instellen, maar niet nooit sproeien als de zon volop schijnt.

2.2 Eisen en wensen: MosCow

De eisen en wensen breng ik in kaart door middel van het MosCow model op basis van de eerder aangegeven requirements. Ik heb vanwege tijdsgebrek niet alles kunnen realiseren, ik zal aan het einde dit model nog een keer gebruiken maar dan met wat wel en niet gerealiseerd is.

Must have

- ✓ Alle lampen kunnen aan of uit per kamer of verdieping
- ✓ Een lamp die je kunt dimmen
- ✓ Alles wordt opgeslagen in een database, zodat je ook vanuit een andere kamer de status van de lamp kunt bekijken
- ✓ User login met rollen (admin, editor)
- ✓ Lichtsensor en/of bewegingssensor

Should have

- ✓ Snack licht: zodat je de weg naar de ijskast kunt vinden

Could have

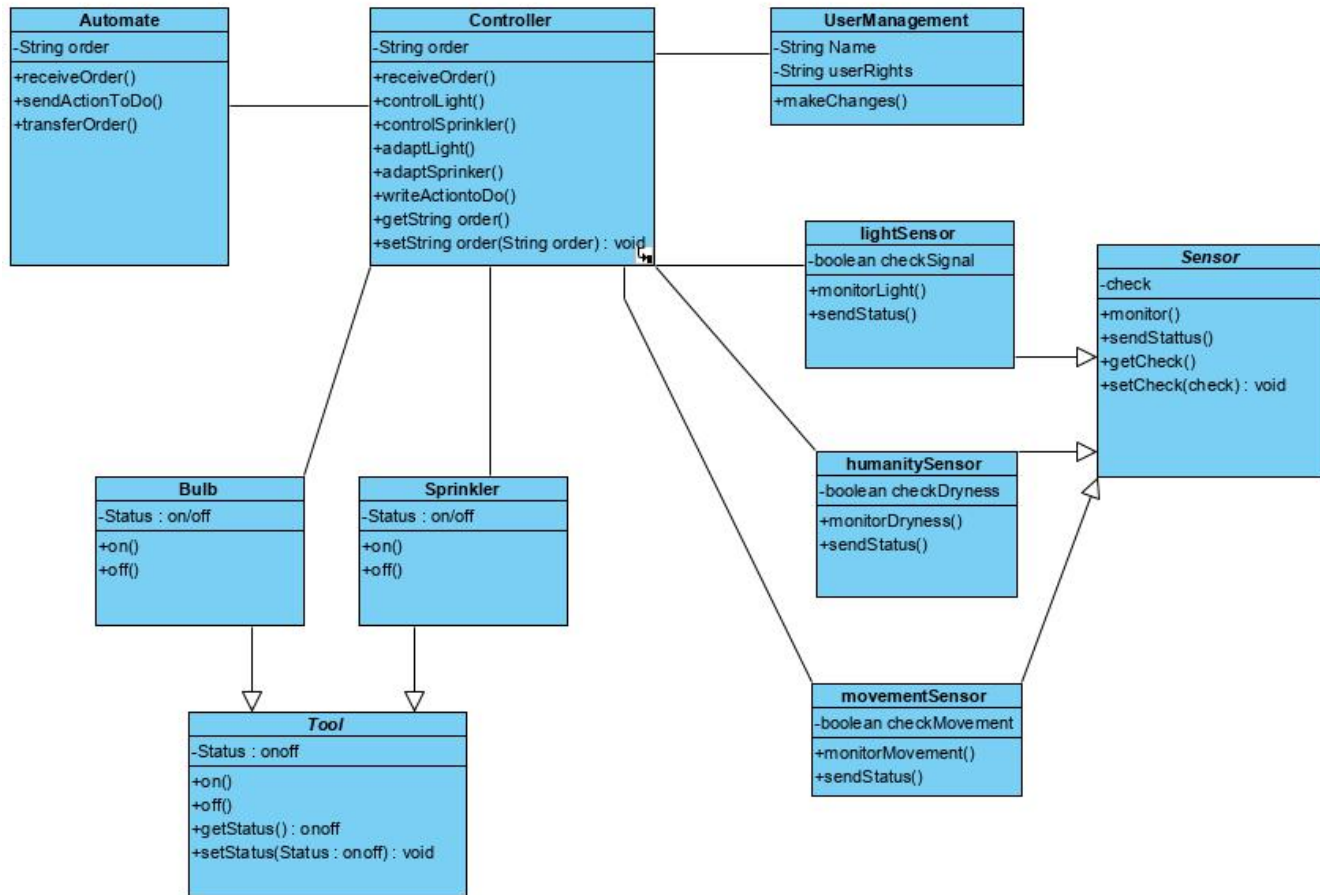
- ✓ Sprinkler aansturen: als de grond te droog wordt, gaan de sproeiers on de tuin aan
- ✓ Automatisch de tijd van de sproeiers kunnen instellen, maar niet nooit sproeien als de zon volop schijnt.

Won't have

-

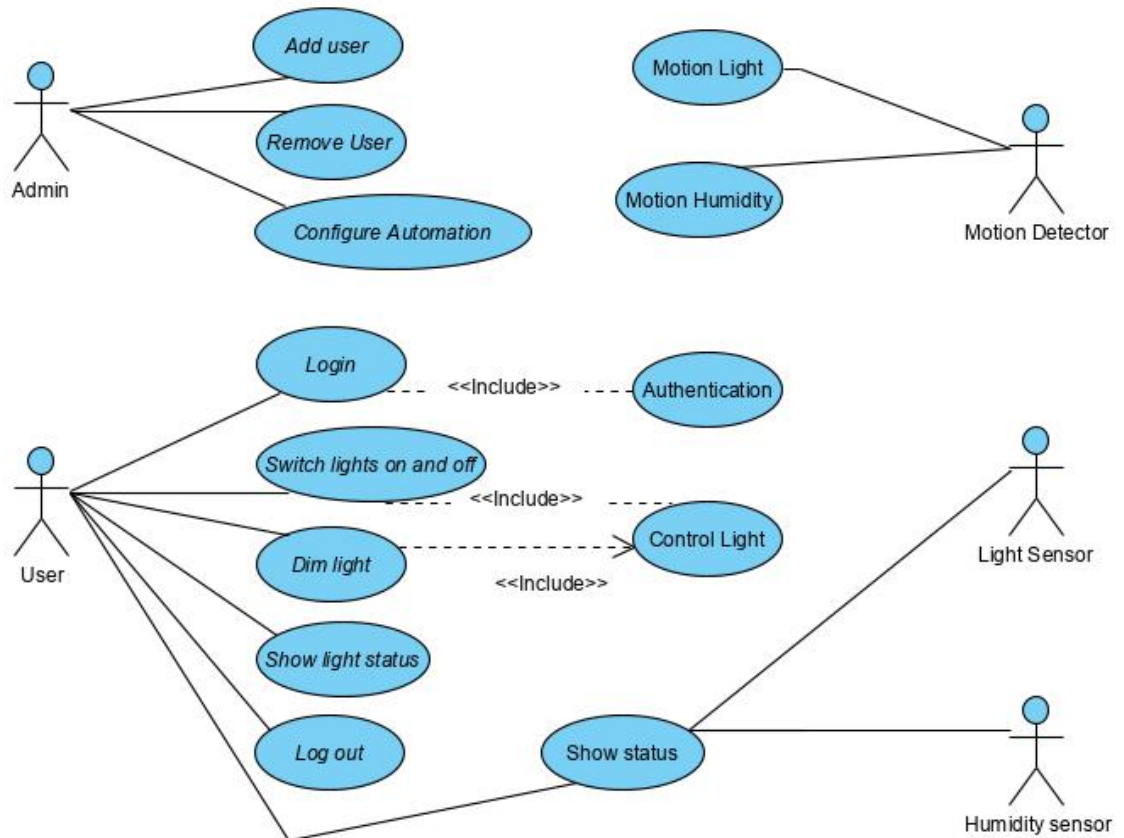
2.3 Class diagram

In onderstaand class diagram worden de verschillende classes en methodes die gebruikt weergegeven.



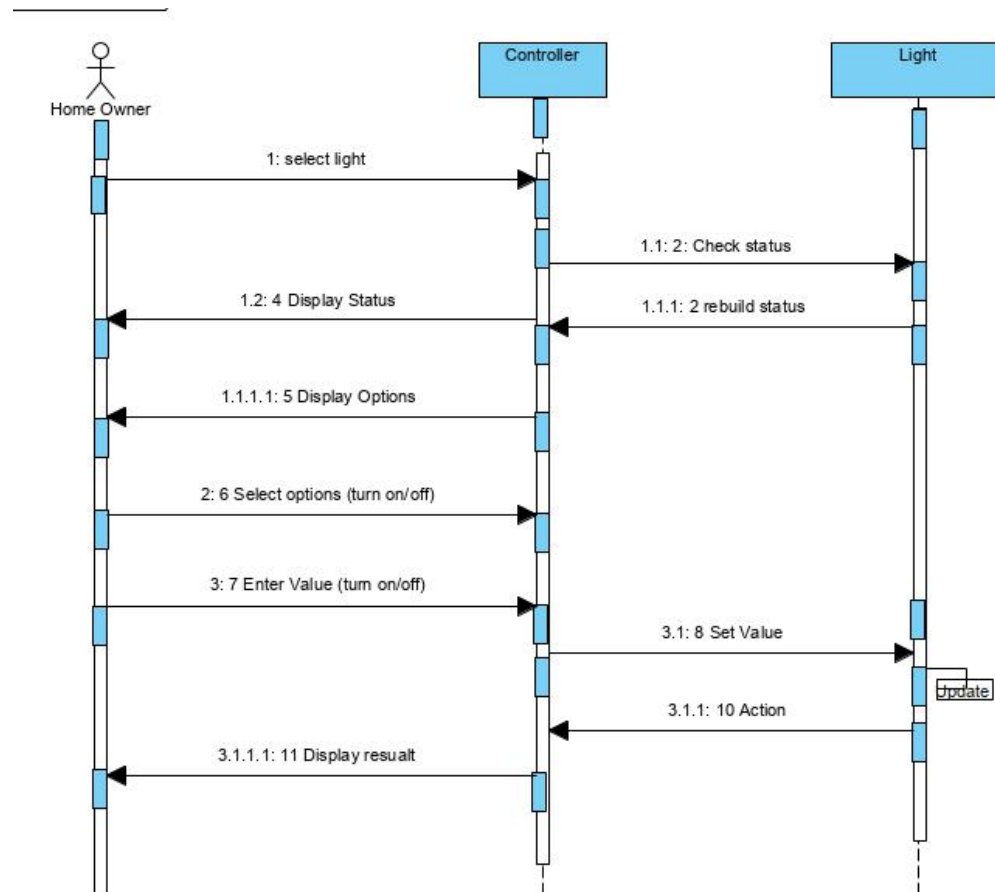
2.4 User Case Diagram

Onderstaand Use Case Diagram geeft een grafisch overzicht van de functionaliteiten van de app, namelijk usecases en de onderlinge relaties tussen deze usecases.



2.5 Sequence Diagram

Onderstaand sequentie diagram, toont de verbanden tussen de verschillende processen door de oproepen tussen verschillende objecten in een bepaalde sequentie te tonen.



3 UX / UI

Met een UX ga ik kijken naar de ervaring van gebruikers, namelijk:

- Technisch: hoe krijg ik toegang tot de dienst? Hoe presteert de service?
- Logisch: kan ik de taken uitvoeren die ik moet doen? Begrijp ik de interface?
- Emotioneel: geniet ik van het gebruik van de service?
- Transformationeel: maakt de service mijn leven beter?

3.1 Resultaten persoonlijke interviews

Om te onderzoeken of deze app gebruiksvriendelijk is en doet wat het moet doen heb ik een aantal mensen gevraagd om er eens naar te kijken en doorheen te klikken en gevraagd wat ze ervan vonden. Doel is om te achterhalen wat de gebruikers ervan vonden en eventuele bugs of missing elementen te achterhalen.

De resultaten van deze persoonlijke interviews zijn:

- ✓ Technisch: Login werkt niet
- ✓ Logisch: Als je de deur opent komt deze tegen het bed aan
- ✓ Technisch: Het weer image is statisch en niet actueel (opgelost)
- ✓ Emotioneel: de sprinkler is die voor buiten of heb je een terrarium binnen staan?
- ✓ Logisch: Woon je in een caravan?
- ✓ Technisch: Wat gebeurt er als ik settings klik?
- ✓ Technisch: Links in het menu werken niet (opgelost)
- ✓ Technisch: Ik kan de status per lamp niet zien (opgelost)
- ✓ Transformationeel: kun je dat ook voor mijn huis bouwen?
- ✓ Emotioneel: graag uitbreiden voor andere apparaten

3.2 Persona

Als uitgangspunt heb ik een vriendin genomen die pas een duurzaam huis heeft laten bouwen. Haar jongste dochter heeft een fysieke beperking, waardoor ze heel beperkt gebruik kan maken van haar armen.

Met dit als uitgangspunt zien hun persona's er als volgt uit:



3.3 User stories

Hieronder een korte beschrijving van mySmartBulb, geschreven vanuit het oogpunt van die eindgebruiker.

“Als gebruiker wil ik mijn lampen kunnen aansturen. Ik wil de lampen aan- en uit kunnen zetten, ik wil de lampen kunnen dimmen en ik wil dat de lampen automatisch dimmen op basis van de hoeveelheid licht. Ik wil de lampen manueel kunnen aansturen, maar dit moet ook automatisch kunnen.”

“Als hoofdgebruiker wil ik lampen kunnen toevoegen of verwijderen en de status van de lamp zien. “

“Als gebruiker wil ik de lampen kunnen bedienen, zowel vanuit thuis of vanaf afstand.”

3.4 Customer journey map

Uiteindelijk wil je je hele huis kunnen aansturen met een app. Een customer journey zou er dan zo uit kunnen zien.

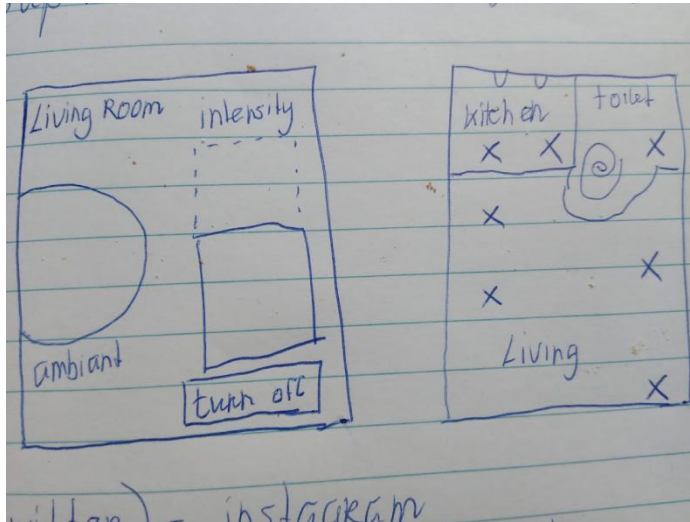


4 Prototyping

Gedurende de Bootcamp kwam langzaam een prototype naar voren. Een prototype is "een eerste voorlopig model van iets. Vanuit hier kun je andere vormen ontwikkelen en kopiëren.


4.1 Paper prototyping

Het begon met een simpele schets, om de eerste gedachten te schetsen:



4.2 Schetsen

Tot een meer uitgewerkte versie in de vorm van schetsen: de eerste schetsen, waarmee de applicatie opgezet.

Home	Lights	Sprinkler	Login	Settings
<p>Hallo!</p> 				
<p>Actueel weer</p>				

Home	Lights	Sprinkler	Login	Settings
<p>Gebruikersnaam: <input type="text"/></p> <p>Wachtwoord: <input type="password"/></p> <p><input type="button" value="Inloggen"/></p>				
<p>Actueel weer</p>				

4.3 Usability testing

Dit doen we aan de hand van de 10 heuristieken van Nielsen

1: Laat zien wat er gebeurt.

Je kunt lampjes aan en uit zitten door erop te klikken of een lampje dimmer met een zichtbare slider. Ook is een dashboard waarin je in een oogopslag de status van het lampje per kamer kunt zien.

Heuristiek 2: Overeenkomst systeem en echte wereld

Omdat ik met deze app geen echt lampen aanstuur, heb ik de lampjes gesimuleerd.

Heuristiek 3: Gebruiker heeft controle en vrijheid.

Gebruiker kan makkelijk de lampjes aan en uitzetten en in het dashboard ook meteen de totale status van de lamp. Makkelijk te vinden via het menu.

Heuristiek 4: Wees consistent.

Menu staat altijd bovenaan. Er is altijd een actueel weeroverzicht.

Heuristiek 5: Hulp bij fouten.

Er kunnen weinig fouten gemaakt worden in deze app, als er iets fout gaat, is een probleem in de code.

Heuristiek 6: Zorg dat de bezoeker niks hoeft te onthouden.

Alle statussen staan op het dashboard. Navigatie is eenvoudig. Makkelijk te switchen.

Heuristiek 7: Flexibiliteit en efficiency.

De website is afgestemd op de bezoekers.

Heuristiek 8: Hou het minimaal en verfijnd.

De website is eenvoudig in functionaliteiten en vormgeving. Functionaliteiten kunnen zeker uitgebreid worden, maar de vormgeving houd ik simpel en eenvoudig.

Heuristiek 9: Maak foutmeldingen minder eng.

Er kunnen weinig fouten gemaakt worden in deze app, als er iets fout gaat, is een probleem in de code.

Heuristiek 10: Bied een helpende hand (help en documentatie).

De website al helemaal duidelijk zijn zonder hulp.

5 Installatie Guide

5.1 Benodigde programma's:

Om de app te kunnen laten werken moet je de volgende programma's installeren:

Software	Versie	Rol	URL
Open jDK	8	Java Runtime	https://adoptopenjdk.net/
Intelij IDEA	2019.3 ultimate	Java + webeditor	https://www.jetbrains.com/
PostgreSQL	12.1	Database, SQL	https://www.postgresql.org/
Maven	3.6.3	Build system voor Java	maven.apache.org/
Springboot	2.2.2	Application container	https://start.spring.io/
Gradle	6.6.1	Build system voor Java	https://gradle.org/

5.2 Waar staat de code?

De Code is terug te vinden op <https://github.com/daaniejelle/Bulb>

Na het compileren is de site bereikbaar via <http://localhost:8080/>

- ✓ Alle Java files staan onder Models:
/JavaMVC-master/JavaMVC-master/src/main/java/com/example/restservice/Model
- ✓ De controllers zijn te vinden op
/JavaMVC-master/JavaMVC-master/src/main/java/com/example/restservice/Model
- ✓ De html, CSS, Javascript files en static templates zijn te vinden op
/JavaMVC-master/JavaMVC-master/src/main/resources

5.3 Logins

De database is te bereiken via PostgreSQL:

```
url = jdbc:postgresql://localhost:5432/Bulbs  
url = jdbc:postgresql://localhost:5432/UserDB  
user = postgres  
password =  banana007
```

Om in te loggen kunnen de volgende gegevens worden gebruikt:

Admin login: DaanAdmin
Password: 123

User login: DaanUser
Password: 123

6 Verantwoordingsdocument

Hier geef ik aan voor de verschillende functionaliteiten wat de functies doen en waar je de code kunt vinden.

- ✓ Ik heb Javascript en JQuery gebruikt om de applicatie dynamisch te maken. React heb ik vanwege tijdgebrek even buiten beschouwing gelaten.
- ✓ De app is getest met Google Chrome op een desktop, ik heb voor nu geen rekening gehouden met andere browsers of apparaten.
- ✓ Ik heb veel tijd moeten besteden aan het oplossen van de dependencies, ik kreeg alleen maar foutmeldingen. De dependencies moesten in Gradle worden gedefinieerd in niet in Maven. Dat heeft meer dan een week geduurd voordat ik hierachter gekomen ben.

6.1 Homepage

Voor het bouwen van de app heb ik de volgende uitgangspunten en functionaliteiten gebruikt:

- ✓ Java vormt de basis van de mySmartBulb app
- ✓ In mijn huis (<http://localhost:8080>) staat een image zonder lampen in een floorplanContainer (*index.html*, *coderegel 42*).
- ✓ Het menu is gemaakt in *Menu.css* en wordt opgeroepen in *Main.html* div menu nav.
- ✓ Het actuele weer is een widget en heeft verder geen connectie met andere files, (*index.html*, *coderegel 44* en *Main.js* *coderegel 1*).
- ✓ De greet functie zegt op basis van het tijdstip goedmorgen, goedemiddag of goedenavond (*Index.html*, *coderegel 33* en *Main.js*, *coderegel 9*).
- ✓ De klok toont de actuele tijd (*index.html*, *coderegel 26* en *Clock.js*).
- ✓ De lampjes kunnen aan en uit gezet worden of gedimd, deze informatie wordt opgeslagen in de database, zodat je overal kunt zien, wat de status en/of intensiteit van de lamp is, ook als je bijvoorbeeld in een andere kamer bent.

- ✓ Als je klikt op een (niet dimmable) lampje wordt de volgende methodes aangeroepen:

toggleBulb (bulbController.js, coderegel 41) = get and save in database
setBulbStatus (Bulb.js, coderegel 18) = fetch server response
bulbOnOffHandler (Bulb.js, coderegel 2) = lamp gaat aan of uit
bulbSelectHandler (Bulb.js, coderegel 132) = make Query Object

Is de lamp uit, dan wordt deze aangezet en andersom. Deze informatie wordt opgeslagen in de database door middel van de boolean on.

- ✓ De positie van de lampjes wordt bepaald door `positionBulbs()` (*Bulb.js*, *coderegel 88*). In de `fetch` methode wordt aangegeven wat het id van de image is, de src en CSS code (absoluut nulpunt, grootte en intensiteit. Deze code wordt uitgevoerd nadat hij van de server komt.
- ✓ Er is ook een lampje dat dimmable is. De dimmable lampjes worden in een vierkantje getoond, en kun je aansturen met een slider. Als je op een dimmable lampje klikt (de lampjes op beide nachtkastjes zijn dimmable) dan kun je met behulp van de slider het lampje feller of minder fel maken Als je met de slider (*Slider.css*) schuift dan worden de volgende methodes aangeroepen:

bulbIntensity (bulb.Controller.js, regel 21) = get and save in database
setIntensityStatus (Bulb.js, regel 70) = fetch server respons
rangeIntensity (Bulb.js, regel 38) = change intensity
bulbSelectHandler (Bulb.js, coderegel 132)

- ✓ Ook hier worden in de methode `positionBulbs` (*Bulb.js*, *regel 98*) de details aangegeven voor de afzonderlijke dimmable lampjes. Een src, een class, een `bulbSelectHandler` (wanneer je de slider positie wijzigt, weet je welke bulb selected is, namelijk `$(".dimmableBulb.selected")`).
- ✓ Omdat er heel veel berichten naar de controller gestuurd werden als je de slider bewoog, heb ik een timer gebruikt, die pas een bericht naar de controller stuurt als er 200 mSec geen nieuw slider event is geweest (*Bulb.js coderegel 36*).
- ✓ Sensorlampje: een lampje met de class `dimmableBulbSensor` gaat via een timer op een vast tijdstip lopen (in dit geval tussen 19.00 en 23.00 uur en tussen 06.00 en 9.00, waardoor het lampje steeds feller gaat branden, en om 23.00 wordt de lamp uitgezet. (*Bulb.js coderegel 62*). De timer is gemakkelijk in te stellen: de uren kunnen aangepast worden, de teller kan aangepast worden (let brightness), en de snelheid kan aangepast worden (staat nu op 10 seconden). Als de lamp vol brandt, wordt de intensiteit in de cache opgeslagen, zodat het niet steeds opnieuw begint te branden als je pagina bezoekt

6.2 Dashboard

Op het Dashboard (<http://localhost:8080/dashboard.html>) kun je de lampjes per kamer selecteren en zie je de actuele status van de lamp (is de lamp aan of uit, is de lamp dimmable, welke kleur heeft het lampje, waar staat het lampje, en wat is de intensiteit van het lampje. Code is terug te vinden in (Dashboard.html, Dashboard.js en Dashboard.css). In deze code wordt de informatie uit de server gehaald en in een dropdown gezet.

De id, aan/uit en intensiteit zijn dynamisch en komen uit de database, dus op het moment dat je iets wijzigt op de homepage wijzigt dat hier ook.

6.3 Login module

Gebruikers die inloggen kunnen alleen die pagina's zien die horen bij hun rol. Als ze toegang willen tot protected pagina's waarvoor ze geen rechten hebben, wordt de toegang geweigerd.

Hieronder de opbouw van de login applicatie:



In de database heb ik 3 tabellen aangemaakt, APP_USER, APP_ROLE en USER_ROLE. Er is ook een tabel PERSISTENT_LOGINS, die gebruikt wordt als token informatie (remember me functie).

In SpringBoot heb ik de volgende files aangemaakt:

- Config: WebSecurityConfig
- DOA: AppRoleDOA + AppUserDOA
- Model: AppRole + AppUser + UserRole
- Utils: EncryptedPasswordUtils + Webutils
- Service: UserDetailsServiceImpl
- LoginController

Templates

- 403
- Adminpage
- Loginpage
- Logout
- Userpage

Public

- Login.css
- UserTable.css
- UserTable.js

/userpage

Op de user page kun je (nadat je bent ingelogd) de gebruikers in de database bekijken. Om deze pagina te bekijken moet je inloggen als admin of user.

/admin

Deze pagina is voor administrator. Om deze pagina te bekijken moet je inloggen als administrator. Alleen gebruikers met een **ROLE_ADMIN** hebben toegang.

Voor alle andere pagina's is geen login nodig.

/WebSecurityConfig

De WebSecurityConfig class wordt gebruikt om de beveiliging van de applicatie te configureren door middel van @Configuration. Deze annotatie geeft aan dat het om een configuratie gaat en wordt ingelezen op het moment dat de applicatie draait.

/Remember me

De "Remember me" functie onthoudt de login van de gebruiker om automatisch in te kunnen loggen wanneer hij of zij de volgende keer de website bezoekt. Wanneer de gebruiker inlogt wordt de login informatie in de database opgeslagen als een token.

/DOA

DAO (Data Access Object) classes worden gebruikt om toegang te krijgen tot de database. De DAO-klassen worden gecheckt door @Repository zodat ze als BEANS kunnen worden beheerd.

/UserDetailsService

UserDetailsService is een service die kijkt naar het gebruikersaccount en gebruikersrollen. Het wordt door Spring Security gebruikt wanneer gebruikers inloggen in het systeem.

6.4 Chat functie

Ik heb een eenvoudige Chat applicatie gemaakt met Spring Boot en WebSocket. STOMP (Streaming Text Oriented Messaging Protocol) is een onderdeel van WebSocket. Wanneer de client en de server via dit protocol contact met elkaar opnemen, kunnen ze elkaar berichten sturen.

Hiervoor heb ik de volgende files aangemaakt

- Config: websocketConfig
- Interceptor: httpHandshakeEventListener
- Listener: websocketEventListener
- Model: ChatMessage
- ChatController
- WebSocketController

- Chat.html
- Chatlogin.html
- Chat.js
- Chat.css

MessageBroker

MessageBroker is een tussenprogramma dat berichten ontvangt die voor de distributie naar de benodigde adressen worden verzonden. De MessageBroker legt het eindpunt bloot zodat de klant contact kan opnemen en een verbinding kan vormen. Om contact op te nemen gebruikt de klant de:

```
var socket = nieuwe SockJS('/ws');  
stompClient = Stomp.over(socket);  
stompClient.connect({}, onConnected, onError);
```

SockJS

Niet alle browsers ondersteunen het WebSocket-protocol. Daarom is de SockJS (een JavaScript-bibliotheek) een terugvaloptie, die wordt geactiveerd voor de browsers die de WebSocket niet ondersteunen.

@EnableWebSocketMessageBroker

Deze annotatie meldt: "laten we WebSocket Server inschakelen".

HTTP-handshake

Om gemakkelijk te kunnen implementeren, gebruikt de WebSocket HTTP Handshake om te upgraden. Dat betekent dat de client een HTTP-gebaseerd verzoek naar de server stuurt, waarbij hij de server vertelt dat het geen HTTP is, maar upgradet naar de WebSocket, en zo vormen ze een verbinding.

6.5 CRUD RESTFul API

De CRUD RESTFul API is een User Management Systeem, waarmee je gebruikers kunt toevoegen, bekijken, updaten en deleten.

1. Create AppUser (JSON respons)

HTTP Method: POST

Request URL: <http://localhost:8080/api/v1/appUsers>

2. Get AppUser by ID

HTTP Method: GET

Request URL: <http://localhost:8080/api/v1/appUsers/3>

3. Get all AppUses

HTTP Method: GET

Request URL: <http://localhost:8080/api/v1/appUsers>

4. Update AppUser

HTTP Method: GET

Request URL: <http://localhost:8080/api/v1/appUsers/3>

5. Delete AppUser

HTTP Method: DELETE

Request URL: <http://localhost:8080/api/v1/appUsers/3>

Gebruikte bestanden:

- Repository : UserRepository
- Exeption
 - ErrorDetails
 - GlobalExceptionHandler
 - ResourceNotFoundException
- Model AppUser
- UserController

- UserTable.css
- UserTable.js

Exeptions

Door middel van Exception (de @ResponseStatus annotatie) wordt de status van uitzonderingen gespecificeerd (bv user not found). Om ErrorDetails te gebruiken om de error respons te retourneren, is een GlobalExceptionHandler class aangemaakt met een @ControllerAdvice annotatie. Deze class bekijkt specifieke en globale uitzonderingen.