Full STEK je moestuin met Node.js, Stencil en BLE

Door middel van het volgen van de onderstaande stappen zal je straks een virtuele moestuin omgeving creëren in je browser.

Systeemvereisten:

Voordat je begint aan het werken aan het maken van je moestuin moet je ervoor zorgen dat de applicatie op jouw machine kan draaien. Je moet in ieder geval Node versie 10+ en Git geïnstalleerd hebben.

Windows

Heb je een Windows machine dan is het van belang dat je de Windows Build Tools en Python 2.7 geïnstalleerd hebt. Deze kun je installeren door het volgende commando uit te voeren in je terminal:

```
npm install --global --production windows-build-tools --vs2015
```

MacOS

Wanneer je een Mac machine hebt dan is het van belang dat je Xcode command line tools geïnstalleerd hebt. Dit kun je doen door het volgende commando uit te voeren in je terminal:

```
xcode-select --install
```

Ook dien je Python 2.7 geïnstalleerd te hebben. In veel gevallen staat deze al op de Mac, anders moet je het zelf even installeren via de Python website. Of je Python 2.7 al op je Mac hebt staan kun je controleren met het volgende commando:

```
python -V
```

Installeren dependencies:

Wanneer je je machine ingericht hebt gaan we verder met de volgende stap. Als je beide projecten opent, open je voor beide projecten de terminal en vul het volgende in:

```
npm install
```

Indien je IntelliJ gebruikt moet je nog even de juiste versie van JavaScript instellen. Dit doe je op de volgende manier: IntelliJ ► Preferences ► Languages & Frameworks ► JavaScript ► ECMASCRIPT 6. Vervolgens klik je op ‘OK’ om de wijzigingen op te slaan.

Nu je bovenstaande stappen doorlopen hebt is het tijd om te beginnen met de opdracht. Deze is te vinden op de volgende pagina, succes!

Opdracht:

Als eerste stap zullen we de constructor van *DeviceData* moeten aanmaken, plus een functie die de attributen van *DeviceData* wegschrijft in JSON-formaat.

1) Voeg de volgende code toe aan device-data.js in het node project:

```
constructor(deviceId, temperature, lux, moisture, fertility) {
        this.deviceId = deviceId;
        this.temperature = temperature;
        this.lux = lux;
        this.moisture = moisture;
        this.fertility = fertility;
}
getJSON() {
     return {
          deviceId: this.deviceId,
          temperature: this.temperature,
          lux: this.lux,
          moisture: this.moisture,
          fertility: this.fertility
     }
}
```

Nu dat we het object kunnen aanmaken met de bijbehorende attributen, kunnen we deze data doorsturen door middel van een *EventEmitter* bij de *parseData* functie toe te voegen.

2) Voeg de volgende code toe aan de *parseData* functie in miflora.js in het node project:

```
this.emit('data', deviceData);
```

De *emit* methode zal luisteren naar het *'data'* event. Als je vervolgens naar rest-server.js gaat zal je ook zien dat bij de *flora.on* parameters het *'data'* event wordt aangeroepen. De data die daaruit komt is dan ook de opgehaalde data van de sensoren in de buurt.
Maar voordat je data kan ophalen uit de sensor zullen we eerst moeten scannen naar alle beschikbare sensoren in de omgeving.

3) Probeer om in rest-server.js in het node project, met de *flora* variabele de *startScanning* methode aan te roepen om ervoor te zorgen dat als je de rest-server start, de server zal gaan scannen naar alle sensoren in de buurt.

Als je deze 3 stappen correct hebt uitgevoerd kan je als je *"node rest-server.js"* aanroept in je terminal, de data zien van alle sensoren in je omgeving.

Het resultaat in de terminal zal er ongeveer zo uit moeten zien:

```
Received device information from id: ab14d94dc3f6431489c027493d18a207
Received data: {"deviceId":"ab14d94dc3f6431489c027493d18a207","temperature":20.5,"lux":193,"moisture":0,"fertility":0}
Received device information from id: b208223a717044ba9e97f112f83c4e2d
Received data: {"deviceId":"b208223a717044ba9e97f112f83c4e2d","temperature":20.8,"lux":211,"moisture":0,"fertility":0}
Received device information from id: 0a917b01e9474f00844d3ba719c20b15
Received data: {"deviceId":"0a917b01e9474f00844d3ba719c20b15","temperature":21.2,"lux":55,"moisture":0,"fertility":0}
Received device information from id: 153afa7a8f084cc686c71f1440c914b8
Received data: {"deviceId":"153afa7a8f084cc686c71f1440c914b8","temperature":20.6,"lux":148,"moisture":0,"fertility":0}
Received device information from id: b208223a717044ba9e97f112f83c4e2d
Received data: {"deviceId":"b208223a717044ba9e97f112f83c4e2d","temperature":20.7,"lux":194,"moisture":0,"fertility":0}
Received device information from id: ab14d94dc3f6431489c027493d18a207
Received data: {"deviceId":"ab14d94dc3f6431489c027493d18a207","temperature":20.5,"lux":193,"moisture":0,"fertility":0}
Received device information from id: 850d282782c54f7386510ecd97d4ba94
Received data: {"deviceId":"850d282782c54f7386510ecd97d4ba94","temperature":20.9,"lux":312,"moisture":0,"fertility":0}
```

Op Windows gaat dit waarschijnlijk niet in één keer goed en krijg je de foutmelding dat er geen compatible Bluetooth 4.0 device is gevonden. Hoe je Windows gerelateerde issues oplost kun je vinden in het readme_windows.pdf bestand.

De volgende stap is om in plaats van de data in de terminal te laten zien, de data door te sturen naar de browser.

Nu hebben wij al een aantal dingen voor je gedaan om de server aan te maken, er moeten alleen nog 2 dingen gedaan worden.

4) Ga naar rest-server.js in het node project en probeer de server te laten luisteren op port 8888.
*****Gebruik daarvoor de listen methode*****

Nu dat je de server hebt laten luisteren op de juiste port, moeten we alleen nog de data van de sensoren doorsturen naar de juiste route. We hebben een array aangemaakt waar alle sensoren die in de buurt zijn aan worden toegevoegd en worden geüpdatet genaamd *floraDevices*. *floraDevices* wordt dan ook in een JSON-formaat naar de browser gestuurd.

5) Voeg de volgende code toe aan rest-server.js in het node project:

```
app.get('/devices', (req, res) => {
    res.setHeader("Access-Control-Allow-Origin", "*");
    res.setHeader("Access-Control-Allow-Credentials", "true");
    res.json(floraDevices);
});
```

Als je deze 2 stappen ook correct uitgevoerd hebt, *"node rest-server.js"* nogmaals aanroept in je terminal en vervolgens naar http://localhost:8888/devices gaat. Dan kan je opnieuw de data van alle sensoren zien, maar nu in je browser!

Het resultaat in de browser zal er ongeveer zo uit moeten zien:

[{"deviceId":"b208223a717044ba9e97f112f83c4e2d","temperature":20.7,"lux":197,"moisture":0,"fertility":0},
{"deviceId":"153afa7a8f084cc686c71f1440c914b8","temperature":20.6,"lux":147,"moisture":0,"fertility":0},
{"deviceId":"ab14d94dc3f6431489c027493d18a207","temperature":20.6,"lux":179,"moisture":0,"fertility":0},
{"deviceId":"0a917b01e9474f00844d3ba719c20b15","temperature":21.4,"lux":56,"moisture":0,"fertility":0},
{"deviceId":"850d282782c54f7386510ecd97d4ba94","temperature":21,"lux":82,"moisture":0,"fertility":0}]

Nu de backend bijna volledig functioneel is, gaan we proberen om de data in de frontend te krijgen om vervolgens je virtuele moestuin vorm te geven!

Wij gebruiken voor het ophalen van de data in de frontend socket.io, maar omdat socket.io nog niet in ons project zit zal je het volgende moeten doen.

6) Importeer socket.io-client in app-home.tsx in het web project.

Nu we in de front- en backend socket.io hebben geïmporteerd, zijn we klaar om het te kunnen implementeren in onze webapplicatie. Om de data van de backend naar de frontend te sturen via socket.io zal je eerst socket.io moeten laten luisteren naar je server.

7) Laat socket.io luisteren naar de server in rest-server.js in het node project.

Als je het goed hebt gedaan luistert socket.io nu naar de server in de backend. De volgende stap is om de data mee te sturen met socket.io naar de frontend. Dit werkt vrijwel hetzelfde als de 2e stap, want ook hier zal je gebruik maken van de *emit* methode.

8) Voeg de volgende code toe aan de *updateDeviceData* functie in rest-server.js in het node project:

```
io.local.emit('data', floraDevices)
```

Omdat nu alles werkend is in de backend, start de server nog een keer opnieuw op door middel van *"node rest-server.js"* aan te roepen in je terminal.

Er moet nu verbinding worden gemaakt met de backend vanuit de frontend met socket.io om de data binnen te halen in de frontend.

9) Voeg de volgende code toe aan app-home.tsx in het web project:

```
  @Prop()
  socketConnection: any = io.connect("http://localhost:8888");
```

Door middel van deze code zal er verbinding worden gemaakt met de backend, zodat je de data van de sensoren binnen kan krijgen in je frontend.

We hebben nu verbinding gemaakt met socket.io in de backend, dus nu is het tijd om de data binnen te halen in de frontend.

10) Maak nu een socket.io event in de *updateFlowerData* functie waarbij je het data event aanroept en vervolgens de opgehaalde data toekend aan this.flowers in app-home.tsx in het web project.

Nu we eindelijk de data binnen halen moet de data alleen nog met de web componenten mee worden gegeven als het ingeladen wordt.

11) Roep de *updateFlowerData* methode aan in *componentWillLoad* in app-home.tsx in het web project.

Om vervolgens nog data met de bloemetjes mee te laden zal je nog een methode moeten maken met de **@Watch()** decorator. Wanneer er nieuwe sensor data binnenkomt via socket.io ziet de **@Watch()** decorator dat en vervangt hij de huidige data met de nieuwe data.

12) Voeg de volgende code toe aan app-flower.tsx in het web project:

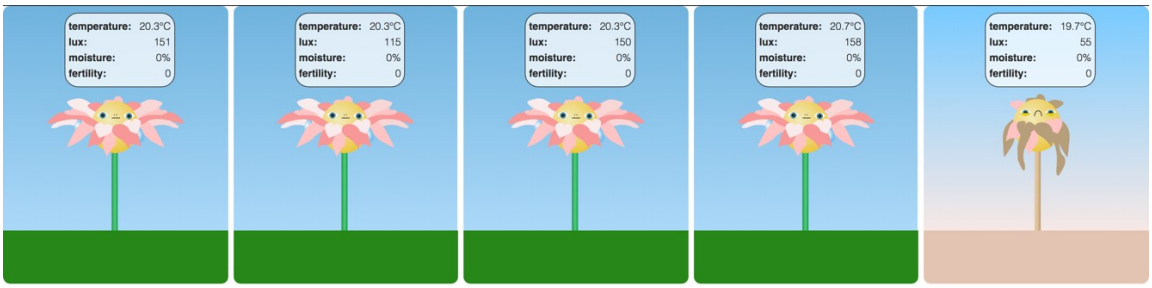
```
@Watch('flower')
    onFlowerChanged(value: IDeviceValues) {
        this.deviceId = value.deviceId;
        this.temperature = value.temperature;
        this.lux = value.lux;
        this.moisture = value.moisture;
        this.fertility = value.fertility;this.calculateFlowerState();
    }
```

Om de virtuele moestuin omgeving compleet te maken staat er nog maar 1 ding te doen.

13) Roep *onFlowerChanged* aan in componentWilload met als parameter *this.flower* in app-flower.tsx in het web project.

Als je alle stappen correct hebt doorlopen, gefeliciteerd! De virtuele moestuin werkt! Je zal nu verschillende bloemetjes zien. Het bloemetje zijn humeur is nu gebaseerd op het aantal licht dat hij krijgt.

Als je nu in je terminal in de frontend *"npm start"* aanroept zal het resultaat er in de browser er ongeveer zo uit moeten zien:



BONUS!
Probeer tot slot eens aan de gang te gaan met *calculateFlowerState* om te kijken of je gebaseerd op andere variabele het bloemetje zijn humeur kan laten veranderen.