

# Opdracht 5

---

## Cloud Technology & Security

Qing Scholten

---

### Opdracht 1: Function met mqtt publish

1.

```
const { app } = require('@azure/functions');

app.http('MQTTPublish', {
  methods: ['POST'],
  authLevel: 'anonymous',
  handler: async (request, context) => {
    const mqtt = require("mqtt");
    const client = mqtt.connect("mqtt:broker.mqttdashboard.com");
    var publishtopic = "Alarmlicht";

    context.log(`Http function processed request for url "${request.url}"`);

    let text = "Test";
    client.publish(publishtopic, text);

    return { body: `Publish is verzonden: ${text}!` };
  }
});
```

POST http://localhost:7071/a ● | +

 http://localhost:7071/api/MQTTPublish

POST  http://localhost:7071/api/MQTTPublish


Params   Authorization   Headers (8)   Body   Scripts   Settings

Query Params

	Key
	Key

Body   Cookies   Headers (4)   Test Results

Pretty   Raw   Preview   Visualize

Text 



1 Publish is verzonden: Test!

Publish

Topic

testtopic/1

QoS

0

Retain

☐

Publish

Message

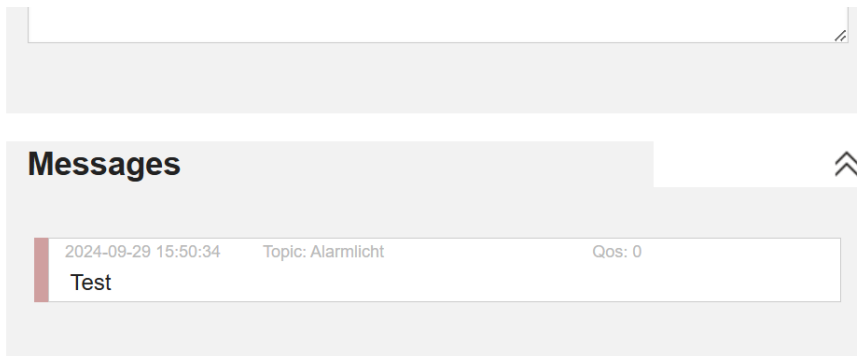
Subscriptions

Add New Topic Subscription

Qos: 2

Alarmlicht

x



2.

```
app.http('AlarmAansturing', {
  methods: ['PUT'],
  authlevel: 'anonymous',
  route: 'alarm/{id:int?}',
  handler: async (request, context) => {
    const mqtt = require("mqtt");
    const client = mqtt.connect("mqtt:broker.mqttdashboard.com");
    var publishtopic = "Alarmlicht";

    context.log(`Http function processed request for url "${request.url}"`);

    try {
      var id = request.params.id;
      if (id == 1) {
        var bod = await request.json();
        var text = JSON.stringify(bod);
        client.publish(publishtopic, text);
        return {
          status: 200
        }
      }
      else {
        return {
          status: 404,
          body: `Alarm met ID ${id} bestaat niet.`
        }
      }
    }
    catch (e) {
      context.log(`Fout bij het aansturen van het alarm`, e);
      return {
        status: 500,
        body: 'Interne serverfout bij aansturen van alarm.'
      }
    }
  }
});
```

C:\Windows\System32\cmd.exe X Windows PowerShell X + v

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows  
  
PS C:\Users\Kwik> cd '.\Documents\HBO-ICT\IOT\Cloud Technology & Security\  
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> node .\Alarm.js  
aan|

PUT http://localhost:7071/ap + No environment

http://localhost:7071/api/alarm/1 Save Share

PUT http://localhost:7071/api/alarm/1 Send

Params Authorization Headers (9) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON v Beautify

1 {"rood":255, "groen": 0, "blauw": 0}

Body Cookies Headers (3) Test Results 200 OK 272 ms 92 B Save Response

Pretty Raw Preview Visualize Text 1

The screenshot shows a Windows PowerShell terminal window and a REST client interface. In the PowerShell terminal, the user navigates to the directory `C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security\` and runs `node .\Alarm.js`. The terminal output shows a message: `Received message on topic Alarmlicht: {"rood":255,"groen":0,"blauw":0}`. Below the terminal, the REST client shows a `PUT` request to `http://localhost:7071/api/alarms/2` with a JSON body: `{"rood":255, "groen": 0, "blauw": 0}`. The response is a `404 Not Found` status with the message: `Alarm met ID 2 bestaat niet.`

3.

```

app.http('AlarmAansturing', {
  methods: ['PUT'],
  authlevel: 'anonymous',
  route: 'alarm/{id:int?}',
  handler: async (request, context) => {
    const mqtt = require("mqtt");
    const client = mqtt.connect("mqtt:broker.mqttdashboard.com");
    const alarmlichten = ["Alarmlicht", "Alarmlicht2"]
    var publishtopic = "Alarmlicht";

    context.log(`Http function processed request for url "${request.url}"`);

    try {
      var id = request.params.id - 1;
      if (id < alarmlichten.length) {
        var bod = await request.json();
        var text = JSON.stringify(bod);
        client.publish(alarmlichten[id], text);
        return {
          status: 200
        }
      }
    }
  }
})

```

```
        else {
            return {
                status: 404,
                body: `Alarm met ID ${id} bestaat niet.`
            }
        }
    }
}
catch (e) {
    context.log(`Fout bij het aansturen van het alarm`, e);
    return {
        status: 500,
        body: 'Interne serverfout bij aansturen van alarm.'
    }
}
});
```

PUT http://localhost:7071/api/alarms/1

PUT http://localhost:7071/api/alarms/1

Send

Params Authorization Headers (9) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

1 { "rood": 0, "groen": 0, "blauw": 255 }

Body Cookies Headers (3) Test Results

200 OK • 12 ms • 92 B • Save Response

Pretty Raw Preview Visualize Text

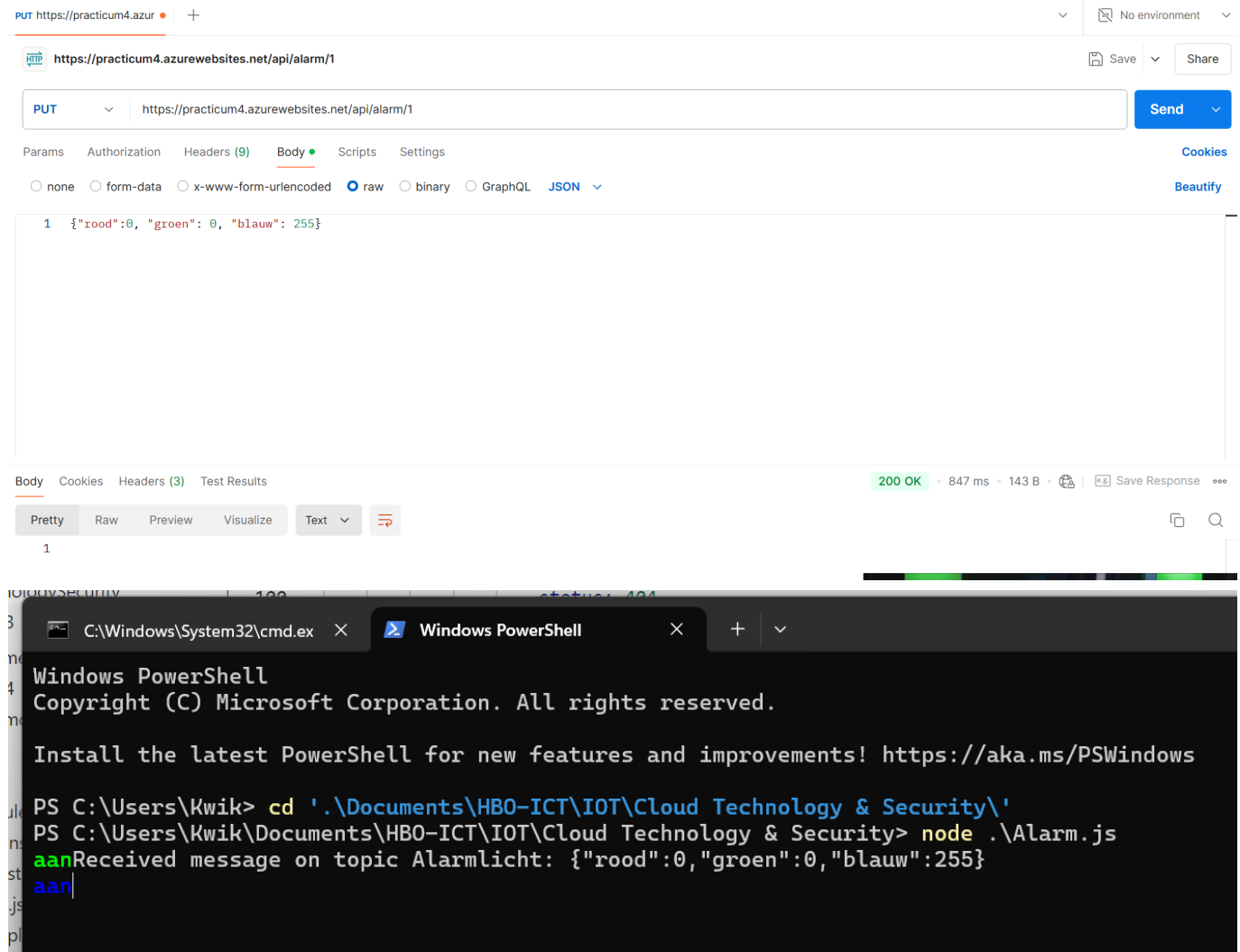
1

```
C:\Windows\System32\cmd.exe × Windows PowerShell × + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Kwik> cd '.\Documents\HBO-ICT\IOT\Cloud Technology & Security\'
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> node .\Alarm.js
Received message on topic Alarmlicht: {"rood":0,"groen":0,"blauw":255}
aan
```

4.

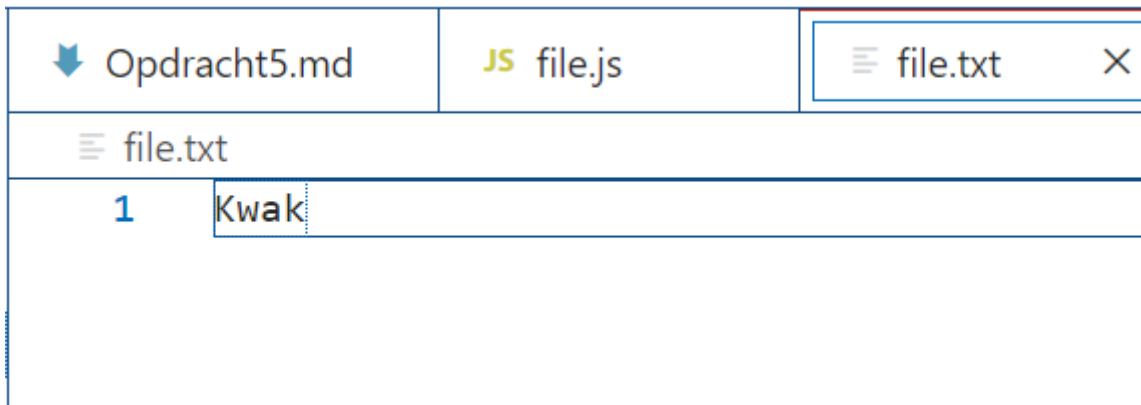


The screenshot shows a web browser with a PUT request to `https://practicum4.azurewebsites.net/api/alarm/1`. The request body is a JSON object: `{ "rood": 0, "groen": 0, "blauw": 255 }`. The response status is 200 OK. Below the browser, a Windows PowerShell terminal window is open, showing the command `node .\Alarm.js` being executed. The output of the command is: `aanReceived message on topic Alarmlicht: {"rood":0,"groen":0,"blauw":255}`.

## Opdracht 2: Await en Async in node.js

5.

```
var fs = require ('fs');
fs.readFile('file.txt', function(e, data) {
  if (e) {
    console.log(`Error bij het lezen van bestand: ${e}`);
    return;
  }
  else {
    console.log(`Bestandsinhoud: ${data}`);
    return;
  }
});
console.log("Hey Ho");
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Kwik> cd '.\Documents\HBO-ICT\IOT\Cloud Technology & Security\'
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> node .\file.js
Hey Ho
Bestandsinhoud: Kwak
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> |
```

6.

```
async function ReadFile() {
  var fs = require('fs/promises');
  try {
    const data = await fs.readFile('file.txt');
    console.log(`Bestandsinhoud: ${data}`);
  }
  catch (e) {
    console.log(`Error bij het lezen van bestand: ${e}`);
  }
}

ReadFile();
console.log("Hey Ho")
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Kwik> cd '.\Documents\HBO-ICT\IOT\Cloud Technology & Security\'
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> node .\file.js
Hey Ho
Bestandsinhoud: Kwak
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> |
```

7.



```
app.http('LeesBestand', {
  methods: ['GET'],
  authlevel: 'anonymous',
  route: 'leesbestand',
  handler: async (request, context) => {
    async function ReadFile() {
      var fs = require ('fs/promises');
      try {
        const data = await fs.readFile('package.json');
        console.log(`Bestandsinhoud: ${data}`);
      }
      catch (e) {
        console.log(`Error bij het lezen van bestand: ${e}`);
      }
    }

    ReadFile();
    console.log("Hey Ho")
  }
})
```

```
2024-09-30T07:34:37.490 [Information] Hey Ho
2024-09-30T07:34:37.490 [Information] Executed 'Functions.LeesBestand' (Succeeded, Id=f54de6a4-7df2-4e0d-9732-bac39c218b9d, Duration=3ms)
2024-09-30T07:34:37.491 [Information] Bestandsinhoud: {
2024-09-30T07:34:37.491 [Information]   "name": "",
2024-09-30T07:34:37.491 [Information]   "version": "1.0.0",
2024-09-30T07:34:37.491 [Information]   "description": "",
2024-09-30T07:34:37.491 [Information]   "main": "src/functions/*.js",
2024-09-30T07:34:37.491 [Information]   "scripts": {
2024-09-30T07:34:37.491 [Information]     "start": "func start",
2024-09-30T07:34:37.491 [Information]     "test": "echo \\\"No tests yet...\\\""}
2024-09-30T07:34:37.491 [Information] },
2024-09-30T07:34:37.491 [Information]   "dependencies": {
2024-09-30T07:34:37.491 [Information]     "@azure/functions": "^4.0.0",
2024-09-30T07:34:37.491 [Information]     "chalk": "^4.1.2",
2024-09-30T07:34:37.491 [Information]     "mqt": "^5.10.1"
2024-09-30T07:34:37.491 [Information]   },
2024-09-30T07:34:37.491 [Information]   "devDependencies": {
2024-09-30T07:34:37.491 [Information]     "azure-functions-core-tools": "^4.x"
2024-09-30T07:34:37.491 [Information]   }
2024-09-30T07:34:37.487 [Information] Executing 'Functions.LeesBestand' (Reason='This function was programmatically call
ed via the host APIs.', Id=f54de6a4-7df2-4e0d-9732-bac39c218b9d)
2024-09-30T07:34:37.490 [Information] Executed 'Functions.LeesBestand' (Succeeded, Id=f54de6a4-7df2-4e0d-9732-bac39c218b9d, Duration=3ms)
```

GEThttps://practicum4.azure

No environment

https://practicum4.azurewebsites.net/api/leesbestand

SaveShare

GEThttps://practicum4.azurewebsites.net/api/leesbestandSend

ParamsAuthorizationHeaders (7)BodyScriptsSettingsCookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

BodyCookiesHeaders (2)Test Results204 No Content31 ms132 BSave Response

PrettyRawPreviewVisualizeText

1

Opdracht 3: Azure lot hub en functio

8.

```
app.http('AlarmIOT', {
  methods: ['PUT'],
  authLevel: 'anonymous',
  route: 'alarm2/{id:int?}',
  handler: async (request, context) => {
    var Client = require('azure-iot-hub').Client;
    var connectionString = process.env.IOTHUB_CONNECTION_STRING;
    var client = Client.fromConnectionString(connectionString);
    context.log(`Http function processed request for url "${request.url}"`);

    try {
      var id = request.params.id;
      if (id == 1) {
        var bod = await request.json();
```

```
var text = JSON.stringify(bod);
text = JSON.parse(text);
var methodParams = {
  methodName: text.method,
  payload: text.payload,
  responseTimeoutInSeconds: 15
}
var targetDevice = "NewAlarm";
client.invokeDeviceMethod(targetDevice, methodParams, function
(err, result) {
  if (err) {
    console.error('Failed to invoke method \'' +
methodParams.methodName + '\': ' + err.message);
  }
  else {
    console.log(methodParams.methodName + ' on ' +
targetDevice + ':');
    console.log(JSON.stringify(result, null, 2));
  }
})
return {
  status: 200
}
}
else {
  return {
    status: 404,
    body: `Alarm met ID ${id} bestaat niet.`
  }
}
}
catch (e) {
  context.log(`Fout bij het aansturen van het alarm`, e);
  return {
    status: 500,
    body: 'Interne serverfout bij aansturen van alarm.'
  }
}
});
```

The screenshot shows a REST client interface. The URL bar displays `http://localhost:7071/api/alarm2/1`. The method is set to `PUT`. The body is a JSON object: `{ "method": "changeColour", "payload": { "red": 255, "green": 0, "blue": 0 } }`. The response status is `200 OK` with a response time of `937 ms` and a size of `92 B`. The response body is empty.

The screenshot shows a Windows PowerShell terminal window. The prompt is `PS C:\Users\Kwik>`. The user enters `cd '..\Documents\HBO-ICT\IOT\Cloud Technology & Security\'`. The prompt changes to `PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security>`. The user enters `node .\AlarmAzure.js`. The output shows: `Connected to device. Registering handlers for methods.`, `Received method call for method 'changeColour'`, `Payload: [object Object]`, and `Response to method 'changeColour' sent successfully.`. The user enters `aan`.

9.

```
var alarms = {
  "Alarms": [
    {
      "Method": "MQTT",
      "Name": "Alarmlicht"
    },
    {
      "Method": "Azure",
      "Name": "NewAlarm"
    }
  ]
}

app.http('AlarmAzureIOT', {
  methods: ['PUT'],
  authLevel: 'anonymous',
```

```

route: 'alarm/{id:int?}',
handler: async (request, context) => {
  var Client = require('azure-iot-hub').Client;
  var connectionString = process.env.IOTHUB_CONNECTION_STRING;
  var client = Client.fromConnectionString(connectionString);
  context.log(`Http function processed request for url "${request.url}"`);
  try {
    var id = request.params.id - 1;
    if (id < alarms.Alarms.length) {
      var bod = await request.json();
      var text = JSON.stringify(bod);
      text = JSON.parse(text);
      if (alarms.Alarms[id].Method === "MQTT"){
        const mqtt = require("mqtt");
        const clientmqtt =
mqtt.connect("mqtt:broker.mqttdashboard.com");
        var temptext = JSON.stringify(bod);
        clientmqtt.publish(alarms.Alarms[id].Name, temptext);
        return {
          status: 200
        }
      }
      else if (alarms.Alarms[id].Method === "Azure") {
        var methodParams = {
          methodName: "change",
          payload: text.payload,
          responseTimeoutInSeconds: 15
        }
        client.invokeDeviceMethod(alarms.Alarms[id].Name,
methodParams, function (err, result) {
          if (err) {
            console.error('Failed to invoke method \'' +
methodParams.methodName + '\': ' + err.message);
          }
          else {
            console.log(methodParams.methodName + ' on ' +
targetDevice + ':');
            console.log(JSON.stringify(result, null, 2));
          }
        })
      }
      else {
        return {
          status: 500,
          body: 'Interne serverfout bij aansturen van alarm.'
        }
      }
    }
    else {
      return {
        status: 404,
        body: `Alarm met ID ${id} bestaat niet.`
      }
    }
  }
}

```

```
    }  
    catch (e) {  
        context.log(`Fout bij het aansturen van het alarm`, e);  
        return {  
            status: 500,  
            body: 'Interne serverfout bij aansturen van alarm.'  
        }  
    }  
}  
});
```

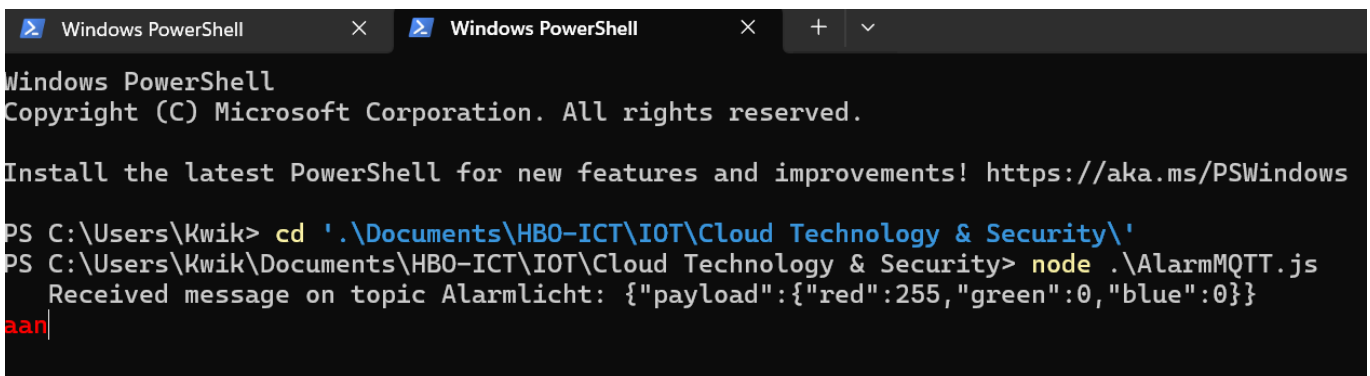
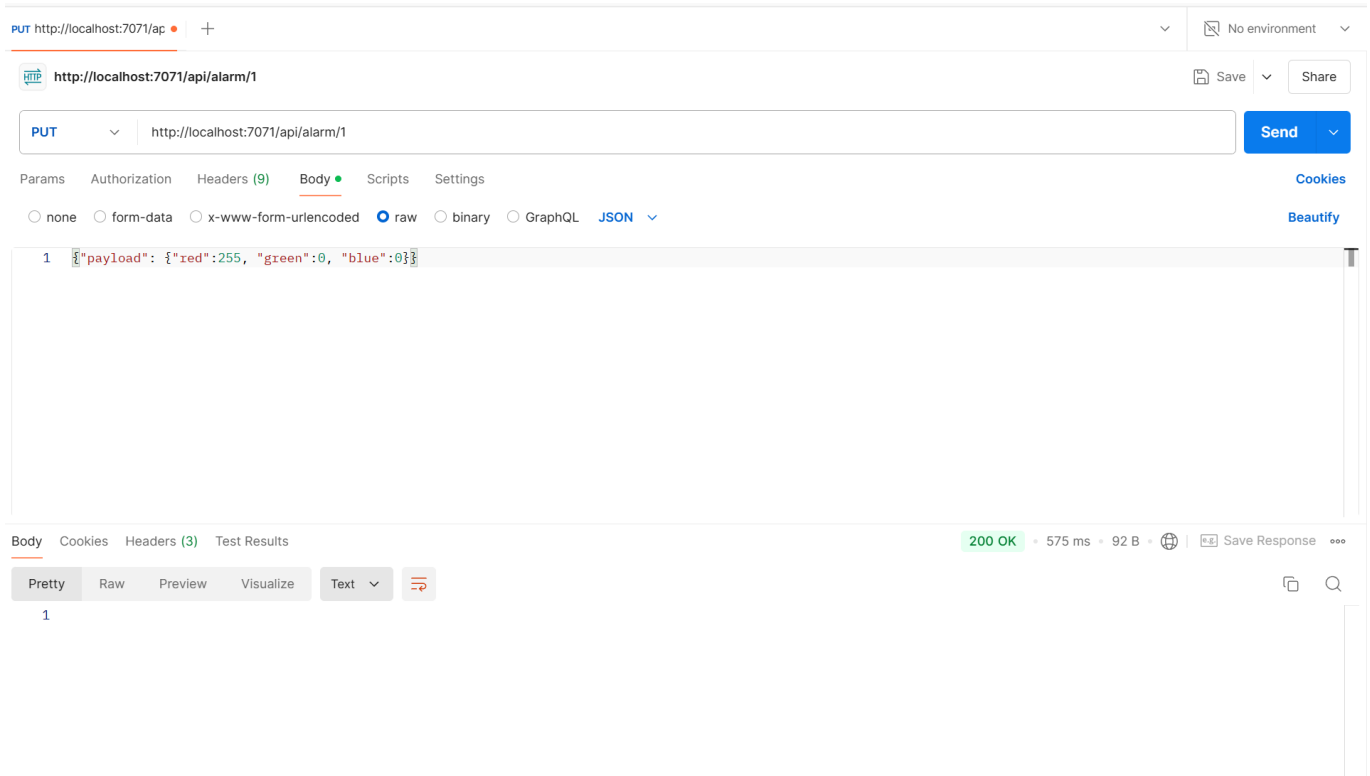
The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:7071/api/alarm/2
- Body:**

```
{ "payload": { "red": 255, "green": 0, "blue": 0 } }
```
- Response:** 204 No Content, 245 ms, 100 B
- Headers:** 3
- Test Results:** 1

The screenshot shows a Windows PowerShell terminal with the following commands and output:

```
PS C:\Users\Kwik> cd '.\Documents\HBO-ICT\IOT\Cloud Technology & Security\'  
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> node .\AlarmAzure.js  
Connected to device. Registering handlers for methods.  
  
PS C:\Users\Kwik\Documents\HBO-ICT\IOT\Cloud Technology & Security> node .\AlarmAzure.js  
Connected to device. Registering handlers for methods.  
aanReceived method call for method 'change'  
Payload:  
[object Object]  
Response to method 'change' sent successfully.  
aan
```



10. Omdat je nu een universele aansturing hebt voor verschillende soorten systemen, in dit geval Alarm systemen. Nu is de client niet meer afhankelijk van de implementatie van de verschillende alarm systemen (MQTT aangestuurd en Azure aangestuurd).

11.

```
app.http('AlarmGET', {
  methods: ['GET'],
  authLevel: 'anonymous',
  route: 'alarm/{id:int?}',
  handler: async (request, context) => {
    context.log(`Http function processed request for url "${request.url}"`);

    var id = request.params.id - 1;
    var al = alarms.Alarms[id];
    if (isNaN(id)) {
      context.log(`Verzoek tot opvragen gehele alarmlijst.`);
      var textlist = JSON.stringify(alarms);
    }
    else if (id > alarms.Alarms.length || id < 0) {
      context.log(`Verzoek tot opvragen van niet bestaand alarm!`)
    }
  }
})
```

```

        return {
            status: 404,
            body: "404! Dit alarm bestaat niet!"
        }
    }
    else {
        context.log(`Verzoek tot opvragen van alarm ${id}.`)
        var textlist = JSON.stringify(al);
    }
    return {
        status: 200,
        body: `${textlist}` };
    }
})

app.http('AlarmToevoegen', {
    methods: ['Post'],
    authLevel: 'anonymous',
    route: 'alarm',
    handler: async (request, context) => {
        context.log(`Http function processed request for url "${request.url}"`);
        try {
            var bod = await request.text();
            var nieuwalarm = JSON.parse(bod);
            if (!nieuwalarm.hasOwnProperty("Method")) {
                context.log("Poging tot toevoeging alarm zonder methode van
communicatie.");
                return {
                    status: 404,
                    body: "Een alarm kan niet toegevoegd worden zonder methode van
communicatie."
                }
            }
            else if (!nieuwalarm.hasOwnProperty("Name")) {
                context.log("Poging tot toevoeging alarm zonder naam.");
                return {
                    status: 404,
                    body: "Een alarm kan niet toegevoegd worden zonder naam."
                }
            }
            else {
                alarms.Alarms.push(nieuwalarm);
                context.log(`Alarm toegevoegd: ${JSON.stringify(nieuwalarm)}`);
                return {
                    status: 200,
                    body: `Alarm succesvol toegevoegd:
${JSON.stringify(nieuwalarm)}`
                }
            }
        }
        catch (e) {
            context.log(`Fout bij het verwerken van het toevoegen van een alarm`,
e);
            return {

```



```
status: 500,
body: 'Interne serverfout bij toevoegen alarm.'
}
}
}
});
```

GET http://localhost:7071/ap

POST http://localhost:7071/a

+

http://localhost:7071/api/alarm/

Save

Share

GET

http://localhost:7071/api/alarm/

Send

Params

Authorization

Headers (7)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

Body

Cookies

Headers (4)

Test Results

200 OK

410 ms

228 B

Save Response

...

Pretty

Raw

Preview

Visualize

Text

1

{ "Alarms": [ { "Method": "MQTT", "Name": "Alarmlicht" }, { "Method": "Azure", "Name": "NewAlarm" } ] }

GET http://localhost:7071/ap

POST http://localhost:7071/a

+

http://localhost:7071/api/alarm/

Save

Share

POST

http://localhost:7071/api/alarm/

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{ "Method": "MQTT", "Name": "NieuwAlarm" }

Body

Cookies

Headers (4)

Test Results

200 OK

30 ms

206 B

Save Response

...

Pretty

Raw

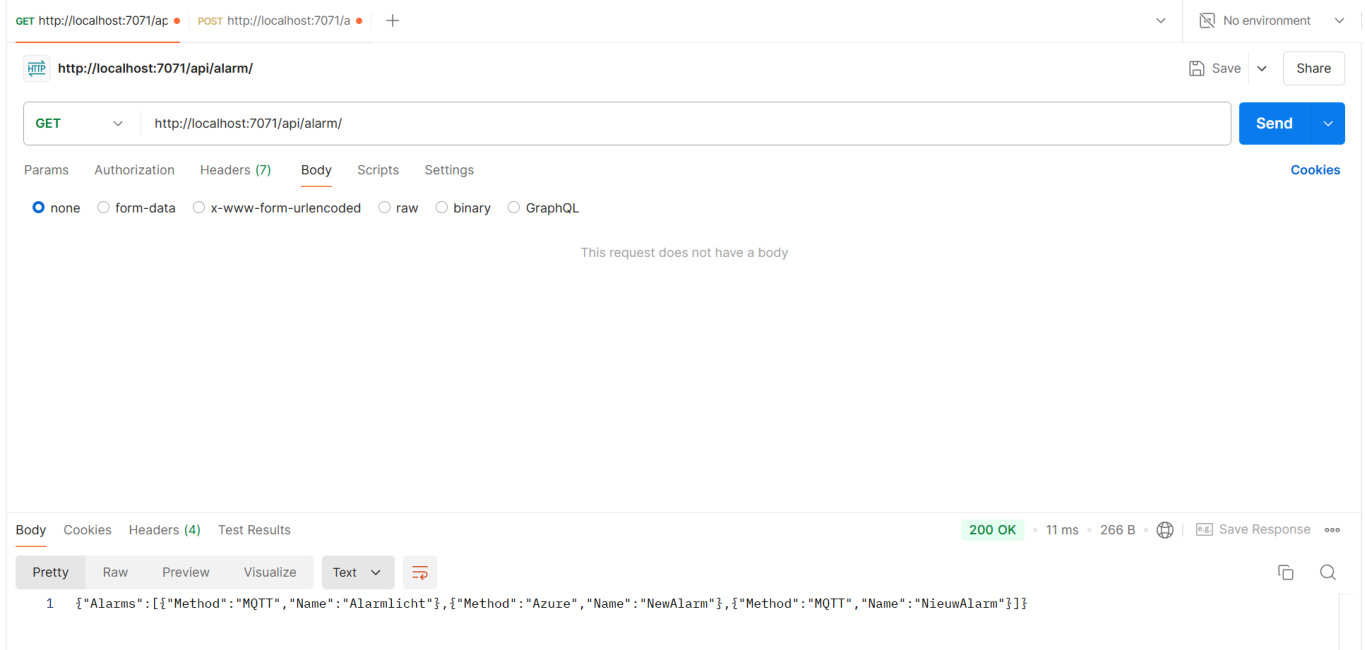
Preview

Visualize

Text

1

Alarm succesvol toegevoegd: { "Method": "MQTT", "Name": "NieuwAlarm" }



12.

```
app.http('DeurAansturing', {
  methods: ['PUT'],
  authLevel: 'anonymous',
  route: 'deur/{id:int?}',
  handler: async (request, context) => {
    var Client = require('azure-iot-hub').Client;
    var connectionString = process.env.IOTHUB_CONNECTION_STRING;
    var client = Client.fromConnectionString(connectionString);
    context.log(`Http function processed request for url "${request.url}"`);
    try {
      var id = request.params.id - 1;
      if (id < deuren.Deuren.length) {
        var bod = await request.json();
        var text = JSON.stringify(bod);
        text = JSON.parse(text);
        var methodParams = {
          methodName: "changeStatus",
          payload: text.payload,
          responseTimeoutInSeconds: 15
        }
        console.log(deuren.Deuren[id].Name);
        var result = await
        client.invokeDeviceMethod(deuren.Deuren[id].Name, methodParams);
        console.log(result.result);
        return {
          status: 200,
          body: JSON.stringify(result.result)
        }
      }
    }
    else {
      return {
        status: 404,

```

```
        body: `Deur met ID ${id+1} bestaat niet.`
    }
}
}
}
catch (e) {
    context.log(`Fout bij het aansturen van de deur`, e);
    return {
        status: 500,
        body: 'Interne serverfout bij aansturen van deur.'
    }
}
}
});
```

PUT http://localhost:7071/ap

No environment

http://localhost:7071/api/deur/1

Save Share

PUT http://localhost:7071/api/deur/1

Send

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {"payload": {  
2 | "status": "open"  
3 | }}

Body Cookies Headers (4) Test Results

200 OK 177 ms 184 B Save Response

Pretty Raw Preview Visualize Text

1 {"status":200,"payload":"Deur is nu open."}

PUT http://localhost:7071/ap

No environment

http://localhost:7071/api/deur/1

Save Share

PUT http://localhost:7071/api/deur/1

Send

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {"payload": {  
2 | "status": "open"  
3 | }}

Body Cookies Headers (4) Test Results

200 OK 183 ms 184 B Save Response

Pretty Raw Preview Visualize Text

1 {"status":200,"payload":"Deur is al open."}

PUT http://localhost:7071/ap

No environment

http://localhost:7071/api/deur/1

Save Share

PUT http://localhost:7071/api/deur/1

Send

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Cookies Beautify

1 {"payload": {  
2 | "status": "dicht"  
3 | }  
4 | }  
5 | }

Body Cookies Headers (4) Test Results

200 OK 135 ms 185 B Save Response

Pretty Raw Preview Visualize Text

1 {"status":200,"payload":"Deur is nu dicht."}

PUT http://localhost:7071/ap

No environment

http://localhost:7071/api/deur/1

Save Share

PUT http://localhost:7071/api/deur/1

Send

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Cookies Beautify

1 {"payload": {  
2 | "status": "dicht"  
3 | }  
4 | }  
5 | }

Body Cookies Headers (4) Test Results

200 OK 205 ms 185 B Save Response

Pretty Raw Preview Visualize Text

1 {"status":200,"payload":"Deur is al dicht."}