HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

Software Engineering Bachelor

Embedded Systems

Projektplan

Heizungssteuerung mit ZigBee

Eugen Schlecht, Florens Hückstädt, Florian Wohlgemuth, Patrick Wohlgemuth

Inhaltsverzeichnis

1 Einführung

Im Zuge der Vorlesung Embedded Systems des Studiengangs Software Engineering Bachelor wird ein Projekt im Bereich der Heimautomatisierung durchgeführt. Thema des Projekts ist die Konzeption und Erstellung einer Heizungssteuerung. Dabei wird an einem im Sommersemester 2015 durchgeführten Projekt angeknüpft.

2 Ausgangssituation

3 Zieldefinition

Gegenstand des Projekts ist die Weiterführung des über mehrere Semester bearbeitete Thema Hausautomatisierung mit ZigBee. Der Stand der Vorgruppe ist ein durchgeführter Proof-Of-Concept mit Entwicklung eines prototypischen Systems zur Steuerung eines Heizungsventils. Die Absicht der Weiterführung des Themas ist Verarbeitung der gewonnen Erkenntnisse und die Entwicklung eines einsetzbaren Hausautomatisierungssystems. Der Aspekt der Einsetzbarkeit ist dabei so definiert, dass das System Gütekriterien aktueller Produkte im Bereich der Hausautomatisierung entspricht. Als exemplarisches Endgerät, das über das System angesteuert werden kann, bleibt weiterhin das Heizungsventil im Fokus. Der Anspruch an dieses ist nun die Kapselung der Elektronik in ein geschlossenes Gehäuse, so dass ein Anschluss an einen Heizkörper möglich ist. Ein Überblick des Systems inform eines Use-Case-Diagramms ist in Abbildung 1 dargestellt. Im Folgendem werden die Teilziele bzw. die Gütekriterien des geplanten Systems erläutert.

3.1 Erweiterbarkeit

Die Erweiterbarkeit des Systems beinhaltet zum einen die Möglichkeit der Ansteuerung vielfältiger Endgeräte wie zum Beispiel Lichtschalter, Türöffner, Jalousien. Die Erweiterbarkeit in diesem Sinne erfordert generische Ansätze in den Bereichen der Kommunikation, Datenhaltung und Darstellung. Zum anderen wird Erweiterbarkeit verstanden als Möglichkeit das Softwaresystem ohne großen Aufwand um neue Anforderungen, wie zum Beispiel die Entwicklung von Benutzeroberflächen für verschiedene Endgeräte, zu ergänzen.

3.2 Benutzerfreundlichkeit

Die Bedienung, Installation und Erweiterung des Systems soll für den Benutzer möglichst einfach erlernbar sein. Die Benutzeroberflächen sollen dabei an Benutzer angepasst werden, die über keine internen Systemkenntnisse verfügen.

3.3 Sicherheit

Das System soll gegen unbefugten Zugriff von außen abgesichert sein. Komponenten der Hausautomatisierung dürfen nicht von Unberechtigten gesteuert werden können.

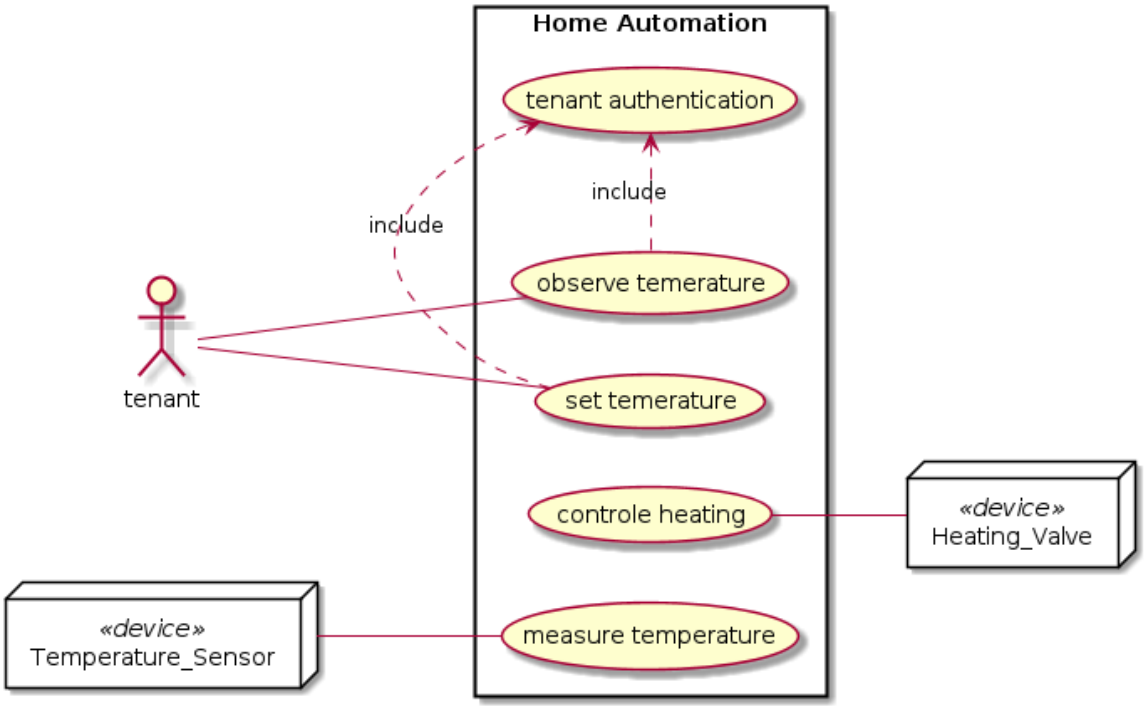

Abbildung 1: Use-Case-Diagramm Heizungssteuerung

4 Architektur und Umsetzung

Im Nachfolgenden werden die entwickelten und eingesetzten Lösungen zu den im Abschnitt 3 beschriebenen hardware- und softwarespezifischen Anforderungen erläutert. Abbildung 2 zeigt die grobe Systemarchitektur der entwickelten Lösung.

Das System besteht im wesentlichen aus vier Komponenten. Dem Endgerät - in unserem Szenarion das Heizungsventil, dem "Room Server", der das Endgerät über das ZigBee-Protokoll steuert, dem "Cloud Backend"welches ein RESTful API zur Verwaltung von Räumen und Geräten bereitstellt und einem Frontend zur Bedienung des Systems.

Die einzelnen Bestandteile des Systems werden in den folgenden Unterkapiteln genau beschrieben.

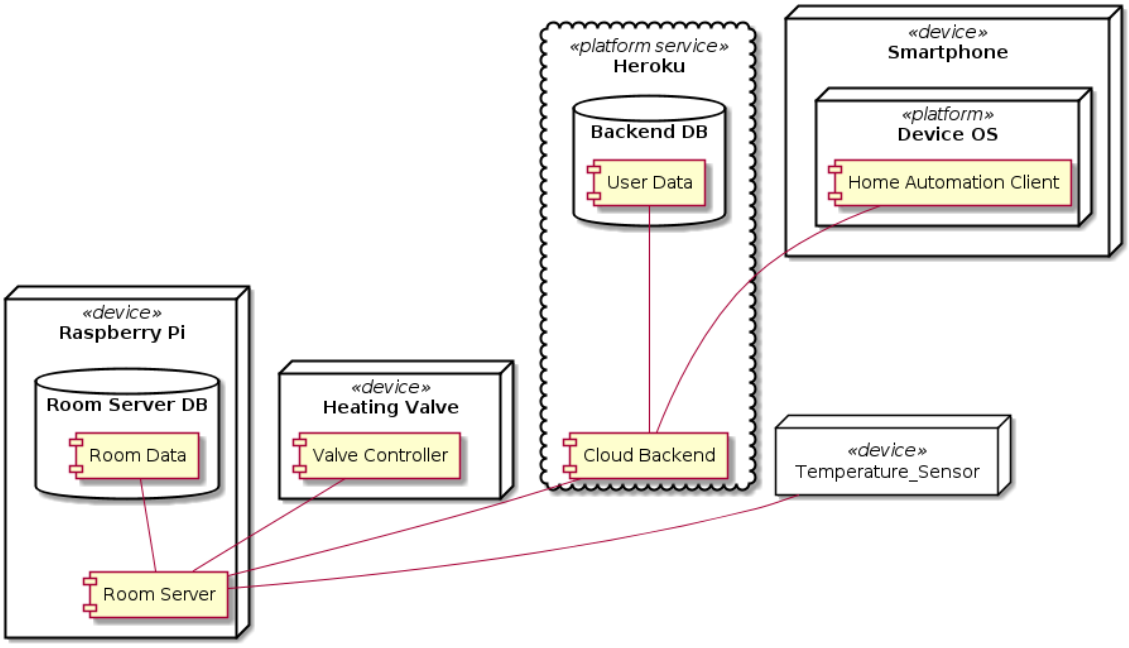

Abbildung 2: Sestemarchitektur

4.1 Cloud Backend

Das Cloud Backend ist eine in Java geschriebene Anwendung, die über das Internet erreichbar ist und ein RESTful API bietet. Über das API können Räume, Geräte und Benutzerdaten verwalten werden. Darunter fällt beispielsweise das Registrieren von Räumen und Geräten für einen bestimmten Benutzer. Darüberhinaus können Sollwerte der Geräte verändert werden. Die Veränderung wird dem tatsächlichen Gerät bzw. dem Room Server, der es verwaltet, über ein Messaging-Verfahren mitgeteilt (siehe dazu Unterunterabschnitt 4.1.2). Das API wird in Unterunterabschnitt 4.1.1 beschrieben. Der Vorgang des Deployments wird in Unterunterabschnitt 4.1.3 erläutert.

4.1.1 API

Das API ist als RESTful Web Service realisiert. Ressourcen können über HTTP mit den Methoden `GET`, `POST`, `PUT`, `PATCH` und `DELETE` angesprochen werden. Eine genaue Auflistung der API-Methoden ist im Anhang im Unterabschnitt A.1 zu finden.

Schema

Aufgrund des Klassenmodells der Domäne Hausautomatisierung, welches in Abbildung 3 dargestellt ist, ergeben sich API Endpoints mit folgender Struktur: `/user/rooms/{roomId}/devices/{deviceId}`.

Sämtlicher Zugriff auf das API erfolgt über HTTPS und kann über `https://enigmatic-waters-31128.herokuapp.com` erreicht werden.

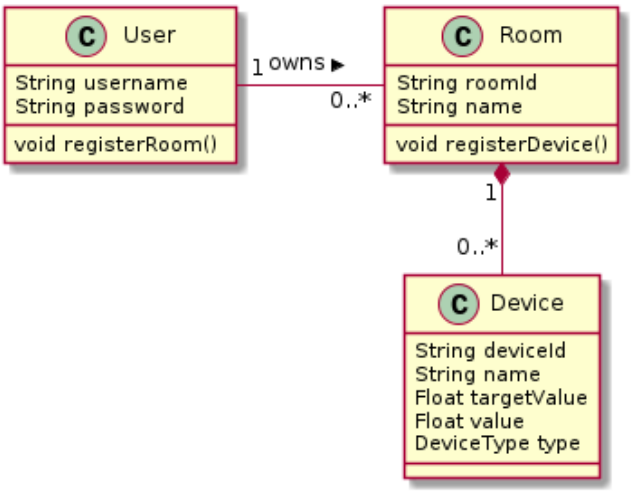

Abbildung 3: Klassendiagramm Hausautomatisierung

Eine beispielhafte Anfrage ist die Ausgabe des Raumes mit der Identifikation `d934eb20-4c6f-4d1c-91c5-61cdeddcf843`.
Dies kann mit einem GET-Request auf folgende URL erreicht werden:
`https://enigmatic-waters-31128.herokuapp.com/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843`.
Das Cloud Backend liefert auf diese Anfrage beispielhaft folgende Antwort:

```
HTTP/1.1 200 OK
Connection: keep-alive
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST, PUT, PATCH, GET, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: x-requested-with, content-type, X-AUTH-TOKEN
Access-Control-Expose-Headers: X-AUTH-TOKEN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sun, 01 May 2016 10:57:10 GMT
Via: 1.1 vegur

{
 "roomId":"d934eb20-4c6f-4d1c-91c5-61cdeddcf843",
 "name":"Wohnzimmer"
}
```

Sowohl beim Abfragen einer Liste als auch beim Abfragen einzelner Ressourcen beinhaltet die Antwort alle Attribute der Ressource.

Authentifizierung

Anfragen, die eine Authentifizierung erfordern, geben im Fehlerfall als Antwort `403 Forbidden` zurück. Zur Authentifizierung muss im Header das Feld `X-AUTH-TOKEN` gesetzt sein. Das entsprechende Token wird nach einer erfolgreichen Login-Anfrage erhalten.

Authentifizierung mit einem Token

```
$ curl 'https://enigmatic-waters-31128.herokuapp.com' -i -H 'X-AUTH-TOKEN: TOKEN'
```

Login

Zum Erhalt eines Tokens muss eine Loginanfrage gestellt werden. Diese enthält die Attribute username und password.

```
$ curl 'https://enigmatic-waters-31128.herokuapp.com/api/login' -i -X POST -H 'Content-Type: application/json
    ' -d '{"username":"foo", "password": "bar"}'

HTTP/1.1 200 OK
Connection: keep-alive
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST, PUT, PATCH, GET, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: x-requested-with, content-type, X-AUTH-TOKEN
Access-Control-Expose-Headers: X-AUTH-TOKEN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
X-Auth-Token: eyJpZCI6MTEsInVzZXJuYW1lIjoidXNlciIsImV4cGlyZXMiOjE0NjI5NjkONzIwMTgsInJvbGVzIjpbIlVTRVIiXX0=.
    LDd74GOyJAuQChYR9POAOmThfylOGGlY19u2DcTcXyQ=
Content-Length: 0
Date: Sun, 01 May 2016 12:24:32 GMT
Via: 1.1 vegur
```

Das Token aus dem Response-Header-Feld `X-AUTH-TOKEN` muss bei zukünftigen Anfragen enthalten sein.

Fehlgeschlagener Login

Loginanfragen mit ungültigen Benutzerdaten werden mit `401 Unauthorized` beantwortet.

```
$ curl 'https://enigmatic-waters-31128.herokuapp.com/api/login' -i -X POST -H 'Content-Type: application/json
    ' -d '{"username":"foo", "password": "bar"}'

HTTP/1.1 401 Unauthorized
Connection: keep-alive
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST, PUT, PATCH, GET, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: x-requested-with, content-type, X-AUTH-TOKEN
Access-Control-Expose-Headers: X-AUTH-TOKEN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sun, 01 May 2016 12:30:03 GMT
Via: 1.1 vegur

{
 "timestamp":1462105803298,
 "status":401,
 "error":"Unauthorized",
 "message":"Authentication Failed: Bad credentials",
 "path":"/api/login"
}
```

4.1.2 Messaging

Bla bla bla.

4.1.3 Build und Deployment

Zum Erstellen der Serverpaketierung kann Gradle verwendet werden. Da das Projekt den Gradle-Wrapper enthält muss Gradle nicht auf dem System installiert sein. Der Befehl zum Erstellen der Serverpaketierung lautet wie folgt:

```
gradlew build
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:findMainClass
:asciidoctor UP-TO-DATE
:jar
:bootRepackage
```

```
:assemble
:check
:build

BUILD SUCCESSFUL
```

Der Server kann dann als Java-Programm gestartet werden:

```
java -jar build/libs/backend-0.1.0.jar
```

Deployment auf Heroku

Zur Bereitstellung des Servers bei dem Cloud-Dienstleister Heroku ist auschließlich ein git push auf das Heroku Repository auszuführen[1]:

```
git push heroku master
Initializing repository, done.
Counting objects: 110, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (87/87), done.
Writing objects: 100% (110/110), 212.71 KiB | 0 bytes/s, done.
Total 110 (delta 30), reused 0 (delta 0)

-----> Java app detected
-----> Installing OpenJDK 1.8... done
-----> Installing Maven 3.3.3... done
-----> Executing: mvn -B -DskipTests=true clean install
   [INFO] Scanning for projects...
...
   [INFO] ------------------------------------------------------------------------
   [INFO] BUILD SUCCESS
   [INFO] ------------------------------------------------------------------------
   [INFO] Total time: 11.417s
   [INFO] Finished at: Thu Sep 11 17:16:38 UTC 2014
   [INFO] Final Memory: 21M/649M
   [INFO] ------------------------------------------------------------------------
-----> Discovering process types
   Procfile declares types -> web
```

Die Anwendung kann anschließend unter folgender URL erreicht werden:

```
https://enigmatic-waters-31128.herokuapp.com
```

4.2 Frontend

Bla Bla Bla.

[1] https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku

5 Projektorganisation

A Anhang

A.1 API Dokumentation

A.1.1 API Aufruf: Registrieren eines neuen Benutzers

Bezeichnung	Registrieren eines neuen Benutzers
URL	`/api/register`
Methode	POST

Parameter	`{`
	`"username":[String],`
	`"password":[alphanumeric]`
	`}`

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`X-AUTH-TOKEN: eyJpZCI6MTIsInVz...`
	`Content-Type: application/json;charset=UTF-8`
	`Content-Length: 72`
	
	`user successfully registered`

Fehlerantwort	`HTTP/1.1 422 Unprocessable Entity`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 17`
	
	`password to short`

Beispiel	`$ curl 'http://localhost:8080/api/register' -i -X POST -H 'Content-Type: application/`
	`json' -d`
	`'{"username":"testuser", "password": "testpassword"}'`

A.1.2 API Aufruf: Als registrierter Benutzer anmelden

Bezeichnung	Als registrierter Benutzer anmelden
URL	`/api/login`
Methode	POST

Parameter	`{`
	`"username":[String],`
	`"password":[alphanumeric]`
	`}`

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`X-AUTH-TOKEN: eyJpZCI6MTEsInVz...`

Fehlerantwort	`HTTP/1.1 401 Unauthorized`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`

Beispiel	`$ curl 'http://localhost:8080/api/login' -i -X POST -H 'Content-Type: application/ json' -d` `'{"username":"user", "password": "user"}'`

A.1.3 API Aufruf: Einen Raum registrieren

Bezeichnung	Einen Raum registrieren
URL	`/api/user/rooms/:id`
Methode	PUT
URL-Parameter	**erforderlich:** `id=[String]` Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843

Parameter	`{`
	`"name":[String]`
	`}`

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 28`
	
	`room successfully registered`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -X PUT -H 'Content-Type: application/json' -H 'X-AUTH-TOKEN: eyJpZCI6MTEsInVzZXJuYW...' -d '{"name":"Wohnzimmer"}'`

A.1.4 API Aufruf: Einen Raum von der Datenbank entfernen

Bezeichnung	Einen spezifischen Raum von der Datenbank entfernen
URL	`/api/user/rooms/:id`
Methode	DELETE
URL-Parameter	**erforderlich:** `id=[alphanumeric]` Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 25`
	
	`room successfully removed`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 14`
	
	`room not found`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -X DELETE -H 'X-AUTH-TOKEN: eyJpZCI6MTEsInVzZXJuYW1l...'`

A.1.5 API Aufruf: Einen spezifischen Raum abrufen

Bezeichnung	Einen spezifischen Raum abrufen
URL	`/api/user/rooms/:id`
Methode	GET
URL-Parameter	**erforderlich:** `id=[alphanumeric]` Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: application/json;charset=UTF-8`
	`Content-Length: 69`
	
	`"roomId":"d934eb20-4c6f-4d1c-91c5-61cdeddcf843","name":"Wohnzimmer"`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -H 'X-AUTH-TOKEN: eyJpZCI6MTEsInVzZXJu...'`

A.1.6 API Aufruf: Eine Liste aller Räume abrufen

Bezeichnung	Eine Liste aller Räume abrufen
URL	`/api/user/rooms`
Methode	GET

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: application/json;charset=UTF-8`
	`Content-Length: 71`
	
	`[{"roomId":"d934eb20-4c6f-4d1c-91c5-61cdeddcf843","name":"Wohnzimmer"}]`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms' -i -H 'X-AUTH-TOKEN: eyJpZCI6MTEsInVzZ67KDv...'`

A.1.7 API Aufruf: Ein neues Gerät registrieren

Bezeichnung	Ein neues Gerät registrieren
URL	`/api/user/rooms/:roomId/devices/:deviceId`
Methode	PUT

URL-Parameter	**erforderlich:**
	`roomId=[alphanumeric]`
	Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843
	
	`deviceId=[alphanumeric]`
	Beispiel: id=a934eb20-4c6f-4d1c-91c5-61cdeddcf843

Parameter	`{`
	`"type":["HEATING" | "LIGHT"],`
	`"name":[String]`
	`}`

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 17`
	
	`device registered`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 14`
	
	`room not found`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843/devices/a614eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -X PUT -H 'Content-Type: application/json' -H 'X-AUTH-TOKEN: eyJpZCI6MT...' -d '{"deviceId":"a614eb20-4c6f-4d1c-91c5-61cdeddcf843","name":"Licht","type":"LIGHT"}'`

A.1.8 API Aufruf: Ein spezifisches Gerät abrufen

Bezeichnung	Ein spezifisches Gerät abrufen
URL	`/api/user/rooms/:roomId/devices/:deviceId`
Methode	GET

URL-Parameter	**erforderlich:**
	`roomId=[alphanumeric]`
	Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843
	
	`deviceId=[alphanumeric]`
	Beispiel: id=a934eb20-4c6f-4d1c-91c5-61cdeddcf843

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: application/json;charset=UTF-8`
	`Content-Length: 81`
	
	`{"deviceId":"a614eb20-4c6f-4d1c-91c5-61cdeddcf843","name":"Licht","type":"LIGHT"}`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843/devices/a614eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -H 'X-AUTH-TOKEN: eyJpZCI6MTEsInVzZXJuYW1lIjoi...'`

A.1.9 API Aufruf: Ein Gerät aus der Datenbank entfernen

Bezeichnung	Ein spezifisches Gerät aus der Datenbank entfernen
URL	`/api/user/rooms/:roomId/devices/:deviceId`
Methode	DELETE

URL-Parameter	**erforderlich:**
	`roomId=[alphanumeric]`
	Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843
	
	`deviceId=[alphanumeric]`
	Beispiel: id=a934eb20-4c6f-4d1c-91c5-61cdeddcf843

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 27`
	
	`device successfully removed`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 16`
	
	`device not found`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843/devices/a614eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -X DELETE -H 'X-AUTH-TOKEN: eyJpZCI6MTEsInVzZXJuYW1...'`

A.1.10 API Aufruf: Alle Geräte eines Raumes auflisten

Bezeichnung	Alle Geräte eines Raumes auflisten
URL	`/api/user/rooms/:id/devices`
Methode	GET
URL-Parameter	**erforderlich:** `id=[alphanumeric]` Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: application/json;charset=UTF-8`
	`Content-Length: 83`
	
	`[{"deviceId":"a614eb20-4c6f-4d1c-91c5-61cdeddcf843","name":"Licht","type":"LIGHT"}]`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Pragma: no-cache`
	`Expires: 0`
	`X-Frame-Options: DENY`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843/devices' -i -H 'X-AUTH-TOKEN: eyJpZCI6MTEsIyu2ZOL67KDvf1RPcmfC8dA...'`

A.1.11 API Aufruf: Verändern der Gerätedaten

Bezeichnung	Verändern von Gerätedaten
URL	`/api/user/rooms/:roomId/devices/:deviceId`
Methode	PATCH

URL-Parameter	**erforderlich:**
	`roomId=[alphanumeric]`
	Beispiel: id=d934eb20-4c6f-4d1c-91c5-61cdeddcf843
	
	`deviceId=[alphanumeric]`
	Beispiel: id=a934eb20-4c6f-4d1c-91c5-61cdeddcf843

Parameter	`{`
	`"targetValue":[numeric],`
	`"value":[numeric]`
	`}`

Erfolgsantwort	`HTTP/1.1 200 OK`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 17`
	
	`device updated`

Fehlerantwort	`HTTP/1.1 404 Not Found`
	`X-Content-Type-Options: nosniff`
	`X-XSS-Protection: 1; mode=block`
	`Cache-Control: no-cache, no-store, max-age=0, must-revalidate`
	`Expires: 0`
	`X-Frame-Options: DENY`
	`Content-Type: text/plain;charset=UTF-8`
	`Content-Length: 14`
	
	`room not found`

Beispiel	`$ curl 'http://localhost:8080/api/user/rooms/d934eb20-4c6f-4d1c-91c5-61cdeddcf843/devices/a614eb20-4c6f-4d1c-91c5-61cdeddcf843' -i -X PATCH -H 'Content-Type: application/json' -H 'X-AUTH-TOKEN: eyJpZCI6MT...' -d '{"deviceId":"a614eb20-4c6f-4d1c-91c5-61cdeddcf843","targetValue":"0"}'`