The API will handle the http requests: GET, POST, … and our API will handle following routes:



When your hosting the api locally, then your request will look like this:

GET every dispenser: <http://localhost:8000/api/dispenser>

GET every company: <http://localhost:8000/api/company>

GET every building: <http://localhost:8000/api/building>

…

POST a dispenser: <http://localhost:8000/api/dispenser>

...

BUT to post a dispenser/company/building/… you also need to send some parameters within the BODY of your http request. NOT as optional parameter where you type ‘?key=value’ directly behind your url.

**Parameters:**

**Dispenser:** 1) waterlevel

2) lastChangedTime

3) temperature

4) mode (this needs to be ‘on’ or ‘off’)

5) area\_id

6) serialNumber (has to be added, not yet implemented)

**Company:** 1) name

**Building:** 1) name

2) city

3) ZIPcode

4) address

5) company\_id

**Floor:**  1) name

2) blueprint (url / reference to an image)

3) building\_id

**Area:** 1) name

2) conversionRate

3) startX

4) startY

5) endX

6) endY

7) floor\_id

So that is that for GET and POST.

If you want to retrieve a specific instance than you can place ‘/{id}’ behind your url so for example:

<http://localhost:8000/api/dispenser/43>

(I noticed that this was available for company, building, floor & area. Dispenser will be there with next merge.)

But in this case the dispenser knows what his id is and as discussed the devices will not know anything except for what their serial number is?

(I’m thinking how I can implement this, maybe not working with id but only serial number instead …)

**Needs to be done:**

But we (as discussed in whatsapp) wanted to work with a dispenser with a serial number. The database has to be changed for that so an extra parameter will be added to dispensers. This time I think I will implement that with a optional parameter.

So IN THE FUTURE if a dispenser wants to retrieve it’s data he has to request it like following example

<http://localhost:8000/api/dispenser?SerialNumber=1234567>

The updating method also needs to be implemented so a dispenser can do this.