



# **web.py**

## **Web-Anwendungen in Python**

**Uwe Berger**  
bergeruw@gmx.net

---

# Uwe Berger



# Motivation



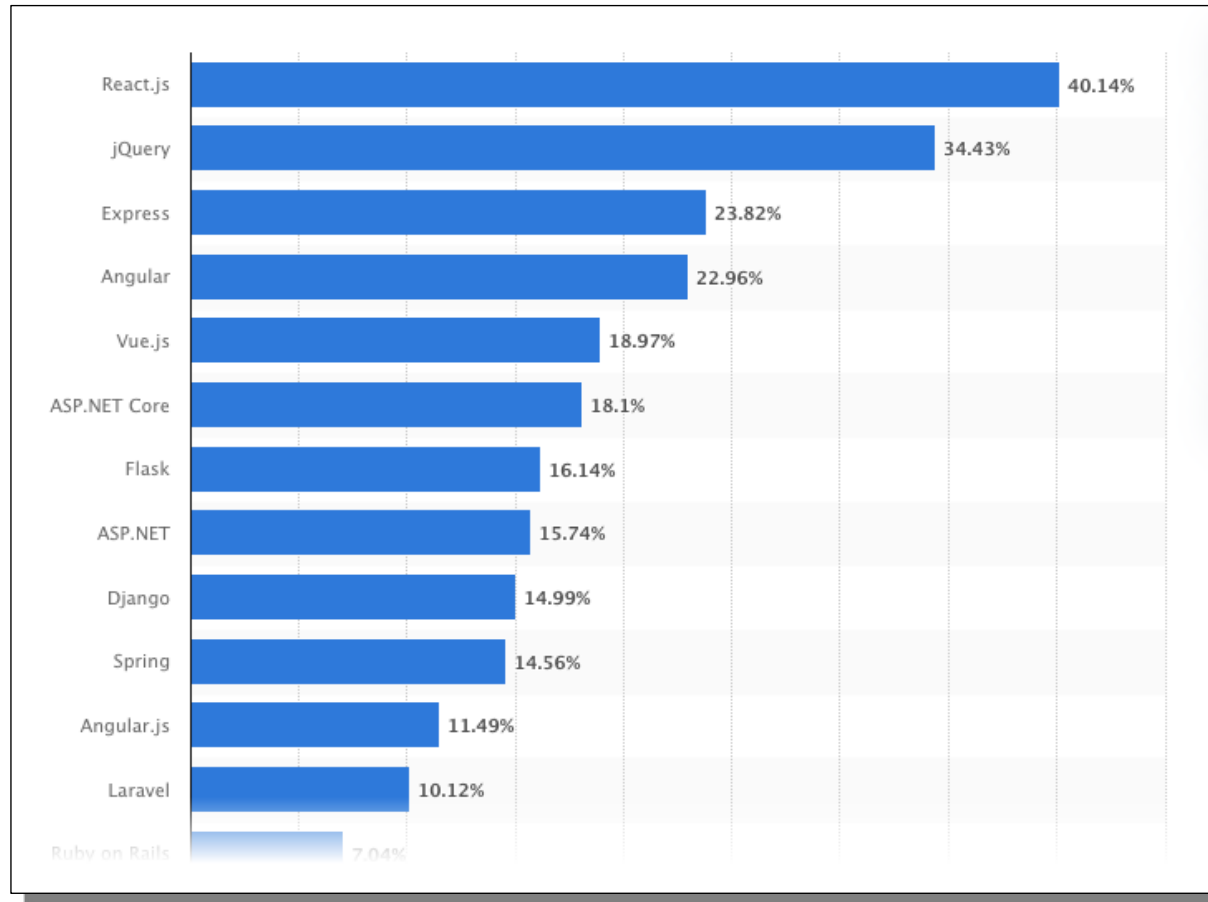
# Was erzähle ich heute?

- Wirklich nur ganz kurz....:
  - Web-Frameworks
  - Python → nö, ...RTFM!
  - „Kennzahlen“ zu web.py
- web.py → 6x „Hello World“
- Meine web.py-Applikationen

# Web-Frameworks

- „Framework“ (Rahmenstruktur): Programmierahmen/-gerüst
- Web-Framework: Programmierahmen zum Entwickeln von dynamischen Webseiten, -anwendungen, -services
- ...meist sind u.a. Mechanismen für folgende Dinge enthalten:
  - Templates
  - Authentifizierung
  - Mailversand
  - Formulare
  - Datenbankzugriffe

# Web-Frameworks



Quelle: <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>

- Aaron Swartz ( → <http://www.aaronsw.com/>)
- <https://webpy.org>
- → <https://github.com/webpy/webpy/blob/master/LICENSE.txt>:  
*„web.py is in the public domain; it can be used for whatever purpose with absolutely no restrictions.“*
- Installation (Alternativen):
  - entspr. Distributionspaket...
  - Download etc. von [github.com](https://github.com)
  - `pip3 install web.py`



# „Hello World!“

Siehe auch:

→ [https://github.com/boerge42/webpy-apps/tree/master/web\\_hello\\_world](https://github.com/boerge42/webpy-apps/tree/master/web_hello_world)

- hello1.py
- ...
- hello6.py



# „Hello World!“ (hello1.py)

hello1.py

- 10 Zeilen Python-Code → Fertig!

# „Hello World!“ (hello1.py)

```
import web

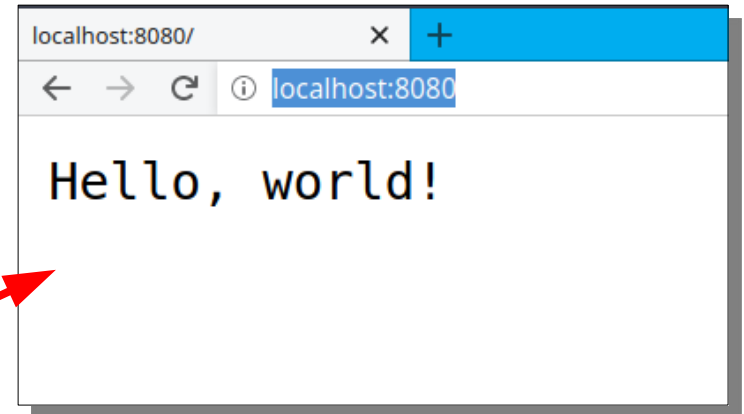
urls = (
    '/', 'index'
)

class index:
    def GET(self):
        return "Hello, world!"

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

>python3 hello1.py

→ http://localhost:8080/



# „Hello World!“ (hello2.py)

hello2.py

- ...plus HTML-Template

# „Hello World!“ (hello2.py)

```
import web

urls = (
    '/', 'index'
)

render = web.template.render('templates/')

class index:
    def GET(self):
        hello = 'Hello world!'
        return render.index(hello)

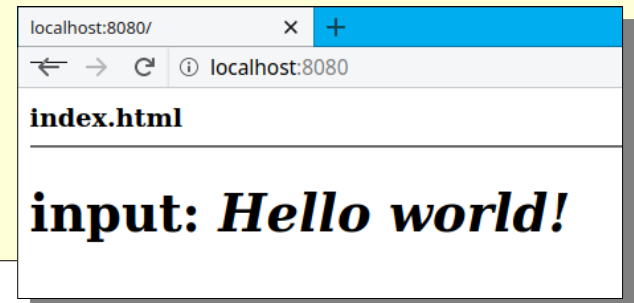
if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

templates/**index.html**

```
$def with (hello)

<b>index.html</b>
<hr>

$if hello:
    <h1>input: <em>$hello</em></h1>
$else:
    <b>...no input!</b>
```



# „Hello World!“ (hello3.py)

hello3.py

- ...plus „Rahmen-Template“

# „Hello World!“ (hello3.py)

```
import web

urls = (
    '/', 'index'
)

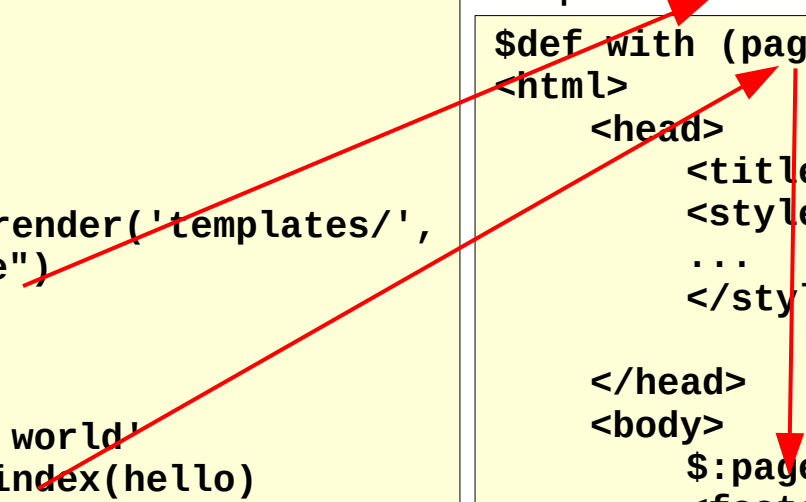
render = web.template.render('templates/',
                             base="base")

class index:
    def GET(self):
        hello = 'Hello world!'
        return render.index(hello)

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

templates/**base.html**

```
$def with (page)
<html>
  <head>
    <title>Hello World...</title>
    <style>
      ...
    </style>
  </head>
  <body>
    $:page
    <footer>
      <hr>Uwe Berger; 2022
    </footer>
  </body>
</html>
```



# „Hello World!“ (hello3.py)

```
import web

urls = (
    '/', 'index'
)

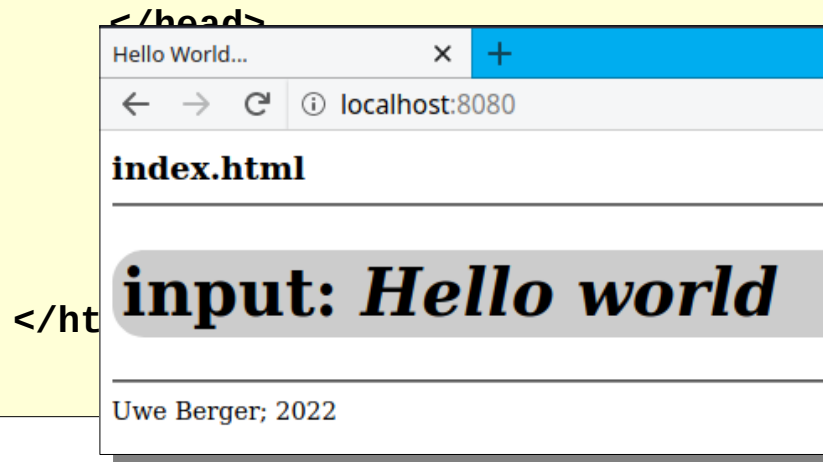
render = web.template.render('templates/',
                              base="base")

class index:
    def GET(self):
        hello = 'Hello world'
        return render.index(hello)

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

templates/**base.html**

```
$def with (page)
<html>
  <head>
    <title>Hello World...</title>
    <style>
      ...
    </style>
```



# „Hello World!“ (hello4.py)

hello4.py

- „Variablenübergabe“ via URL



# „Hello World!“ (hello4.py)

```
import web

urls = (
    '/(.*)', 'index'
)

render = web.template.render('templates/',
                              base="base")

class index:
    def GET(self, hello):
        return render.index(hello)

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

templates/index.html

```
$def with (hello)

<b>index.html</b>
<hr>

$if hello:
    <h1>input: <em>$hello</em></h1>
$else:
    <b>...no input!</b>
```

# „Hello World!“ (hello4.py)

```
import web
```

```
urls = (  
    '/', 'index'  
)
```

```
render('templates/index.html',
```

```
class Context: input: blablaba
```

```
if __name__ == '__main__':
```

```
    app.run()
```

templates/index.html

```
$def with (hello)
```

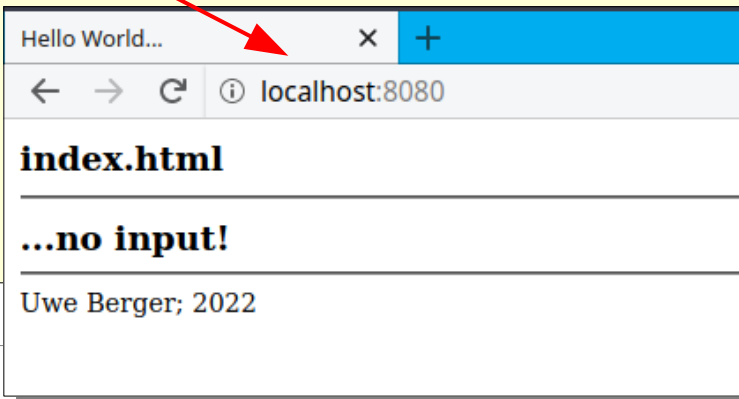
```
<b>index.html</b>  
<hr>
```

```
$if hello:
```

```
    <h1>input: <em>${hello}</em></h1>
```

```
$else:
```

```
    <b>...no input!</b>
```



# „Hello World!“ (hello5.py)

hello5.py

- Variableneingabe via HTML-Formular

# „Hello World“ (hello5.py)

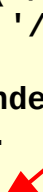
```
...
urls = ('/form', 'form',
        '/(.*)', 'index')
...
class index:
    ...
class form:

    valid = web.form.regexp(r"^[0-9]+$", '--> must be numeric!')

    formular=web.form.Form(
        web.form.Textbox("hello", valid, description="input: "),
        web.form.Button("send..."))

    def GET(self):
        return render.form(self.formular)

    def POST(self):
        if not self.formular.validates():
            return render.form(self.formular)
        else:
            return render.index(self.formular.d.hello)
...
```



templates/index.html

```
...
```

templates/form.html

```
$def with (formular)

<b>form.html</b>
<hr>

<form method="POST">
    $:formular.render()
</form>
```

# „Hello World“ (hello5.py)

```
...
urls = ('/form', 'form',
        '/(.*)', 'index')
...
class index:
    ...
class form:
    valid = web.form.regexp(r"^[0-9]+$", '--> must be numeric!')
    formular=web.form.Form(
        web.form.Textbox("hello", valid, description="input: "),
        web.form.Button("send..."))
    def GET(self):
        return render.form(self.formular)
    def POST(self):
        if not self.formular.validates():
            return render.form(self.formular)
        else:
            return render.index(self.formular.d.hello)
...
```

templates/index.html

```
...
```

templates/form.html

```
$def with (formular)
<b>form.html</b>
<hr>
<form method="POST">
    $:formular.render()
</form>
```

# „Hello World“ (hello5.py)

```
...
urls = ('/form', 'form',
        '/(.*)', 'index')
...
class index:
    ...

class form:

    valid = web.form.regexp(r"^[0-9]+$", '--> must be numeric!')

    formular=web.form.Form(
        web.form.Textbox("hello", valid, description="input: "),
        web.form.Button("send..."))

    def GET(self):
        return render_form(self.formular)

    def POST(self):
        if not self.formular.validates():
            return render_form(self.formular)
        else:
            return render_index(self.formular.d.hello)
...

```

templates/index.html

```
...
```

templates/form.html

```
$def with (formular)
<b>form.html</b>
<hr>
<form method="POST">
    $:formular.render()
</form>

```

# „Hello World“ (hello5.py)

```
...
urls = ( '/form',
         '/*.*' )
...
class index:
    ...

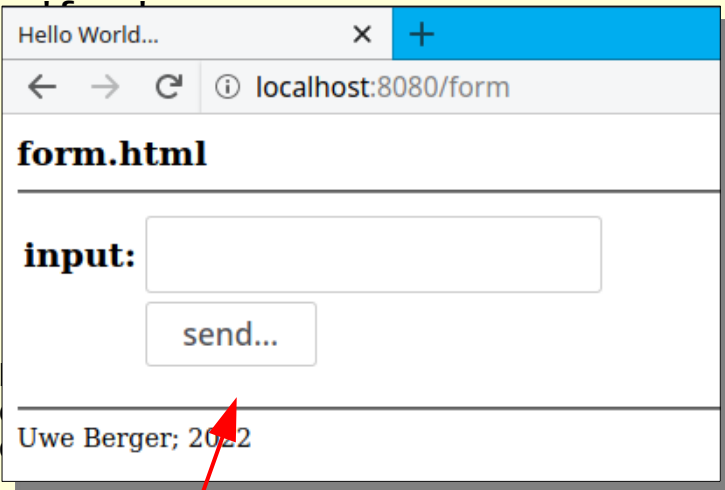
class form:

    valid = web
    formular=we
    web.f
    web.f

    def GET(self):
        return render.form(self.formular)

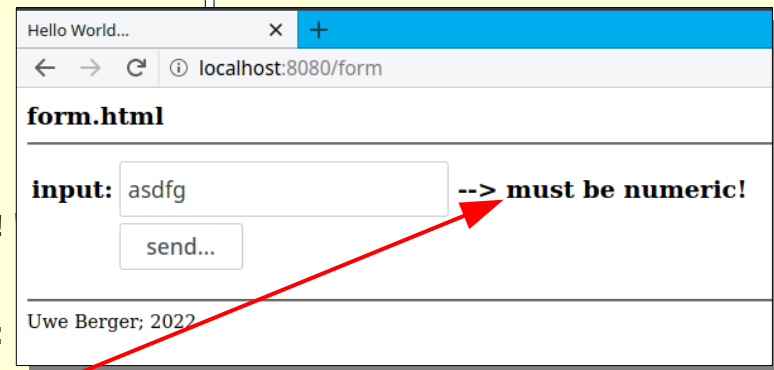
    def POST(self):
        if not self.formular.validates():
            return render.form(self.formular)
        else:
            return render.index(self.formular.d.hello)
...

```

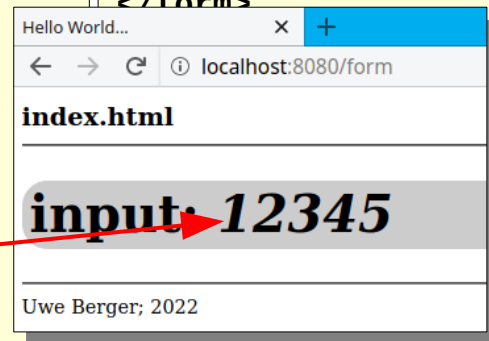


numeric!  
n="input:"

templates/index.html



\$.formular.render()



# „Hello World!“ (hello6.py)

hello6.py

- Datenbankzugriffe



# Datenbank einbinden (hello6.py)

```
import web

urls = (
    '/', 'database'
)

db = web.database(dbn='sqlite', db='db.sqlite')

render = web.template.render('templates/',
                              base="base")

class database:
    def GET(self):
        fn = db.select("person")
        return render.database(fn)

if __name__ == "__main__":
    app = web.application(urls, globals())
    app.run()
```

templates/database.html

```
$def with (fn)

database.html


---






```

# Datenbank einbinden (hello6.py)

```
import web
```

```
sqlite> .schema  
CREATE TABLE person (first text, name text);
```

```
db = web.database(dbn='sqlite', db='db.sqlite')
```

```
render = web.template.render('templates/',  
                              base="base")
```

```
class database:  
    def GET(self):  
        fn = db.select("person")  
        return render.database(fn)
```

```
if __name__ == "__main__":  
    app = web.application(urls, globals())  
    app.run()
```

templates/database.html

```
$def with (fn)  
<b>database.html</b>  
<hr>  
<table border=1 cellpadding="5">  
    <tr>  
        <th>first</th>  
        <th>name</th>  
    </tr>  
    $for l in fn:  
        <tr>  
            <td>${l["first"]}</td>  
            <td>${l["name"]}</td>  
        </tr>  
</table>
```

# Datenbank einbinden (hello6.py)

```
import web

urls = (
    '/', 'database'
)

db = web.database(dbn='sqlite')

render = web.template.base

class database:
    def GET(self):
        fn = db.select('SELECT first, name FROM users')
        return render('database.html', dict(data=fn))

if __name__ == '__main__':
    app = web.application(urls, globals())
    app.run()
```

first	name
Uwe	Berger
Max	Mustermann
Elfriede	Schulze
Wilfried	Schumacher
Linus	Torvalds

Uwe Berger; 2022

templates/database.html

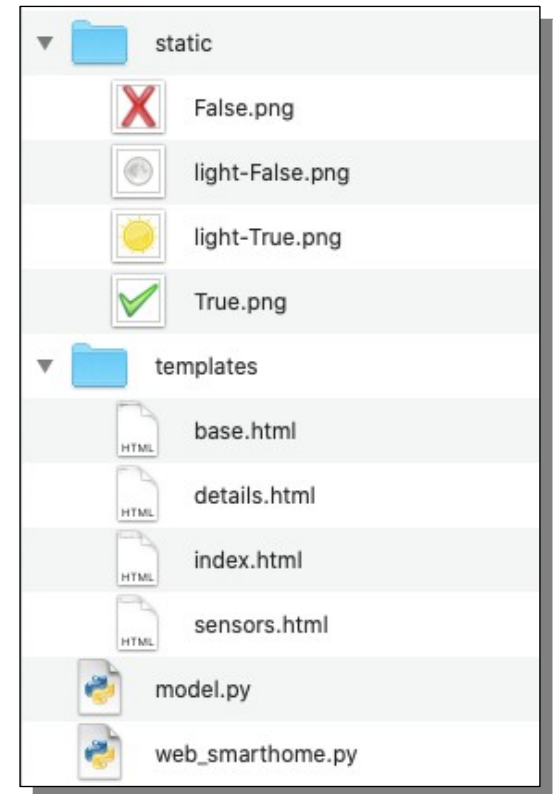
```
$def with (fn)

<b>database.html</b>
<hr>

<table border=1 cellpadding="5">
  <tr>
    <th>first</th>
    <th>name</th>
  </tr>
  $for l in fn:
    <tr>
      <td>${l["first"]}</td>
      <td>${l["name"]}</td>
    </tr>
</table>
```

# (Eine) typische Programmstruktur

- Root-Verzeichnis der Applikation:
  - Eigentliche web.py-Applikation
  - Als Python-Modul gekapselte Zugriffe auf Daten u.ä. (model.py)
- HTML-Templates
  - Verzeichnis „templates“
- „Unveränderliche“ Daten (z.B. Bilder)
  - Verzeichnis „static“



# War das alles?

→ <https://webpy.org/docs/0.3/tutorial>

→ <https://webpy.org/cookbook/>

- „Advanced“ Templates
- Sessions, Cookies
- Authentifizierung, SSL
- Mail
- File-Upload
- ...

## Advanced

- Contextual and Environment variables - web.ctx
- Application processors, loadhooks and unloadhooks
- How to use web.background
- Custom NotFound message
- How to Stream Large Files
- Control over logging for default HTTPServer
- SSL support in built-in cherrypy server
- Run-time language switch

## Sessions and user state

- Working with Session
- Using session with reloader
- Using session in template
- Working with Cookies
- User authentication
- User authentication with http basic auth (RFC2617)
- User authentication with PostgreSQL database
- Sessions with sub-apps
- Unpack session stored in postgresql

## Utils

- Sending Mail
- Sending Mail Using Gmail
- Webservice using soaplib + WSDL

## Templates

- Templator: The web.py templating system
- Using Site Layout Templates
- Alternating Style
- Import functions into templates
- i18n support in template file
- Use Mako template engine in webpy
- Use Cheetah template engine in webpy
- Use Jinja2 template engine in webpy
- How to use templates on Google App Engine
- Concatenate two rendered templates

## Testing

- Testing with Paste and Nose
- RESTful doctest using an application's request method

## User input

- File Upload
- Store an uploaded file
- How to put a limit of size of uploaded files
- Accessing user input through web.input
- How to use forms
- Render individual form fields

# Da war doch was... → CD-Search

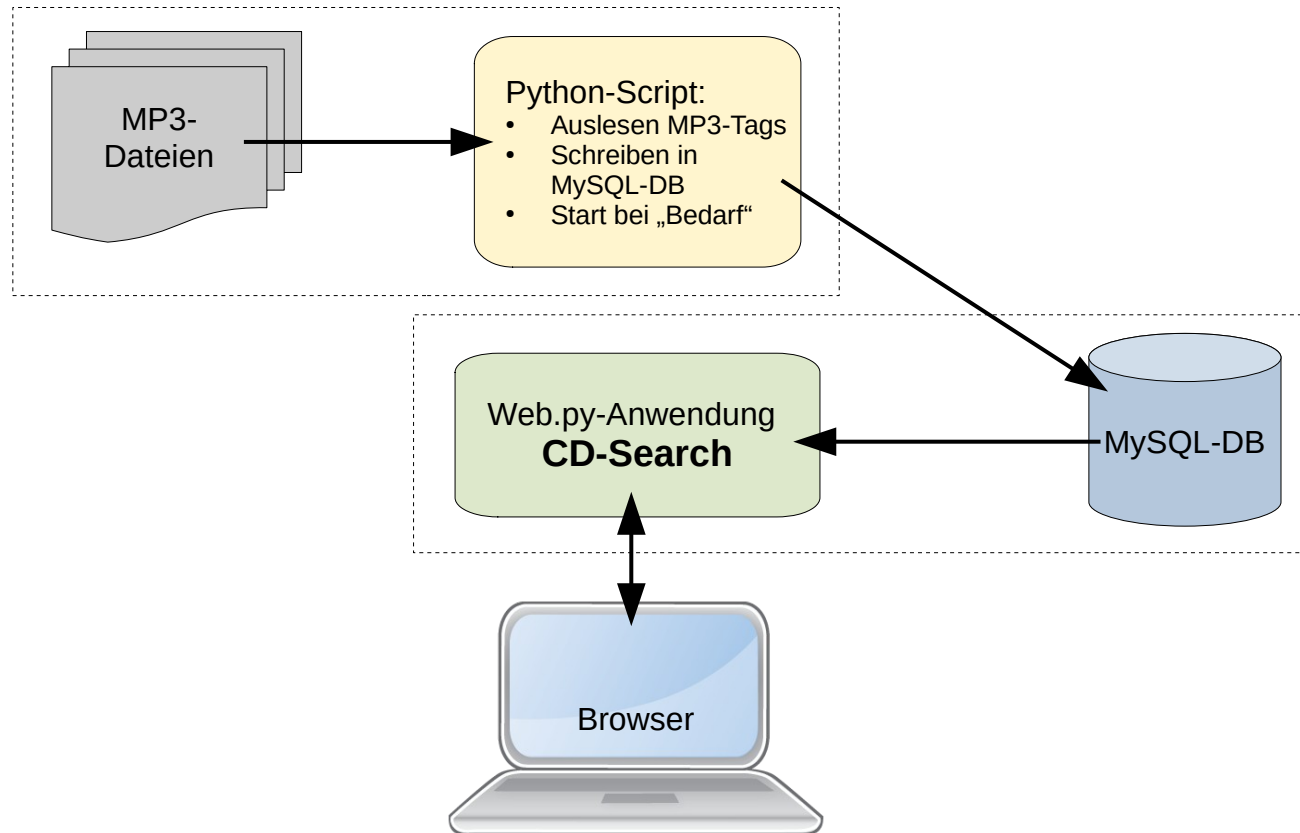
→ [https://github.com/boerge42/webpy-apps/tree/master/web\\_cd\\_search](https://github.com/boerge42/webpy-apps/tree/master/web_cd_search)

- Ziele (CD-Search):

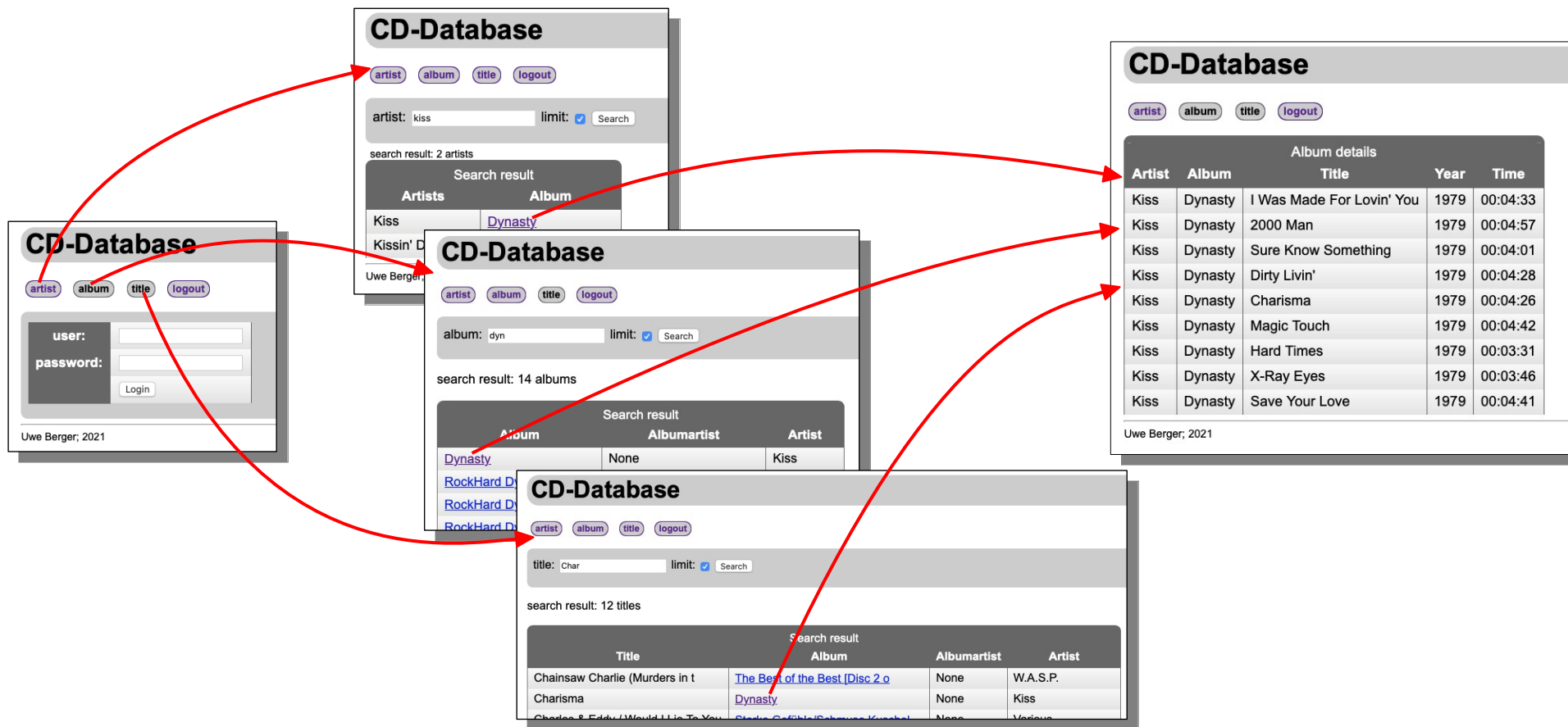
- „Nicht nochmal eine doppelte CD kaufen!“
- eine plattformunabhängige Anwendung
- von überall erreichbar
- möglichst aktuelle Datenbasis



# CD-Search (Architektur)



# CD-Search (Web-Oberfläche)





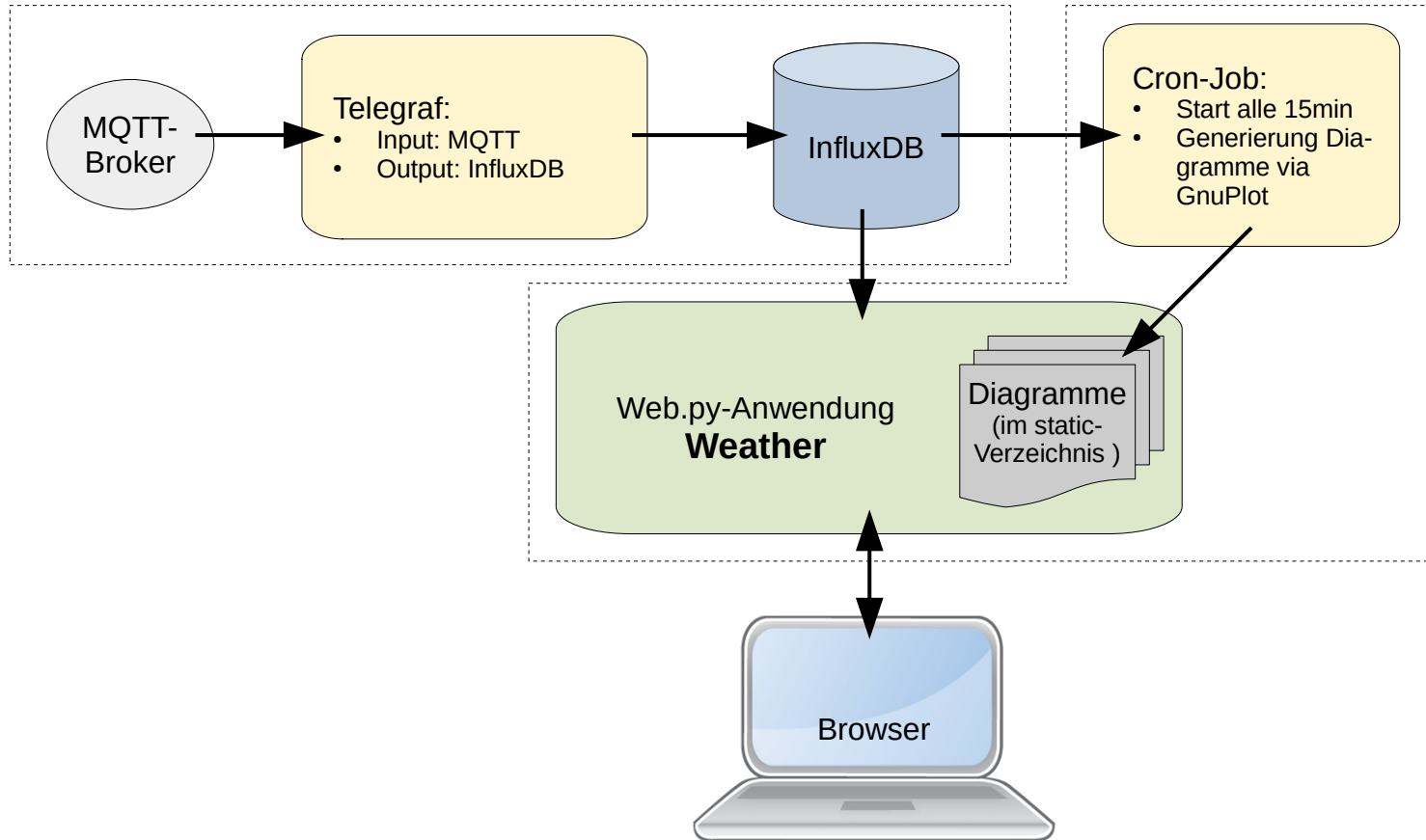
# Web-Anwendung: Weather

→ [https://github.com/boerge42/webpy-apps/tree/master/web\\_weather](https://github.com/boerge42/webpy-apps/tree/master/web_weather)

- Ziele:

- eine steinalte, langweilige Webseite renovieren...
- Python → gnuplot
- es gibt noch andere Datenbanken ... → InfluxDB

# Weather (Architektur)



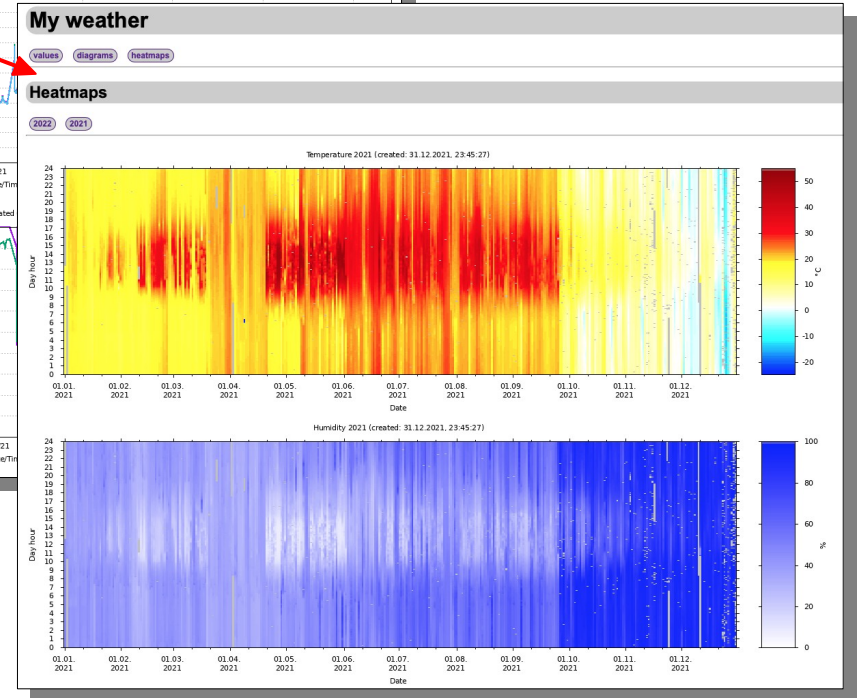
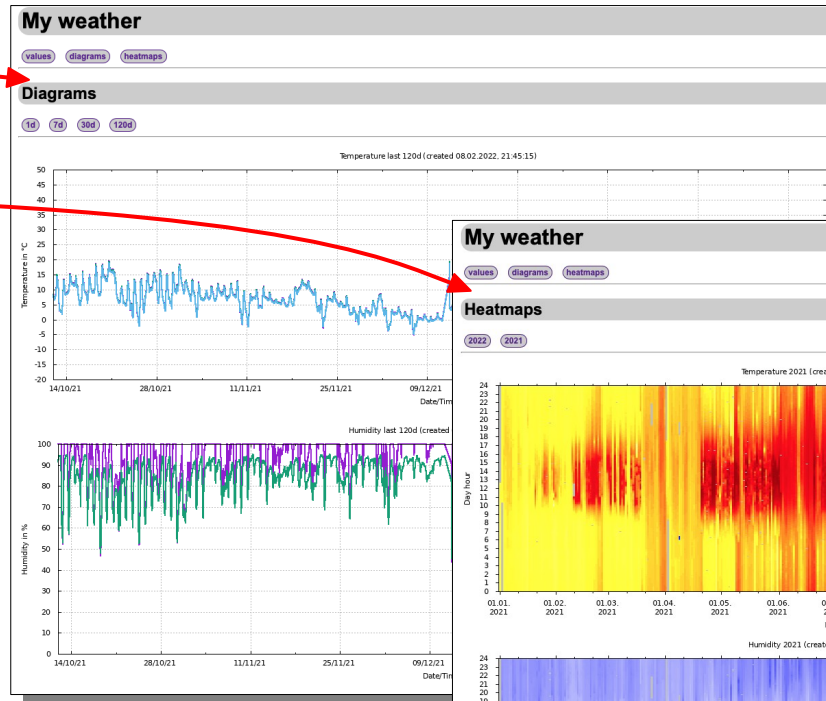
# Weather (Web-Oberfläche)

### My weather

values diagrams heatmaps

Sensor	Temperature	Pressure (rel.)	Humidity	Luminosity
<b>Current</b>				
TMP36	8.5°C	-	-	-
SHT15	8.9°C	-	91.8%	-
BME280	8.8°C	1026.1hPa	100%	-
BH1750	-	-	-	0lux
<b>Min./Max. today</b>				
TMP36	1.3°C/9.2°C	-	-	-
SHT15	1.7°C/9.8°C	-	80.4%/92%	-
BME280	1.4°C/9.6°C	1023.1hPa/1026.7hPa	91.5%/100%	-
BH1750	-	-	-	0lux/546.7lux
<b>Min./Max. 1d</b>				
TMP36	1.3°C/9.2°C	-	-	-
SHT15	1.7°C/9.8°C	-	80.4%/92%	-
BME280	1.4°C/9.6°C	1023.1hPa/1026.7hPa	91.5%/100%	-
BH1750	-	-	-	0lux/546.7lux
<b>Min./Max. 7d</b>				
TMP36	-0.3°C/9.2°C	-	-	-
SHT15	0.2°C/9.8°C	-	56%/93%	-
BME280	-0.3°C/9.6°C	994.6hPa/1026.7hPa	57%/100%	-
BH1750	-	-	-	0lux/2346.7lux
<b>Min./Max. 30d</b>				
TMP36	-4.5°C/9.2°C	-	-	-
SHT15	-4.3°C/9.8°C	-	55.7%/94.9%	-
BME280	-4.4°C/9.6°C	994.6hPa/1040.8hPa	55.4%/100%	-
BH1750	-	-	-	0lux/2346.7lux

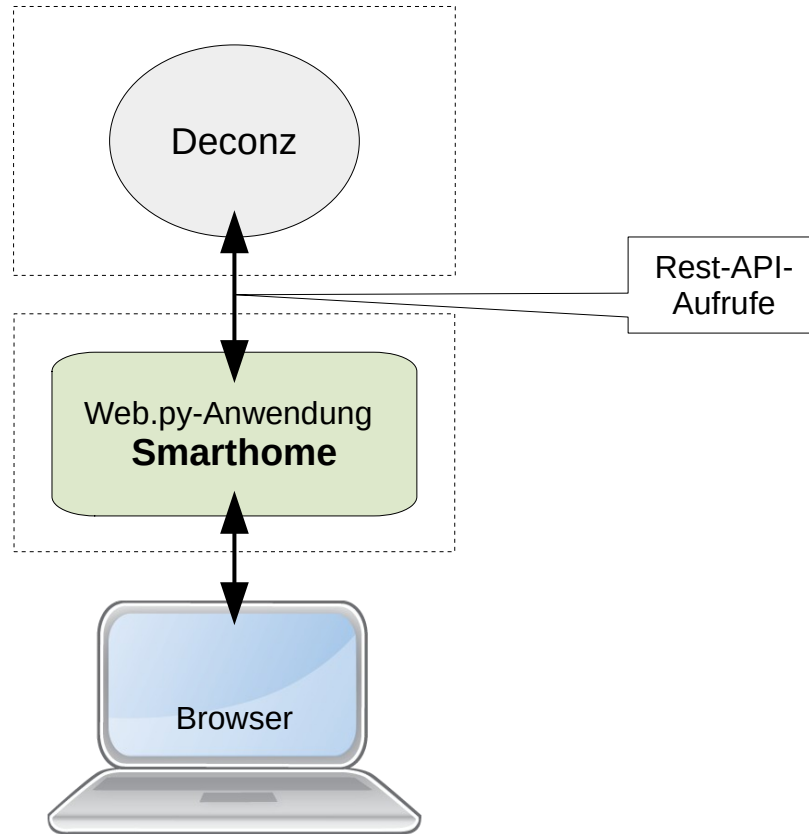
Last ESP-Data: Timestamp: 08.02.2022 21:52:40, Awake time: 1972ms, Vbat: 3.77V, Vcc: 3.29V  
Uwe Berger; 2021, 2022



# Web-Anwendung: Smart Home

- <https://programm.froscon.org/2021/events/2637.html>  
(Titel: „Ich wollte nie Smart Home machen...“)
- [https://github.com/boerge42/webpy-apps/tree/master/web\\_smarthome](https://github.com/boerge42/webpy-apps/tree/master/web_smarthome)
- Ziele:
  - keine Datenbank im Hintergrund; anzuzeigende Daten werden online via Rest-API-Aufrufe ermittelt/gesendet
  - dynamisierte Formulare
  - Validierung von Formulareingaben

# Smarthome (Architektur)



# Smarthome (Web-Oberfläche)

### My smarthome

lights sensors

#### Lights

Light list							
id	name	manufacturername	modelid	type	reachable	lastseen	on
1	<a href="#">Configuration tool 1</a>	dresden elektronik	ConBee II	Configuration tool	✓	08.02.2022 08:53	
2	<a href="#">Schreibtischlampe Dachboden</a>	Philips	LWE002	Dimmable light	✓	08.02.2022 08:54	254
3	<a href="#">Dachboden Decke rechts Mitte</a>	IKEA of Sweden	TRADFRI bulb GU10 WW 400lm	Dimmable light	✓	08.02.2022 08:54	254
4	<a href="#">Dachboden Decke rechts Treppe</a>	IKEA of Sweden	TRADFRI bulb GU10 WW 400lm	Dimmable light	✓	08.02.2022 08:54	254
5	<a href="#">Range extender 5</a>	IKEA of Sweden	TRADFRI Signal Repeater	Range extender	✓	08.02.2022 08:48	

#### Sensors

Sensor list									
id	name	manufacturername	modelid	type	reachable	lastseen	battery	value	unit
1	<a href="#">Daylight</a>	Philips	PHDL00	Daylight					
3	<a href="#">Bewegungsmelder Dachboden</a>	IKEA of Sweden	TRADFRI motion sensor	ZHAPresence	✓	08.02.2022 08:16	87	False	(presence)
14	<a href="#">Schalter1 ON/OFF Flurtreppe</a>	IKEA of Sweden	TRADFRI on/off switch	ZHASwitch	✓	08.02.2022 08:13	21	2002	(buttonevent)
15	<a href="#">Sirene Rauch</a>	IKEA of Sweden	TRADFRI on/off switch	ZHASwitch	✓	08.02.2022 08:17	34	1002	(buttonevent)
6	<a href="#">Bewegungsmelder Flurtreppe</a>	IKEA of Sweden	TRADFRI motion sensor	ZHAPresence	✓	08.02.2022 08:51	87	False	(presence)
7	<a href="#">Multisensor (Schlafzimmer)</a>	LUMI	lumi.weather	ZHATemperature	✓	08.02.2022 08:15	100	12.11	°C
8	<a href="#">Multisensor (Schlafzimmer)</a>	LUMI	lumi.weather	ZHAHumidity	✓	08.02.2022 08:15	100	52.59	%
9	<a href="#">Multisensor (Schlafzimmer)</a>	LUMI	lumi.weather	ZHAPressure	✓	08.02.2022 08:15	100	1017	hPa
10	<a href="#">OPPLE Schalter 3-fach</a>	LUMI	lumi.remote.b686opcnr1	ZHASwitch	✓	08.02.2022 08:37	100	4002	(buttonevent)
11	<a href="#">Taster Dachboden (unten)</a>	LUMI	lumi.remote.b186acn01	ZHASwitch	✓	08.02.2022 08:52	100	1002	(buttonevent)
12	<a href="#">Taster Dachboden (oben)</a>	LUMI	lumi.remote.b186acn01	ZHASwitch	✓	08.02.2022 08:34	100	1002	(buttonevent)
13	<a href="#">Multisensor (Dachboden)</a>	LUMI	lumi.weather	ZHATemperature	✗	30.01.2022 03:30	84	19.15	°C
14	<a href="#">Multisensor (Dachboden)</a>	LUMI	lumi.weather	ZHAHumidity	✗	30.01.2022 03:58	81	38.48	%
15	<a href="#">Multisensor (Dachboden)</a>	LUMI	lumi.weather	ZHAPressure	✗	30.01.2022 03:58	81	1007	hPa
16	<a href="#">Multisensor (Bad)</a>	LUMI	lumi.weather	ZHATemperature	✓	08.02.2022 08:55	100	22.29	°C
17	<a href="#">Multisensor (Bad)</a>	LUMI	lumi.weather	ZHAHumidity	✓	08.02.2022 08:55	100	32.08	%
18	<a href="#">Multisensor (Bad)</a>	LUMI	lumi.weather	ZHAPressure	✓	08.02.2022 08:55	100	1018	hPa
19	<a href="#">Rauchmelder 2</a>	Develco Products A/S	SMSZB-120	ZHAFire	✓	08.02.2022 08:54	100	False	(fire)
20	<a href="#">Rauchmelder 1</a>	Bitron Video	902010/24A	ZHAFire	✓	08.02.2022 08:54	100	False	(fire)
21	<a href="#">Motion Sensor</a>	SILVERCREST	TY020	ZHAPresence	✓	08.02.2022 08:29	100	False	(presence)
22	<a href="#">Rauchmelder 2</a>	Develco Products A/S	SMSZB-120	ZHATemperature	✓	08.02.2022 08:51	100	18.56	°C

Uwe Berger; 2021

### My smarthome

lights sensors

#### Details (lights, id=2)

```
etag: 85c522116a79a528cccd615a221d68e3
hascolor: False
lastannounced: 2022-01-07T22:11:44Z
lastseen: 2022-02-08T08:57Z
manufacturername: Philips
modelid: LWE002
name: Schreibtischlampe Dachboden
state:
  alert: none
  bri: 254
  on: True
  reachable: True
swversion: 1.53.3.227175
type: Dimmable light
uniqueid: 00:17:38:01:08:1c:4f:90-0b
```

Uwe Berger; 2021



### My smarthome

lights sensors

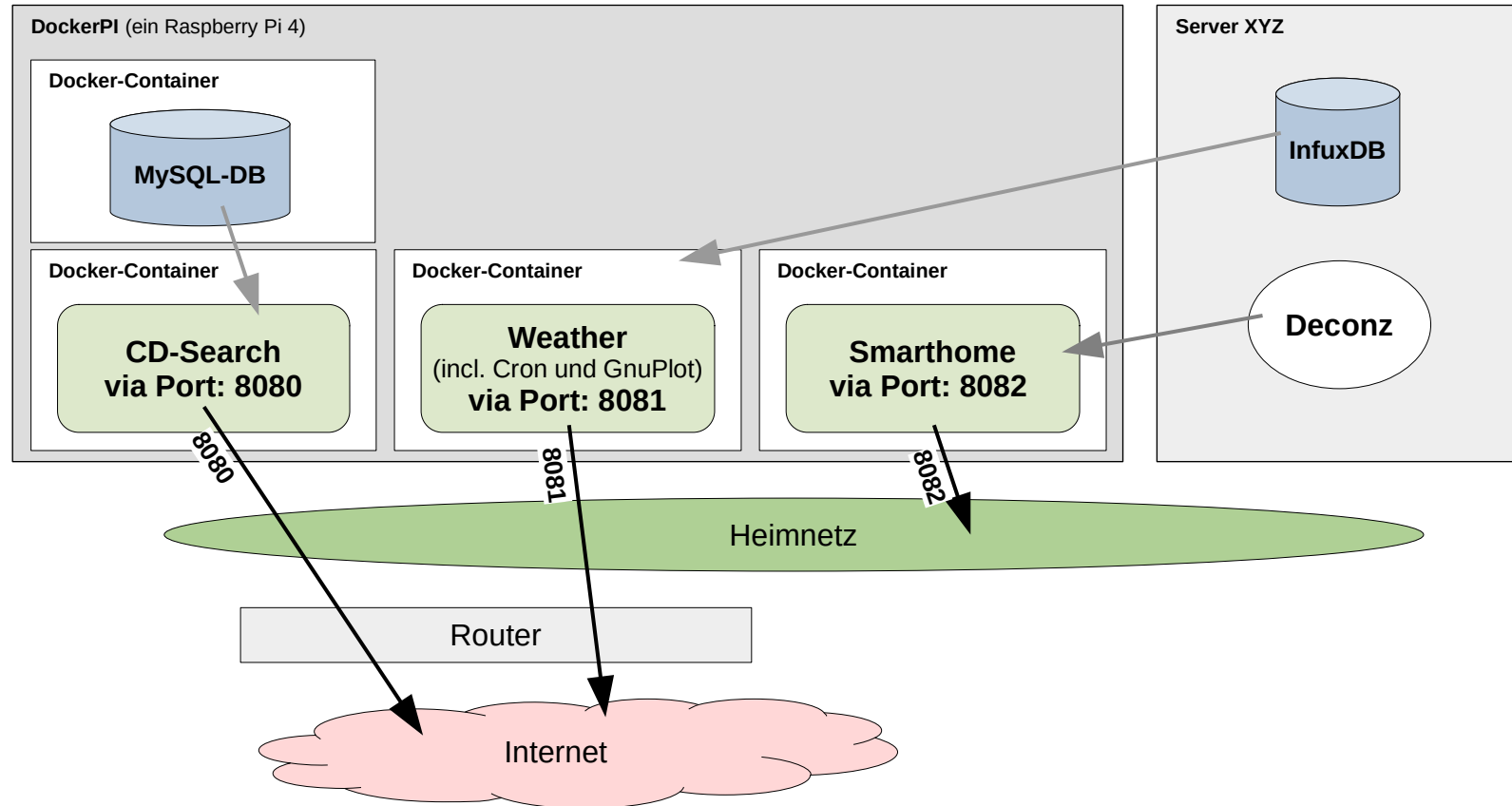
#### Details (sensors, id=3)

```
config:
  alert: none
  battery: 87
  delay: 180
  duration: 60
  group: 4
  on: True
  reachable: True
ep: 1
etag: 852aac820babc4211d9c325882e2e7b
lastseen: 2022-02-08T08:16Z
manufacturername: IKEA of Sweden
modelid: TRADFRI motion sensor
name: Bewegungsmelder Dachboden
state:
  dark: True
  lastupdated: 2022-02-08T07:13:25.217
  presence: False
swversion: 2.0.022
type: ZHAPresence
uniqueid: 84:2e:14:ff:fe:16:fd:1b-01-0006
```

Uwe Berger; 2021



# ...und alles in Docker-Containern



# Links

- <https://webpy.org/>
- <https://github.com/webpy/webpy>
- <https://www.linux-magazin.de/ausgaben/2006/08/die-zeit-laeuft/>
- <https://wiki.python.org/moin/WebFrameworks>
- <https://chemnitzer.linux-tage.de/2021/de/programm/beitrag/128>
- <https://github.com/boerge42/webpy-apps>





# Fragen?

...ansonsten Danke & Ende!