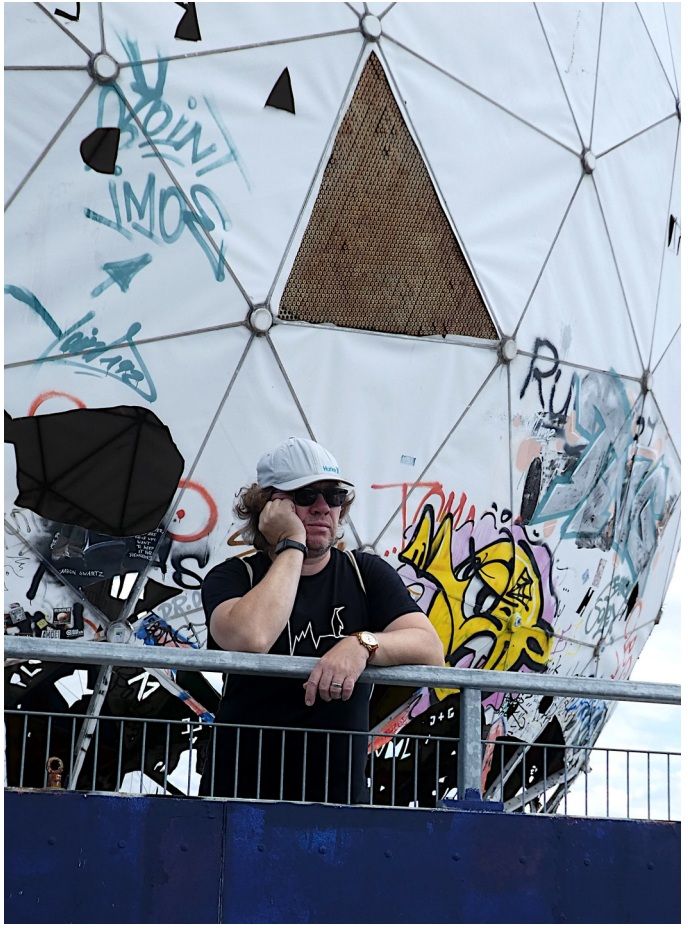
web.py
Web-Anwendungen in Python

Uwe Berger
bergeruw@gmx.net

Uwe Berger



Motivation

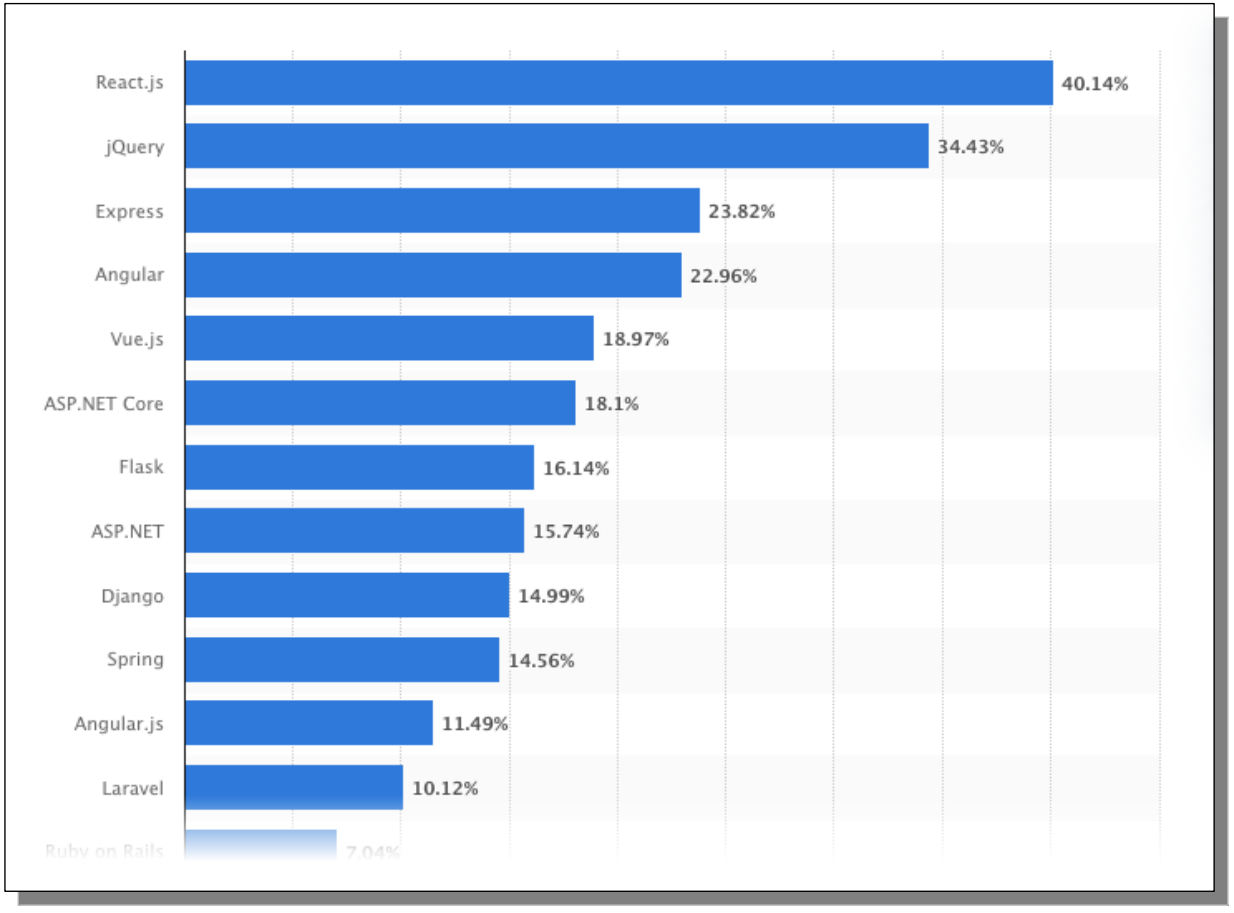
Was erzähle ich heute?

- 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): Programmierrahmen/-gerüst
- Web-Framework: Programmierrahmen 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/

web.py

- 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
  - pip3 install web.py

„Hello World!“

Siehe auch:
→ https://github.com/boerge42/webpy-apps/tree/master/web_hello_world

- hello1.py
- …
- hello6.py

„Hello World!“ (hello1.py)

hello1.py

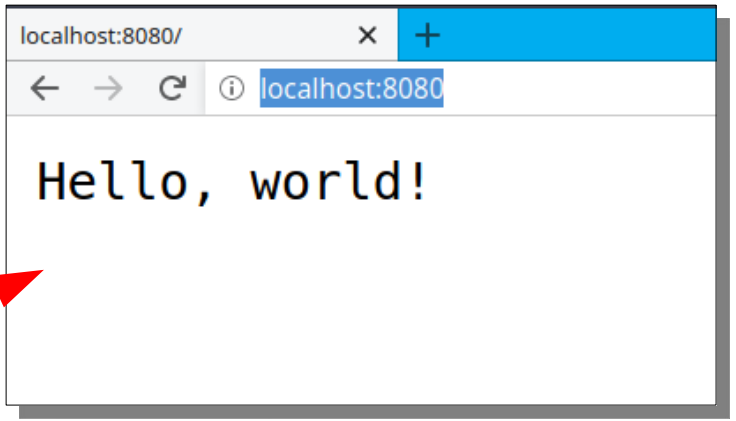
- 10 Zeilen Python-Code → Fertig!

„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

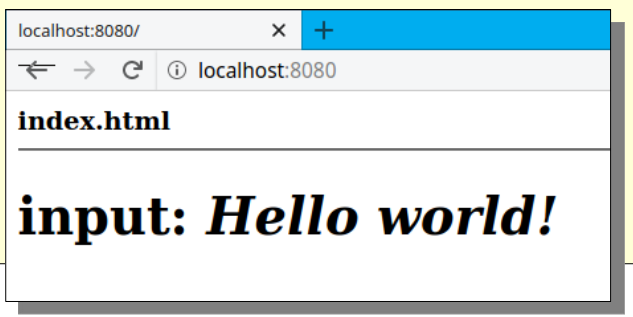
- ...plus HTML-Template

„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

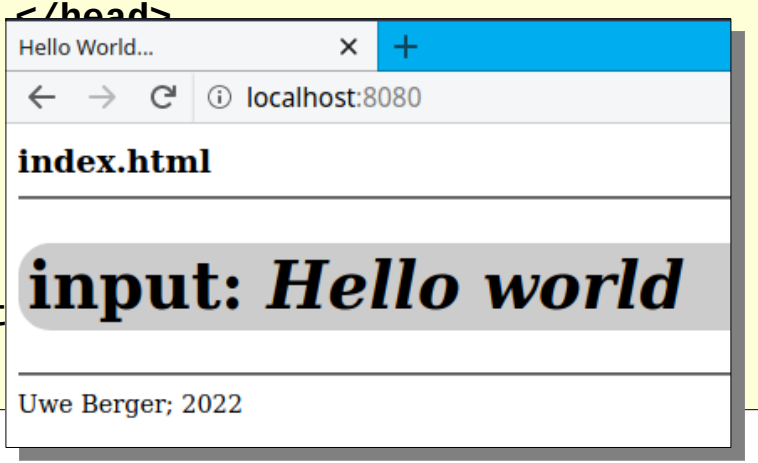
```
$def with (page)
<html>
    <head>
        <title>Hello World...</title>
        <style>
        ...
        </style>

    </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>

    </head>

</ht
```



„Hello World!“ (hello4.py)

hello4.py

- „Variablenübergabe“ via URL

„Hello World!“ (hello4.py)

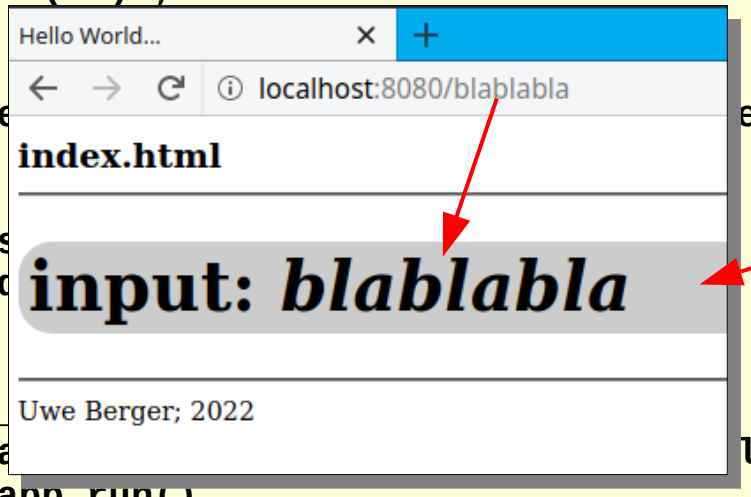
```
import web

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>
```

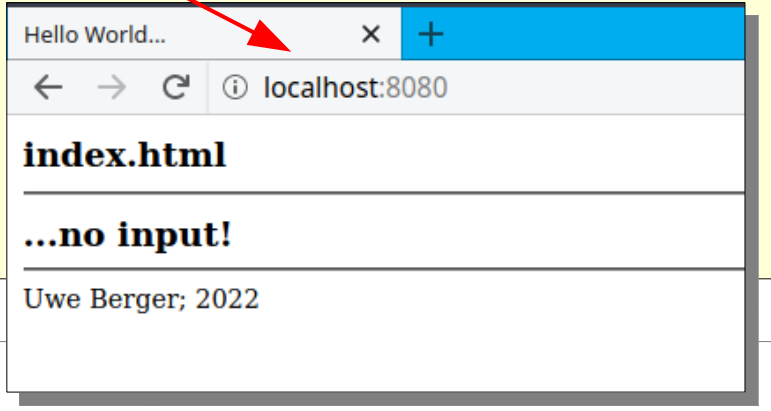
„Hello World!“ (hello4.py)

```
import web

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

rende                                    es/',


class
    c


if __
    a                                    ls())
    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

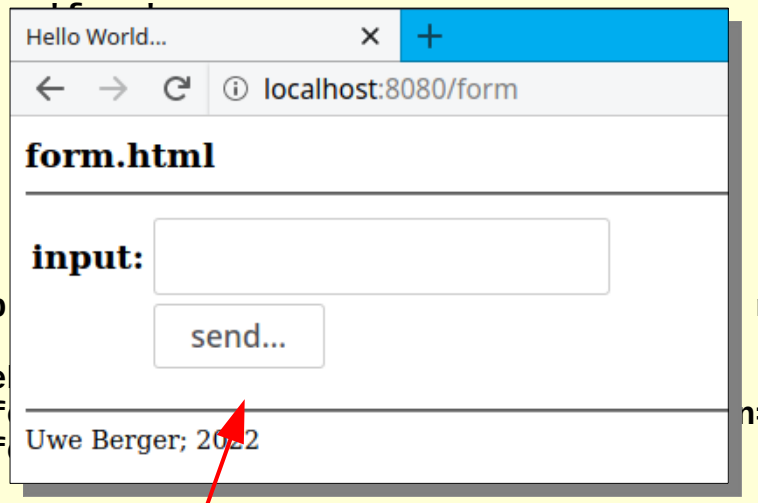
```
$def with (formular)

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

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

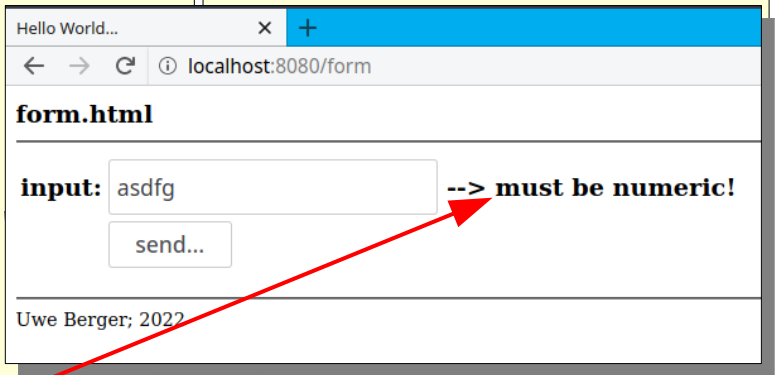
„Hello World“ (hello5.py)

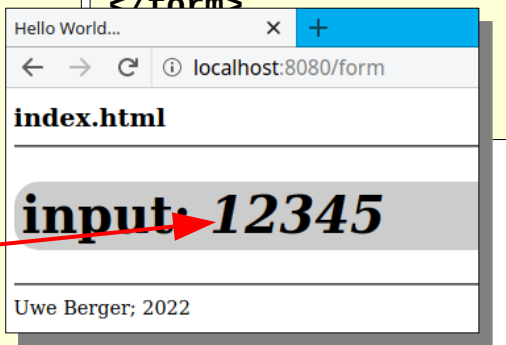
```
...
urls = ('/form'
        '/(.*)'
...
class index:
    ...

class form:

    valid = web                                    numeric!

    formular=we
          web.f                                    n="input:
          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)
...
```

templates/index.html

```
...
    $:formular.render()
</form>
```







„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)

<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)

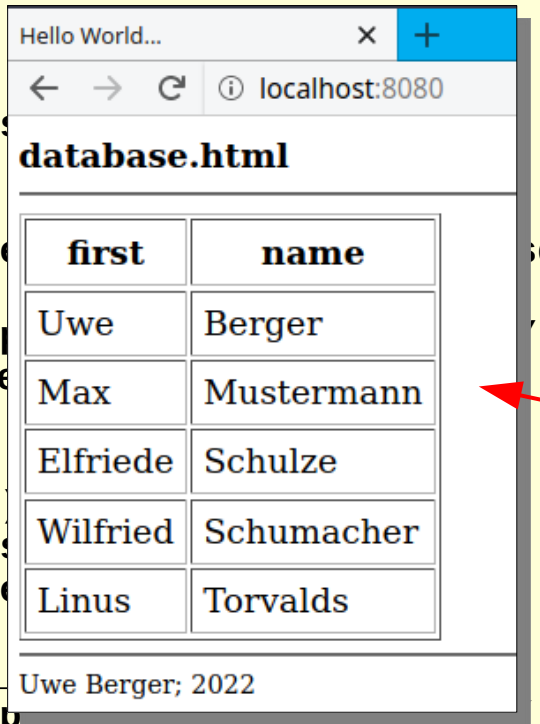
```
import web

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 = (
    '/', 'databa
)

db = web.databas                                    qlite')

render = web.tem                                    ',
             base

class database:
    def GET(self
        fn = db.
        return r

if __name__ == "
    app = web.app                                   ))
    app.run()
```

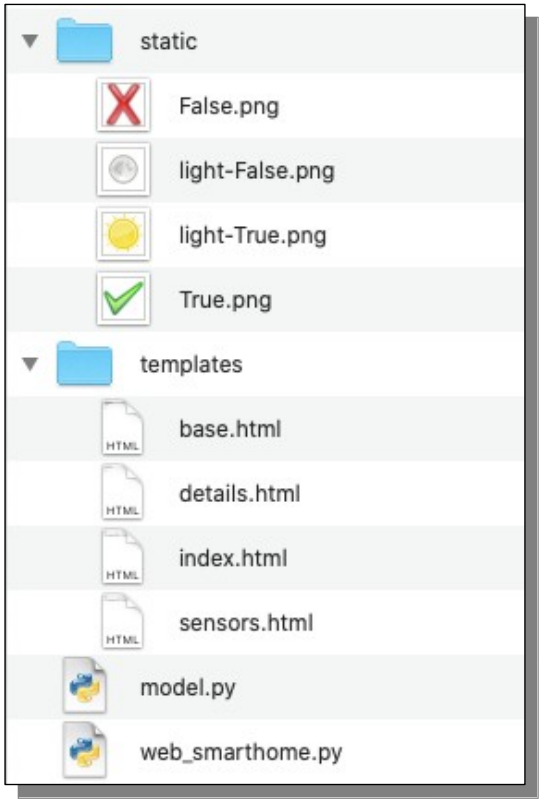


templates/database.html

```
$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>
```

(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“

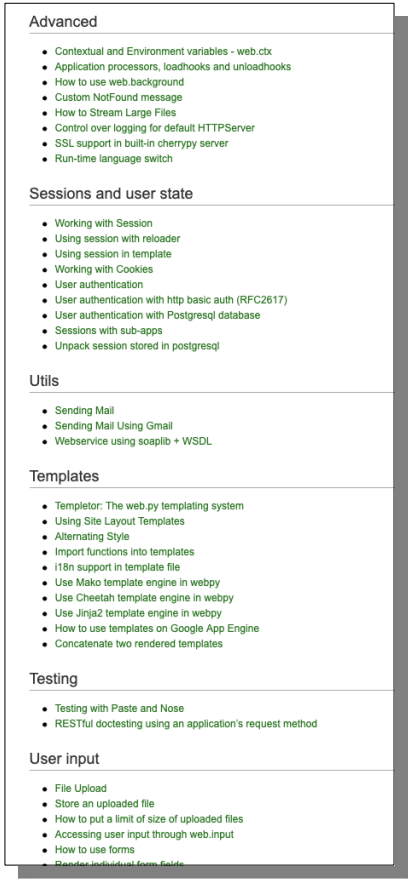


War das alles?

→ 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

- Templetor: 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 doctesting using an application's request method

User input

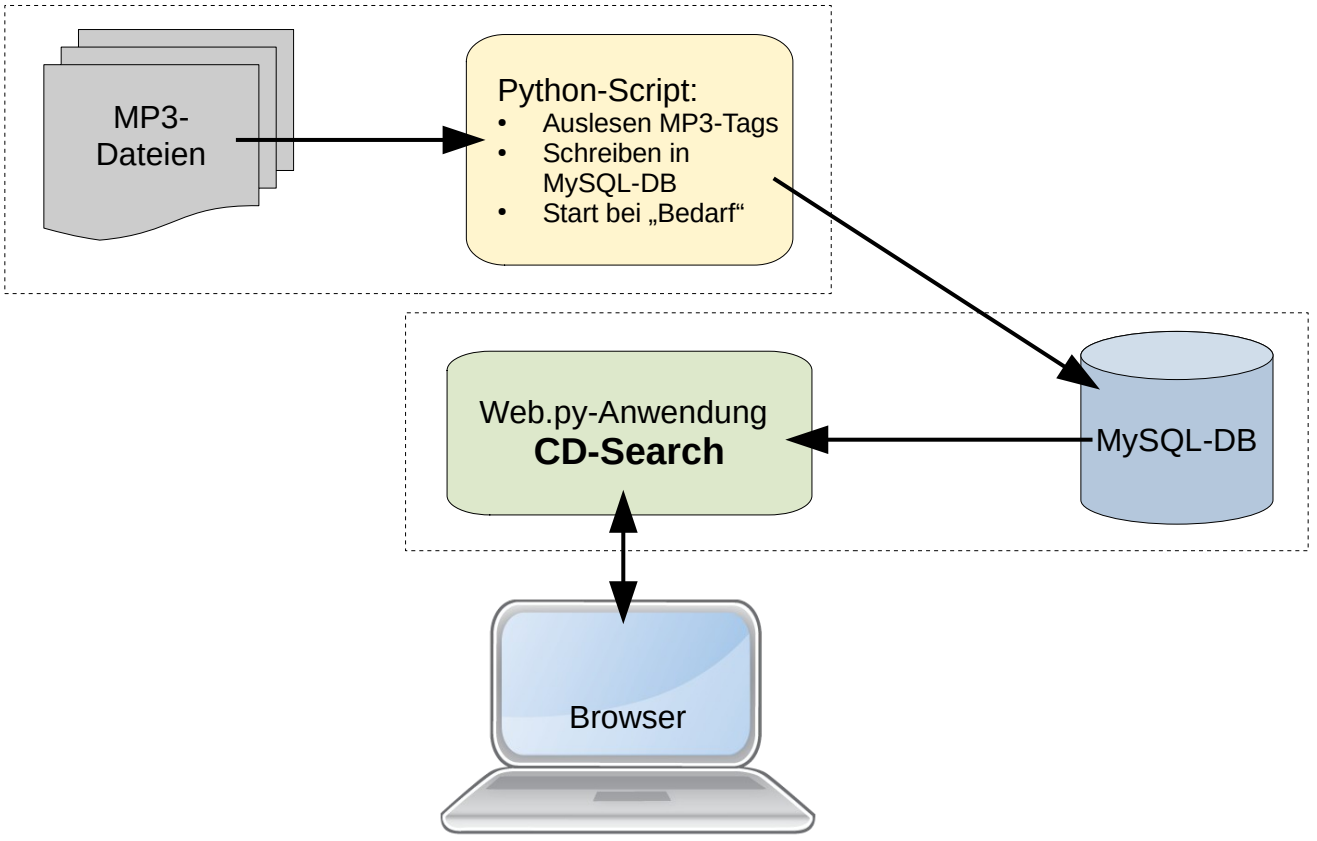
- File Upload
- Store an uploaded file
- How to put a limit of size of uploaded files
- Accessing user input through web.input
- How to use forms

Da war doch was… → CD-Search

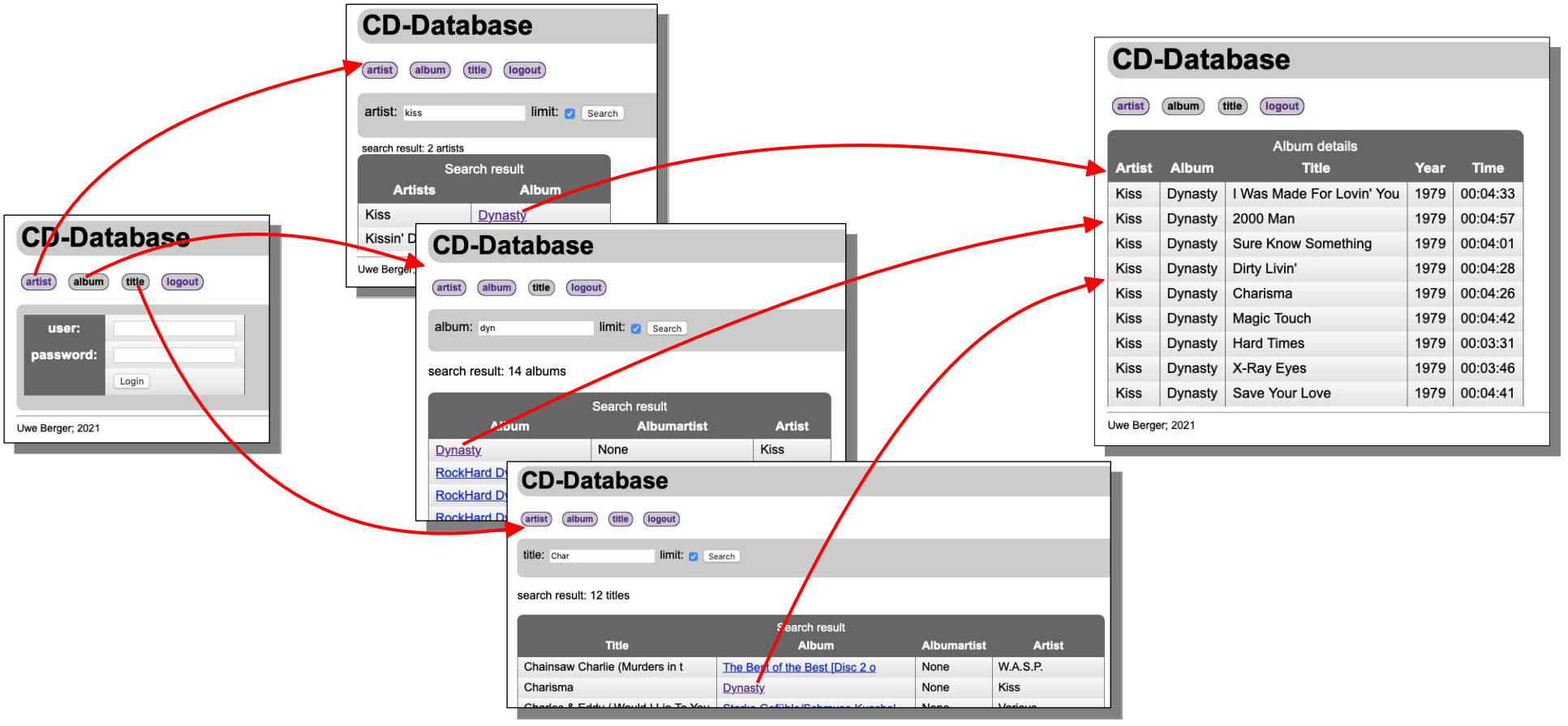
→ https://github.com/boerge42/webpy-apps/tree/master/web_cd_search

- 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)

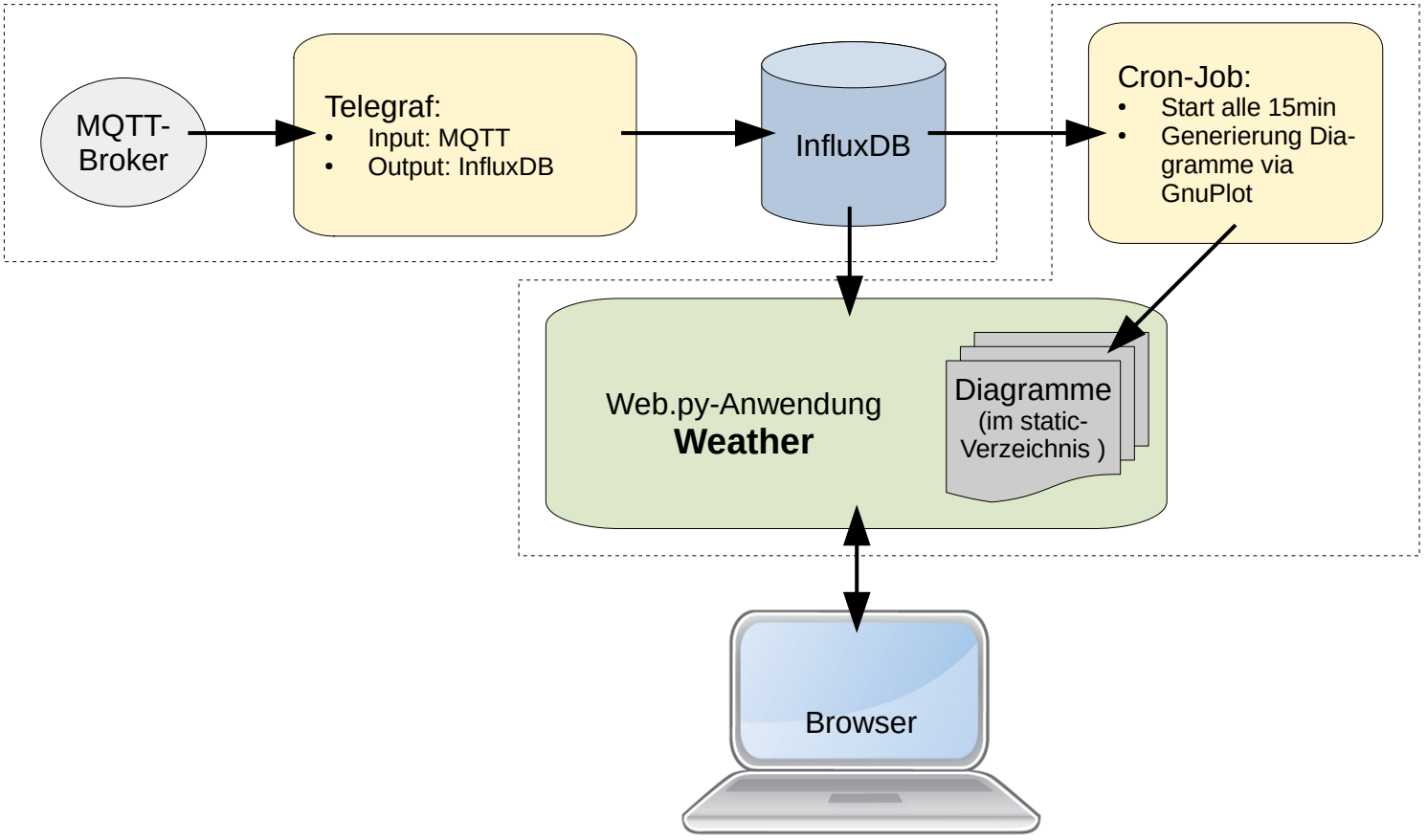


Web-Anwendung: Weather

→ https://github.com/boerge42/webpy-apps/tree/master/web_weather

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

Weather (Architektur)



Weather (Web-Oberfläche)

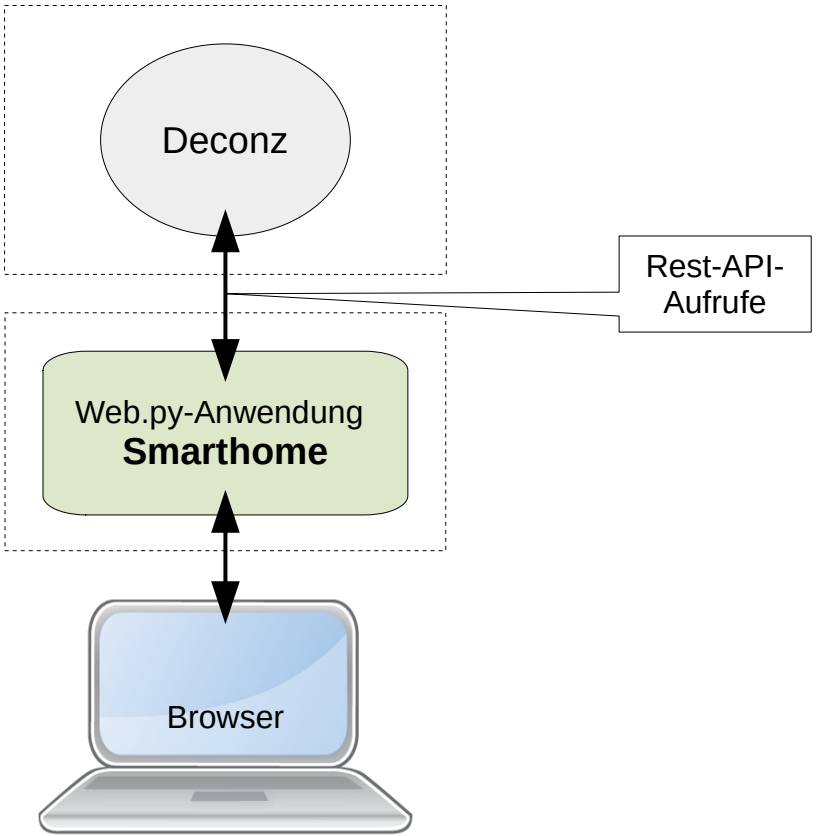
Web-Anwendung: Smart Home

→ https://programm.froscon.org/2021/events/2637.html
(Titel: „Ich wollte nie Smart Home machen…“)

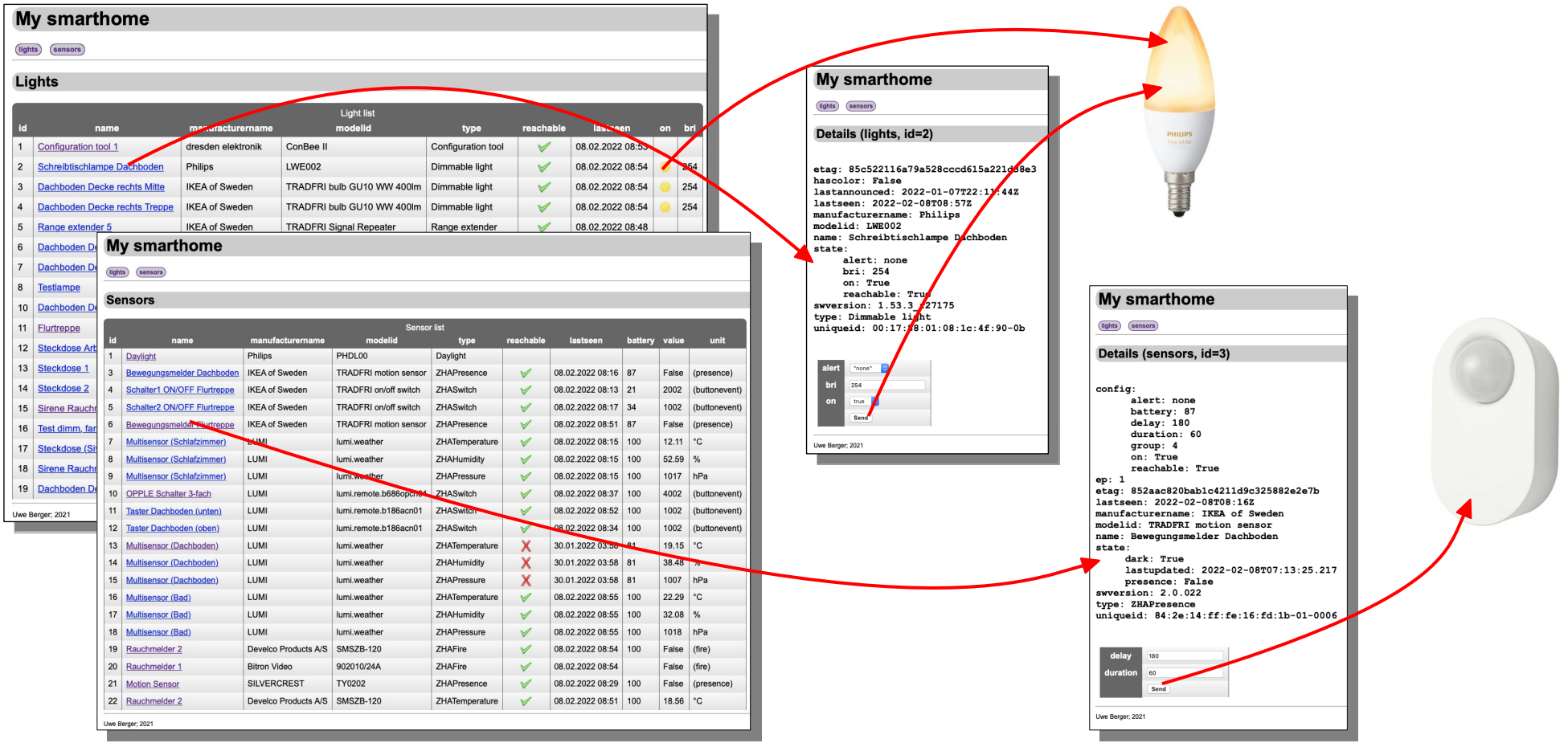
→ https://github.com/boerge42/webpy-apps/tree/master/web_smarthome

- Ziele:
  - keine Datenbank im Hintergrund; anzuzeigende Daten werden online via Rest-API-Aufrufe ermittelt/gesendet
  - dynamisierte Formulare
  - Validierung von Formulareingaben

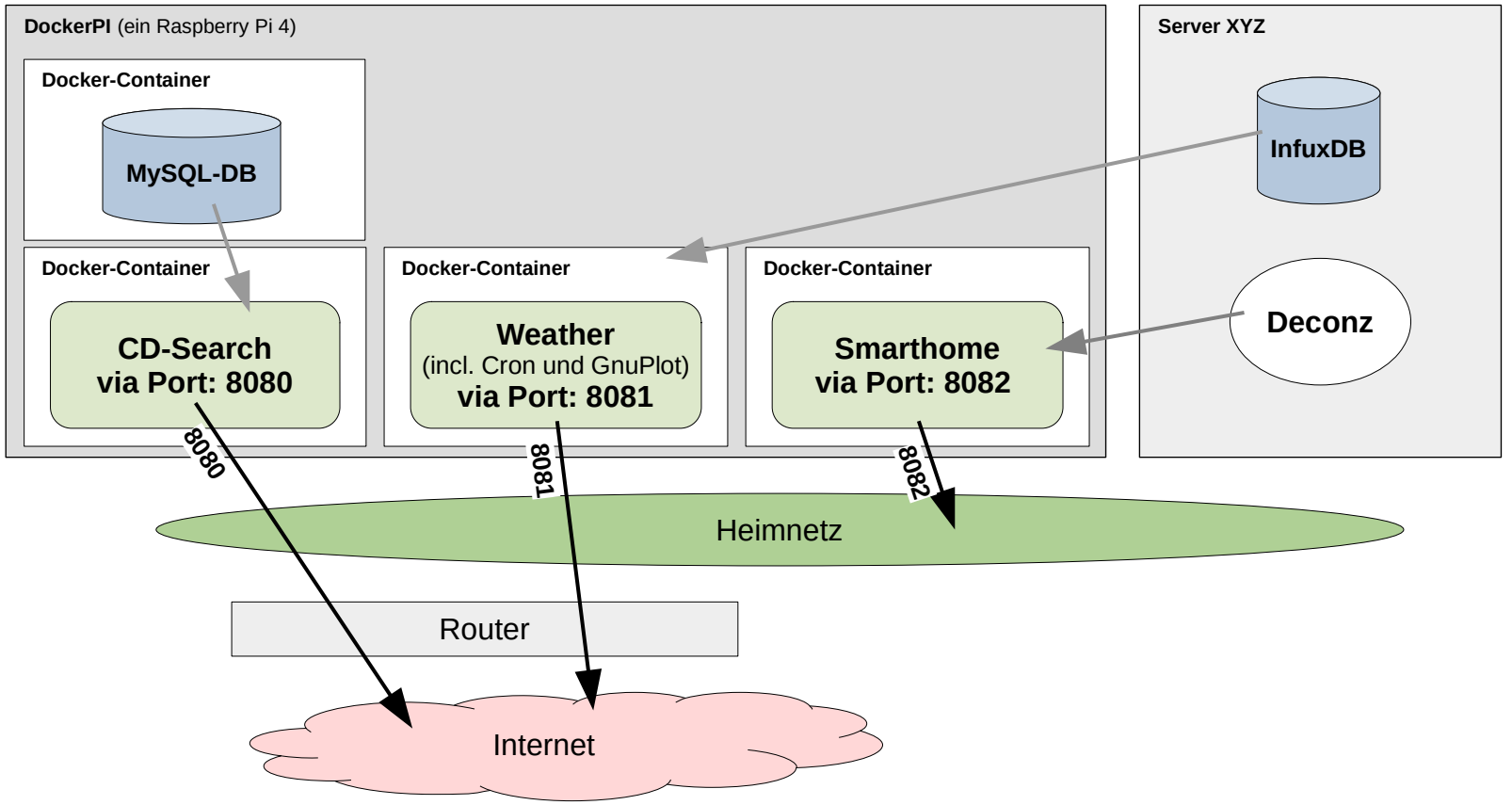
Smarthome (Architektur)



Smarthome (Web-Oberfläche)



...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!