Database Systems Lab
Architecture

Christian Rauch

(changed: August 17, 2025)

Overview

Frontend	HTML, CSS, JavaScript
$\Updownarrow$	HTTP, AJAX, REST
Backend	Python, Flask, WSGI
$\Updownarrow$	DB-API, mysql-connector-python
Database	MariaDB, SQL

Frontend / HTML

The head section contains meta data, scripts, and styles.

```html
<!doctype html>
<html>
  <head>
    <title>...</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles.css">
    <script src="common.js"></script>
  </head>
```

The body section contains the rendered and interactive elements.

```html
  <body>
    ...
  </body>
</html>
```

Use the browser debug function (F12) to inspect the DOM, scripts, styles, network traffic, and more.

Backend / HTML Request Handling

In Flask, you map routes (URL patterns) to functions.

```python
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/profile/<uid>')
def profile(uid): # load uid
  u = {"name": "Penny Low", "id": "c03929"}
  return render_template('profile.html', user=u)
```

The HTML template (‘profile.html‘) will be processed by Jinja2:

```html
<!doctype html>
<html>
  <head>
    <title>Profile: {{ user.id }}</title>
  </head>
  <body>
    <h1>Hello {{ user.name }}</h1>
  </body>
</html>
```

Frontend / AJAX

AJAX is used for asynchronous communication with the server without full page reloads.

```
fetch('/api/data')
  .then(response => {
    if (!response.ok) {
      throw new Error('...');
    }
    return response.json();
  })
  .then(data => {
    ... // process the loaded data
        // e.g., data.forEach(d => { ... })
        // e.g., console.log(data)
  })
  .catch(e => console.error('...', e));
```

Backend / AJAX Request Handling

Flask handles requests and responses on the server side:

```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/data', methods=['GET'])
def get_data():
  data = [
    {"name": "Mary Doe", "id": "d00221"},
    {"name": "Jill Woe", "id": "z09431"}
  ]
  return jsonify(data)

if __name__ == '__main__':
  app.run(debug=True)
```

Backend / Virtual Environment

Virtual environments isolate projects and allow to manage dependencies separately. This ensures predictable workflows and reproducible deployments.

Install venv using pip, create a virtual environment, and activate it.

```
$ python -m venv .venv
$ source .venv/bin/activate
```

Get a list of currently installed components.

```
$ python -m pip freeze
```

Install a list of components.

```
$ python -m pip install -r ./requirements.txt
```

Backend / Dependencies

File requirements.txt lists all direct dependencies (i.e., libraries) required by your system. You usually specify the exact versions.

```
flask==3.0.3
uwsgi==2.0.26
mysql-connector-python==9.0.0
...
```

Do not list indirect sub-dependencies (i.e., dependencies of your dependencies), this is redundant.

Environments are not portable, create them in place. The directory should be added to .gitignore.

```
**/__pycache__
.venv
```

Backend / Deployment (optional)

Install uWSGI using the package manager of your system.

```
sudo apt install uwsgi
sudo apt install uwsgi-plugin-python3
```

Create a WSGI configuration for your application.

```
[uwsgi]
plugins      = python3
socket       = 127.0.0.1:5010
chdir        = /srv/x
protocol     = http
wsgi-file    = app.py
callable     = app
processes    = 4
threads      = 2
buffer-size  = 32768
stats        = 127.0.0.1:9191
virtualenv   = /srv/x/.venv
http-timeout = 86400
uid          = chris
```

Backend / Deployment (optional)

Run the WSGI server with this configuration to test the setup.

```
uwsgi --ini /srv/x/myapp.ini
```

Open localhost:5010 in a browser and check if it works. Kill the process afterwards.

Copy the configuration to the apps-available directory and create a symbolic link in the apps-enabled directory.

```
sudo ln -s /etc/uwsgi/apps-available/myapp.ini
    /etc/uwsgi/apps-enabled
```

Enable and (re)start.

```
sudo systemctl enable uwsgi
sudo systemctl start uwsgi
```

You would typically install a reverse proxy (e.g., Nginx) and set it up to forward requests to WSGI server.

Conclusion

Thank you for your attention!