

MAXIN

The image shows the word "MAXIN" in a stylized, grey, sans-serif font. A bright yellow, curved swoosh underline starts under the 'M', loops under the 'A' and 'X', and ends under the 'N'. The letters are bold and have a slightly irregular, hand-drawn appearance.



Webjive

vincent.hardion@maxiv.lu.se

On behalf of the KITS group

Start with Why?

Web based tool for control system.

User-friendly interactions.

Easy and quick-to-use.

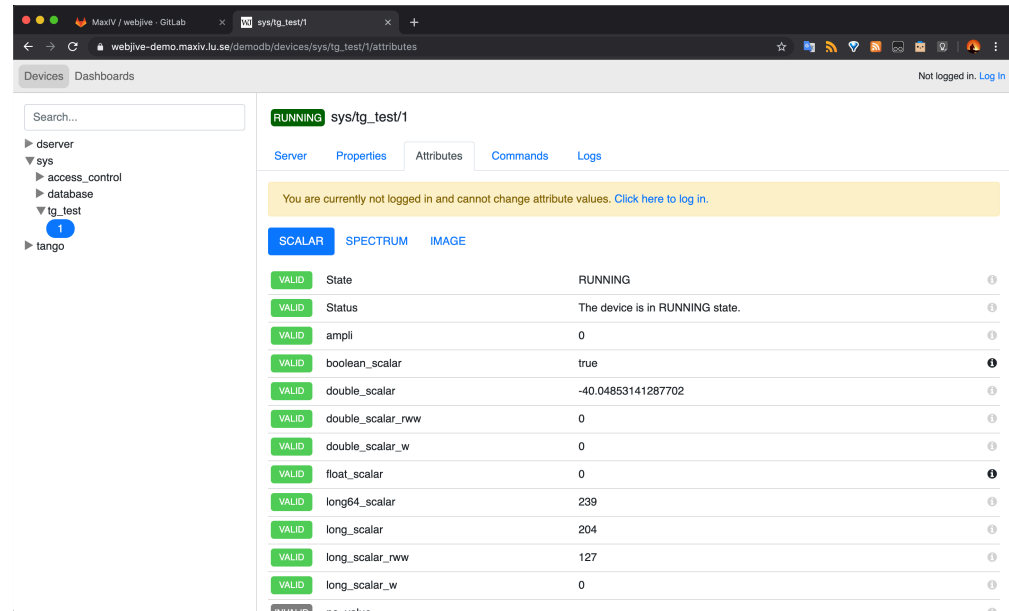
Monitoring as well as control.

WebJive is not Jive

WebJive is a Device explorer built on TangoGQL

With WebJive you can:

- View a list of all Tango devices
- View and modify device properties
- View and modify device attributes
- View and execute device commands
- Create web interfaces for interacting with Tango devices (on /<tangoDB>/ dashboard)



The screenshot shows a web browser window with the URL `webjive-demo.maxiv.lu.se/demodb/devices/sys/tg_test/1/attributes`. The interface displays the following information:

- Device: **sys/tg_test/1** (State: RUNNING)
- Navigation tabs: Server, Properties, Attributes (selected), Commands, Logs
- Message: "You are currently not logged in and cannot change attribute values. [Click here to log in.](#)"
- Attribute list (all are VALID):

Attribute Name	Value
State	RUNNING
Status	The device is in RUNNING state.
ampli	0
boolean_scalar	true
double_scalar	-40.04853141287702
double_scalar_rww	0
double_scalar_w	0
float_scalar	0
long64_scalar	239
long_scalar	204
long_scalar_rww	127
long_scalar_w	0

The Devices View

- Configuration of all the devices in the control system in treelike hierarchy.
- All attributes, commands and properties.
- Automatic detection for inputs.
- Search bar for devices.

The screenshot displays the 'Devices View' interface. On the left, a tree-like hierarchy shows a search bar and a list of devices under the 'b107a' category, including various sub-identifiers like 'b107a-d100730' and 'b107a-eb01'. On the right, the detailed view for a device is shown, with tabs for 'Server', 'Properties', 'Attributes', 'Commands', and 'Logs'. The 'Attributes' tab is active, displaying a table of device attributes. The table has columns for attribute names and their values. The status of the device is 'RUNNING'.

Attribute	Value
State	RUNNING
Status	The device is in RUNNING state.
ampli	0
boolean_scalar	true
double_scalar	0.6560585728221919
double_scalar_rww	167.95099464248113
double_scalar_w	0
float_scalar	0
long64_scalar	5
long_scalar	26
long_scalar_rww	64
long_scalar_w	0
no_value	
short_scalar	105

Open The WebJive Demo

- Use Chrome or Firefox
- Go to <https://webjive-demo.maxiv.lu.se/demodb>
- search for tg_test
- read the attributes double_scalar, double_spectrum_ro
- set the attribute ampli to 20
- ;-{
- log in with demo/demo
- set the attribute ampli to 20
- execute the commands SwitchStates, DevDouble

Installation

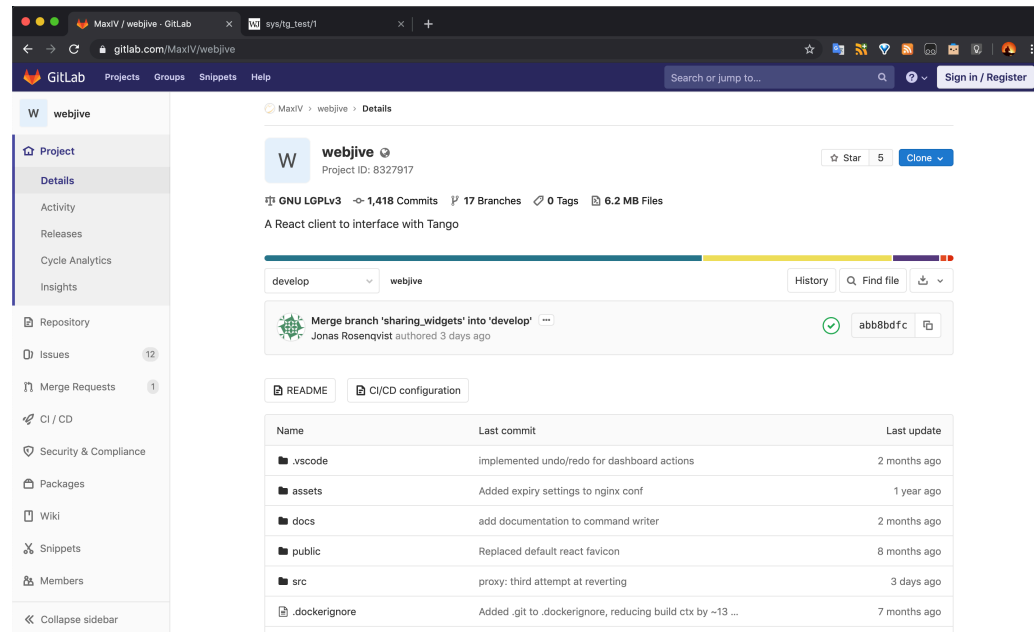
Clone the repository.

```
$ npm install
```

```
$ npm start
```

Minimum node version:
7.6 (introduced `async/`
`wait`)

Verified working node
version: **9.11.2** (currently
used by the `dockerfile`)

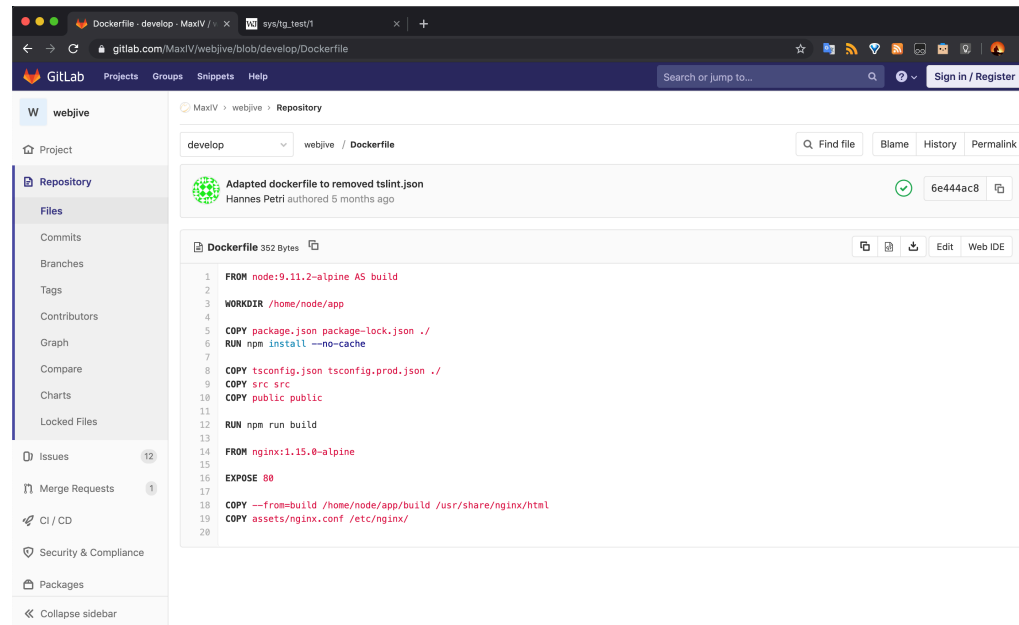


The screenshot shows the GitLab web interface for the 'webjive' project. The project details include the name 'webjive', Project ID '8327917', and a description 'A React client to interface with Tango'. The interface shows a commit history table with columns for Name, Last commit, and Last update.

Name	Last commit	Last update
■ .vscode	implemented undo/redo for dashboard actions	2 months ago
■ assets	Added expiry settings to nginx conf	1 year ago
■ docs	add documentation to command writer	2 months ago
■ public	Replaced default react favicon	8 months ago
■ src	proxy: third attempt at reverting	3 days ago
📄 .dockerignore	Added .git to .dockerignore, reducing build ctx by ~13 ...	7 months ago

Easy with Docker

docker-compose build
docker-compose run



The screenshot shows a GitLab repository page for a project named 'webjive'. The file 'Dockerfile' is selected, showing its content. The Dockerfile is a multi-stage build for a Node.js application with an Nginx reverse proxy. The build process includes installing dependencies, running tests, and building the application. The final stage is an Nginx container that serves the application's static assets.

```
1 FROM node:9.11.2-alpine AS build
2
3 WORKDIR /home/node/app
4
5 COPY package.json package-lock.json ./
6 RUN npm install --no-cache
7
8 COPY tsconfig.json tsconfig.prod.json ./
9 COPY src src
10 COPY public public
11
12 RUN npm run build
13
14 FROM nginx:1.15.0-alpine
15
16 EXPOSE 80
17
18 COPY --from=build /home/node/app/build /usr/share/nginx/html
19 COPY assets/nginx.conf /etc/nginx/
20
```

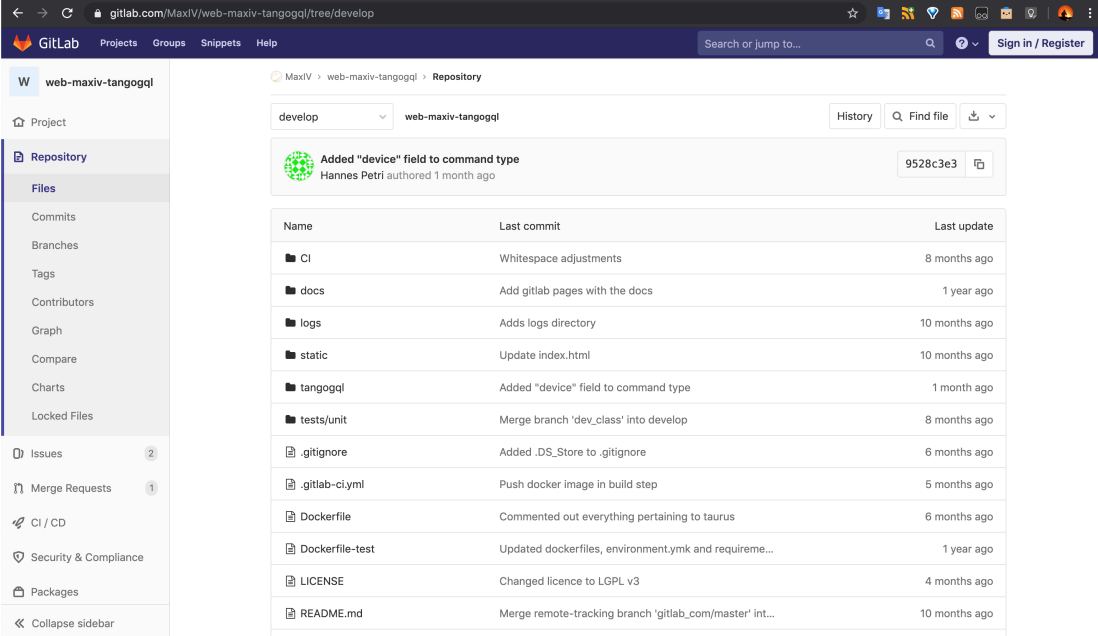

But it needs TangoGQL

WebJive is unable to speak with Tango

WebJive speaks only GraphQL

And TangoGQL translate Tango to GraphQL...

that's good timing.



The screenshot shows the GitLab interface for the repository 'web-maxiv-tangogql'. The left sidebar contains navigation options: Project, Repository, Files, Commits, Branches, Tags, Contributors, Graph, Compare, Charts, Locked Files, Issues (2), Merge Requests (1), CI / CD, Security & Compliance, Packages, and Collapse sidebar. The main content area displays the commit history for the 'develop' branch. A recent commit by Hannes Petri is highlighted, with the message 'Added "device" field to command type' and commit ID '9528c3e3'. Below this is a table of commit history.

Name	Last commit	Last update
CI	Whitespace adjustments	8 months ago
docs	Add gitlab pages with the docs	1 year ago
logs	Adds logs directory	10 months ago
static	Update index.html	10 months ago
tangogql	Added "device" field to command type	1 month ago
tests/unit	Merge branch 'dev_class' into develop	8 months ago
.gitignore	Added .DS_Store to .gitignore	6 months ago
.gitlab-ci.yml	Push docker image in build step	5 months ago
Dockerfile	Commented out everything pertaining to taurus	6 months ago
Dockerfile-test	Updated dockerfiles, environment.yml and requirem...	1 year ago
LICENSE	Changed licence to LGPL v3	4 months ago
README.md	Merge remote-tracking branch 'gitlab.com/master' int...	10 months ago

GraphQL API call example

```
{
  devices(pattern: "sys/tg_test/1"){
    attributes(pattern: "ampli"){
      name
      device
      datatype
      dataformat
      writable
      label
      unit
      description
      displevel
      value
      quality
      minvalue
      maxvalue
      minalarm
      maxalarm
    }
  }
}
```

HTTP
GET



```
{
  "data": {
    "devices": [ {
      "attributes": [ {
        "name": "ampli",
        "device": "sys/tg_test/1",
        "datatype": "DevDouble",
        "dataformat": "SCALAR",
        "writable": "WRITE",
        "label": "ampli",
        "unit": "test",
        "description": "No description",
        "displevel": "OPERATOR",
        "value": 80,
        "quality": "ATTR_VALID",
        "minvalue": null,
        "maxvalue": 120,
        "minalarm": 0,
        "maxalarm": 92
      }
    ]
  }
}
```

Schema and strong type definition based. Act as a contract between back and front end.

Installation

Clone the repository.

```
$ pip install -r requirements.txt
```

```
$ python -m tangogql
```

Conda environment can be created using the `_environment.yml_`.

Dockerfile is provided and can be used to run the server

One command installation (!bandwidth)

Requirements

- Make
- git
- Docker
- docker-compose

```
$ git clone https://gitlab.com/MaxIV/webjive-develop.git
```

```
$ cd webjive-develop
```

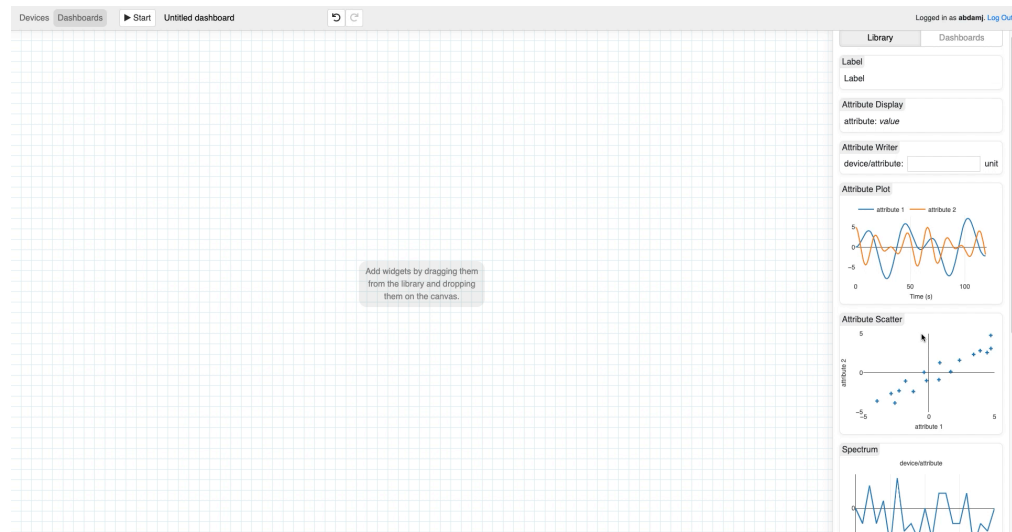
```
$ make
```

```
...
```

```
$ make run
```

Dashboards

- Customizable and shareable views.
- Drag-and-drop from default widgets and connect them to devices and/or attributes.
- Edit mode and run mode.



Create a Dashboard

Login as demo/demo

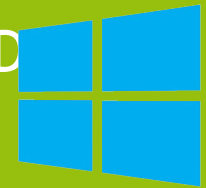
Create a dashboard in WebJive

With One:

- Label
- Attribute Display pointing to the device “sys/tg_test/1” and the attribute “ampli”
- Attribute Writer pointing to the device “sys/tg_test/1” and the attribute “ampli”
- Attribute Plot pointing to the device “sys/tg_test/1” and the attributes “ampli” & “double_scalar”
- Attribute Scatter pointing to the device “sys/tg_test/1” and the attributes “ampli” & “double_scalar”
-

Authentication and persistence

- Authentication and authorization through JSON web tokens (JWT) with AD as source of information.
- Persistence like saving of dashboards using Mongo DB
- Logging for all the mutations in the Tango DB.



Active Directory



Recent user actions Showing the latest entries

Time	User	Device	Name	Action	Additional info
2019-10-03 08:47:20.729	abdamj	sys/tg_test/1	boolean_scalar	Attribute value changed	Value before: true. Value after: false. Current value: false
2019-10-03 08:45:54.715	abdamj	sys/tg_test/1	boolean_scalar	Attribute value changed	Value before: false. Value after: true. Current value: true
2019-10-03 08:43:27.827	abdamj	sys/tg_test/1	boolean_scalar	Attribute value changed	Value before: true. Value after: false. Current value: false
2019-10-03 08:42:48.759	abdamj	sys/tg_test/1	ampli	Attribute value changed	Value before: 1. Value after: 0. Current value: 0

Deployment

- All services within package are containerized.
 - Authentication, Mongo DB, Frontend, Backend
- CI/CD through Ansible.
- Traefik: reverse-proxy between frontend and backends for Tango Databases.
 - Accessible on webjive.maxiv.lu.se (internally)

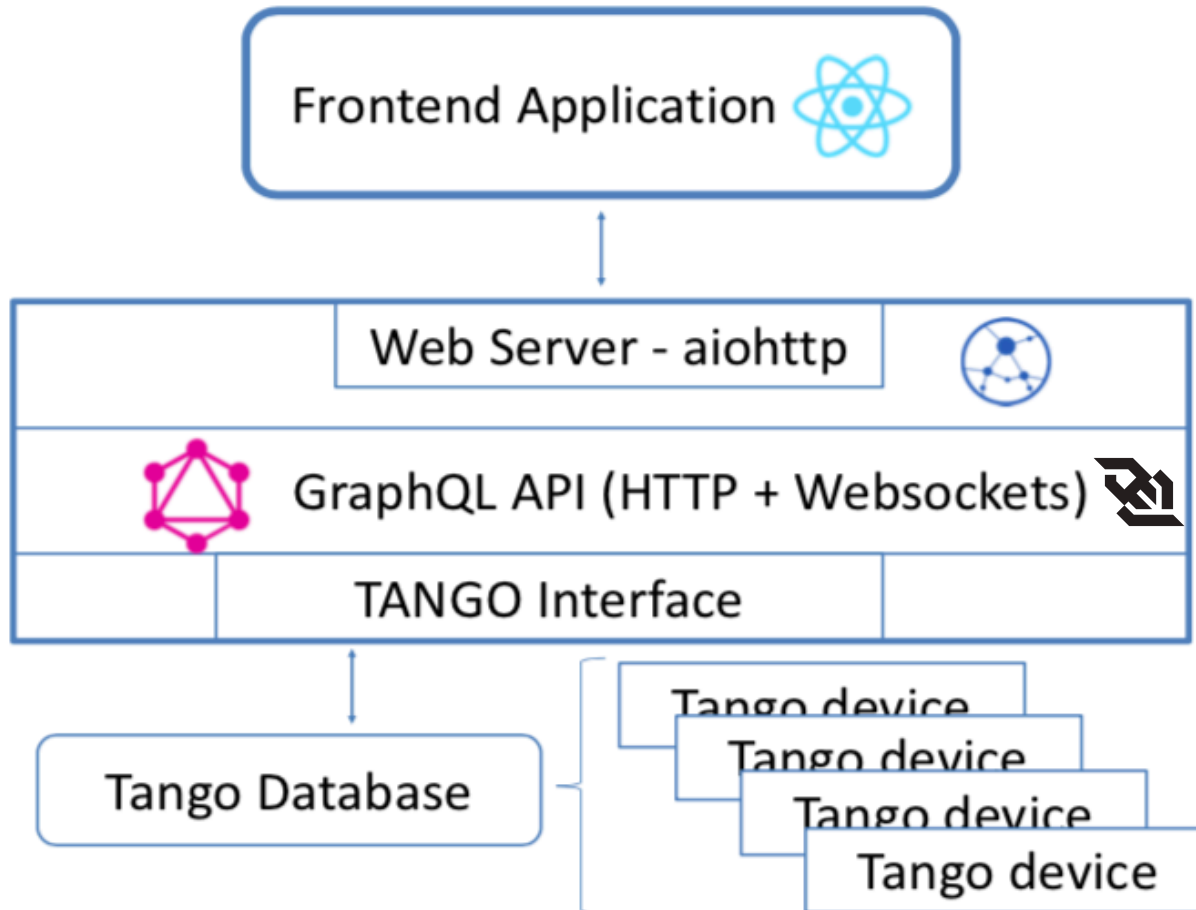


ANSIBLE



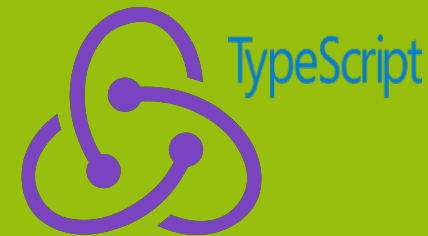
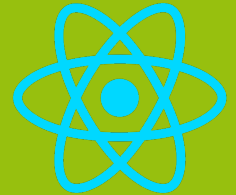
docker

Architecture



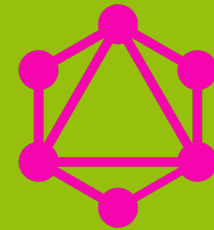
Frontend

- Built using React, Redux, Typescript.
- Bunch of utility libraries for API calls, plotting etc.
- Follows most-recent development techniques and guidelines in React world.
- Updates and maintenance are expected to be quick and pain-free.
- Tango attribute subscriptions, event-based, auto-updates using websockets.
- Two root level-views; Devices and Dashboards.



Backend

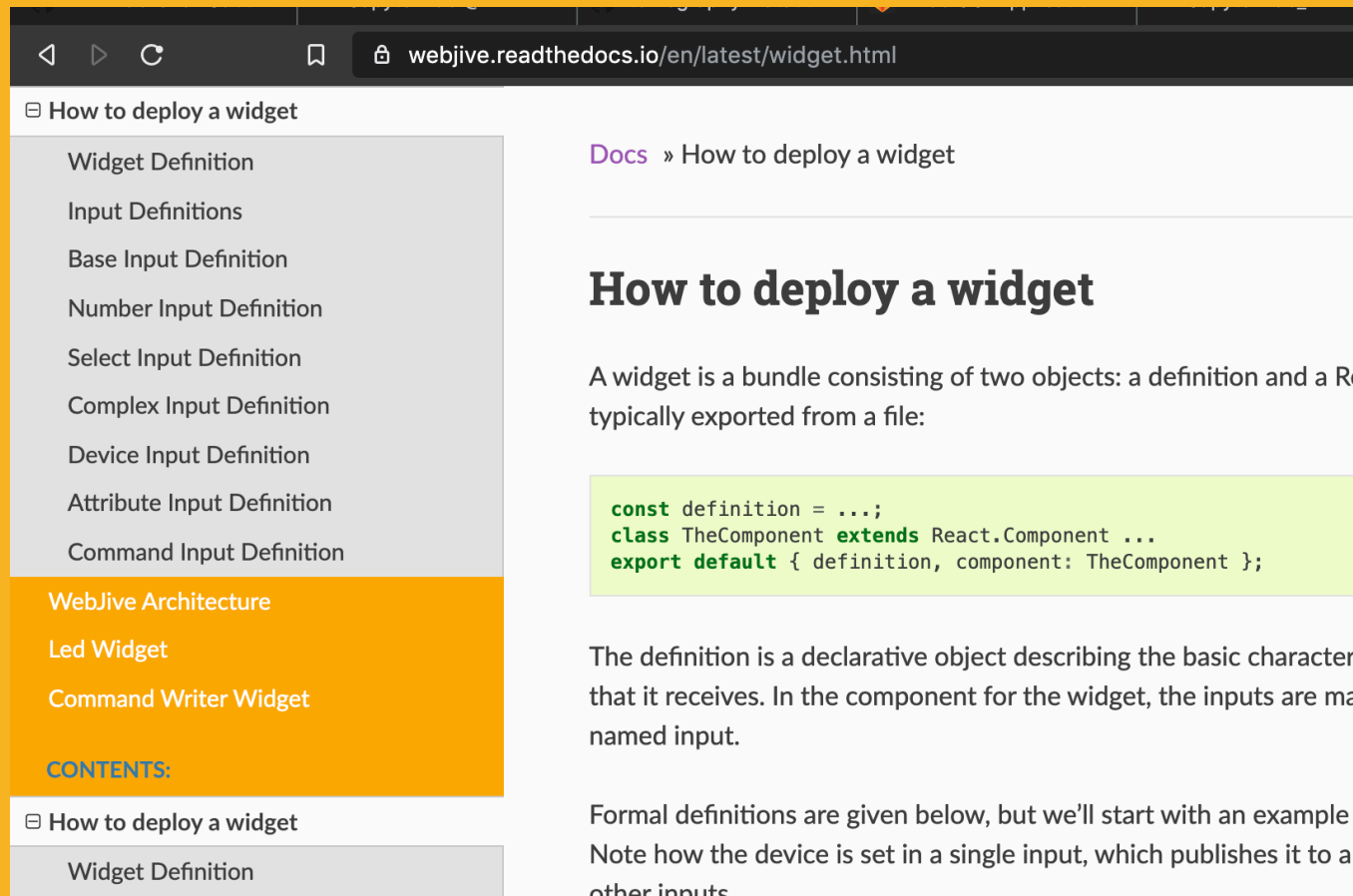
- Aiohttp server with GraphQL API.
- Graphene, python library for GraphQL implementation.
- Client asks explicitly what he needs reducing number of calls.
- One call for nested data instead of multiple calls on multiple end-points (next slide).
- Query: fetch data via resolvers.
- Mutation: create, update and delete.
- Subscription: real-time connection with the server, events from tango through websockets.



For developers

How to create a widget

<https://webjive.readthedocs.io/en/latest/widget.html>



The screenshot shows a web browser window with the address bar displaying `webjive.readthedocs.io/en/latest/widget.html`. The page content includes a navigation sidebar on the left with a tree view of the documentation structure. The main content area shows the title 'How to deploy a widget' and a brief introduction: 'A widget is a bundle consisting of two objects: a definition and a React component, typically exported from a file:'. Below this is a code block with the following JavaScript code:

```
const definition = ...;
class TheComponent extends React.Component ...
export default { definition, component: TheComponent };
```

The text continues: 'The definition is a declarative object describing the basic characteristics of the widget that it receives. In the component for the widget, the inputs are mapped to the widget's named input.' The final sentence is partially visible: 'Formal definitions are given below, but we'll start with an example. Note how the device is set in a single input, which publishes it to a set of other inputs.'

Current Developments and Future

Improving event-subscriptions.

Extending widget library.

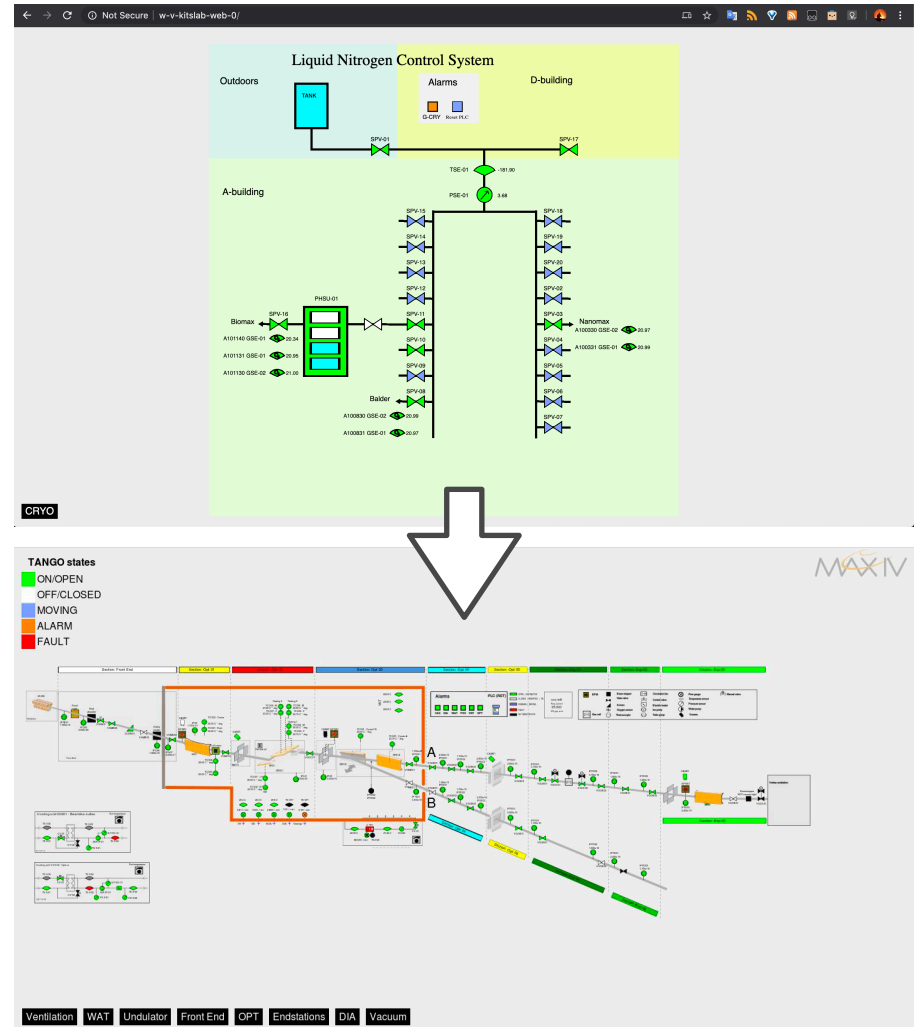
Especially for Image Attributes.

Synoptic View.

Central logging for user actions in Elasticsearch.

Group-editable dashboards.

Performance testing.



Conclusion

WebJive is a device tree

WebJive is a dashboard

WebJive will be a synoptic

WebJive is not only an application,

WebJive is first of all an
architecture.

Acknowledgement

- Abdullah Amjad
- Jonas Rosenqvist
- Antonio Milan Otero
- Antoine Dupré
- Mikel Eguiraun
- Emil Rosendahl
- Fredrik Bolmsten
- Hannes Petri
- Johan Forsberg
- Linh Nguyen
- Matteo Canzari
- Helder Ribeiro
- Mark Nicol
- Ralph Braddock