ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH3003

WALTZ – A PLATFROM FOR TANGO CONTROLS WEB APPLICATIONS

I. Khokhriakov†, F. Wilde, Helmholtz-Zentrum Geesthacht, Geesthacht, Germany
O. Merkulova, IK company, Moscow, Russia

Abstract

The idea of creating Tango Controls web platform was born at Tango Users Meeting in 2013, later a feature request was defined aka v10 roadmap FR6 – to provide a generic web application for browsing and monitoring Tango Controls devices. The work started in 2017. Name "Waltz" was selected by voting at 32nd Tango Users Meeting, Prague, Czech Republic in 2018. Waltz is the result of joint efforts of Tango Community, Helmholtz-Zentrum Geesthacht and IK company.

This paper gives an overview of Waltz as a platform for Tango Controls web applications, the overall framework architecture and presents an end result of real-life applications used in HZG. The work shows that having Waltz platform web developer can intuitively and quickly create full web application for his/her needs. Different architectural layers provide maintainability. The platform has a number of abstractions and ready-to-use widgets that can be used by web developer to quickly produce web based solutions. Among Waltz features are user context saving, device control and monitoring, plot and drag-n-drop interface solutions. Communication with Tango Controls happens via Tango REST API using HTTP/2.0 and Server-Sent Events. Waltz can be also treated as a system for device monitoring and control from any part of the world.

INTRODUCTION

Waltz is a general purpose Tango Controls web application that provides the interface between the Tango Control system and the scientific users who define and calibrate their experiments. It can also be used for live monitoring of a big scientific installations like ESRF or DESY.

Initially, the idea of Waltz (ex. TangoWebapp) was born at the 29th Tango Users Meeting, Krakow, Poland in 2015 [1]. It was marked as Feature Request #6 within future Tango Controls evolution road-map [2-3]. Final project name "Waltz" was chosen by the community at the 32nd Tango Users Meeting, Prague, Czech Republic in 2018.

Waltz was designed to be used in two ways – as a platform and as an end-user application. In this article we are going to describe Waltz as a platform for Tango Controls web applications and will to list some of the features from the end-user application to show to possibilities of the platform.

WALTZ AS A DEVELOPMENT PLATFORM

Waltz considered as a platform that can be used to implement integrated and coherent web based GUIs for Tango Controls [4].

Waltz's flexibility is achieved by a layered architecture: internal event bus (e.g., OpenAjax [5]) and widgets (e.g., UI components) which can be used as building blocks for rich functionality.

Waltz offers developers a number of compact APIs: tango device model API; reusable functional components (e.g., mixins); UI builder API. These will be covered in more details later in this section.

Integration with SVG files [6] allows to implement extremely user-friendly widgets.

Subscriptions allow Waltz to use Server-Sent Events [7-8] to receive notifications about native Tango events with minimal overhead.

Since v0.7 [9] Waltz fully supports JS6 [10] features. So developers are able to use cutting edge features of modern JavaScript to implement required widgets.

Layered Architecture

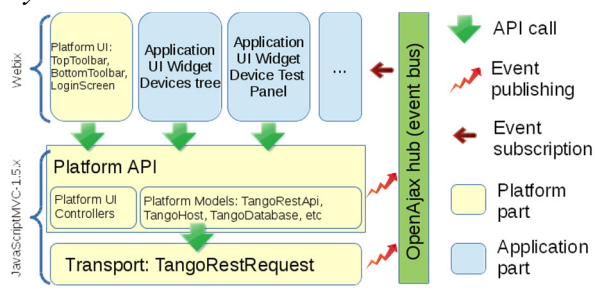

Figure 1: Waltz architecture layers.

The platform has 3 layers (from bottom to top): transport; models; UI (Fig. 1). Lower layers don't know anything about layer(s) on top and they send events via OpenAjax hub. Higher layers talk to the lower ones via API calls.

The Waltz platform is divided into *API* and *UI* parts. *Platform UI* is implemented following the concept of *smart components* – rich components build on top of used in the platform JS framework components (Webix components) [11]. Each UI component is completely standalone and can be used as a building block for more complex widgets or even dedicated applications. Using UI/non-UI components developer creates new UI components for specific use cases. *Platform API*s is conceived for building UI, errors handling, interacting with Tango device model, etc (Fig. 2).

Another perspective of the layered architecture of Waltz is shown in Fig. 2. Different UI/non-UI components communicate with each other via OpenAjax event bus which is a part of JavaScriptMVC framework [12]. Non-UI components or API components are responsible for common functionality of the platform, e.g., storing user context data.

† igor.khokhriakov@hzg.de

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH3003

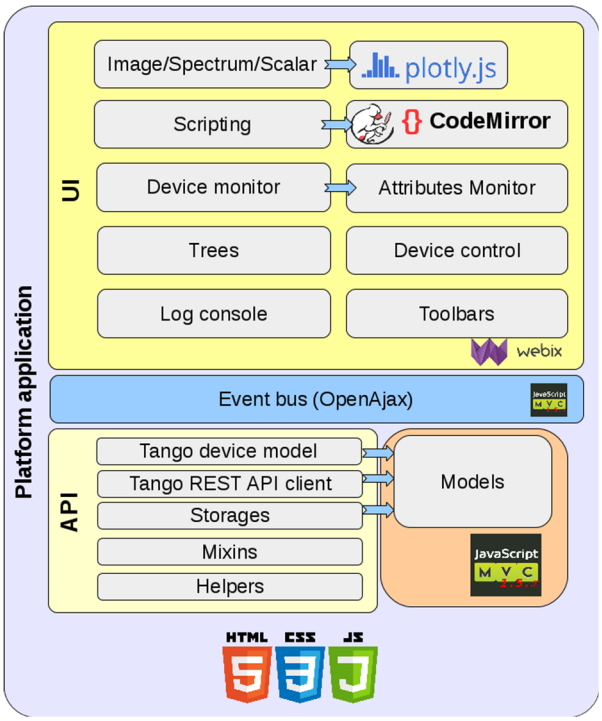

Figure 2: The platform offers a number of UI/non-UI APIs. This figure also shows external dependencies.

Components and Connectors

In this section we will observe how Waltz is implemented from Components and Connectors view of software architecture.

There are three main groups of components: UI (e.g., widgets), State/Context components and backend related components (Fig. 3).

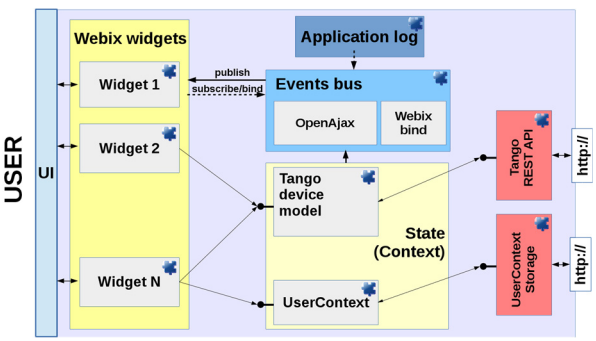

Figure 3: Waltz Components and Connectors diagram.

Backend related components communicate with backend services such as UserContext, Terminal emulator or Tango REST API server. *State/Context components* are responsible for holding the current state of the different entities, such as Tango Attribute info or Tango Command output etc. *UI components* are to represent distinct UI elements. They use direct calls to communicate with State/Context components to perform actions. These actions in their turn may affect state/context. UI components listen to state/context change events via event bus. New states are published via event bus (Fig. 4).

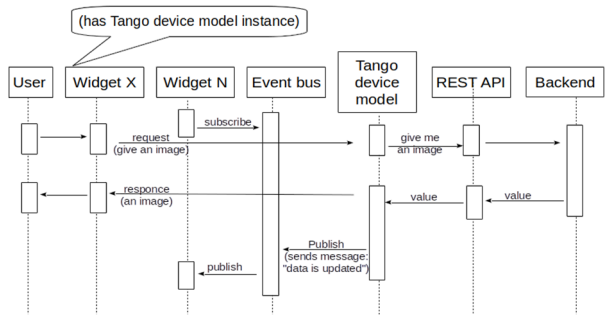

Figure 4: Result of the user's interaction with Widget X is propagated to other widgets via event bus.

Waltz Platform APIs

Waltz comes with a wide range of APIs. Here we present an overview of the most common APIs and those which make Waltz being a platform.

UI Builder API. This API is used to develop Waltz application UI layout, see Fig. 5. Using this API developer may define which widgets and content will be available in which panels/ views. To show the simplicity of development a part of client code of this API is below:

```
function buildUI(platform_api) {

const ui_builder = platform_api.ui_builder;

ui_builder.add_left_sidebar_item({"header":
 "<span.../> Devices" ...});

ui_builder.add_left_sidebar_item(

TangoWebapp.ui.newDeviceTree
(platform_api.context));

ui_builder.add_mainview_item({"header":
"<span.../> Dashboard" ...});

… }
```

PlatformContext, UserContext, Mixins etc. Waltz platform APIs offer a number of convenience and utility functions. For example, a single entry point to the application's state – *PlatformContext* has references to application data and the UserContext API. *UserContext* API allows developers to store user specific data from a widget (i.e. settings). *Mixin* [13] is a way to enrich JS object's functionality by using common functions. The platform provides a number of useful mixins such as Stateful (which allows a widget to persist its state), OpenAjaxListener (which allows widgets to subscribe to the event bus). More details and functions can be found in the project's documentation [14], also in [15].

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH3003

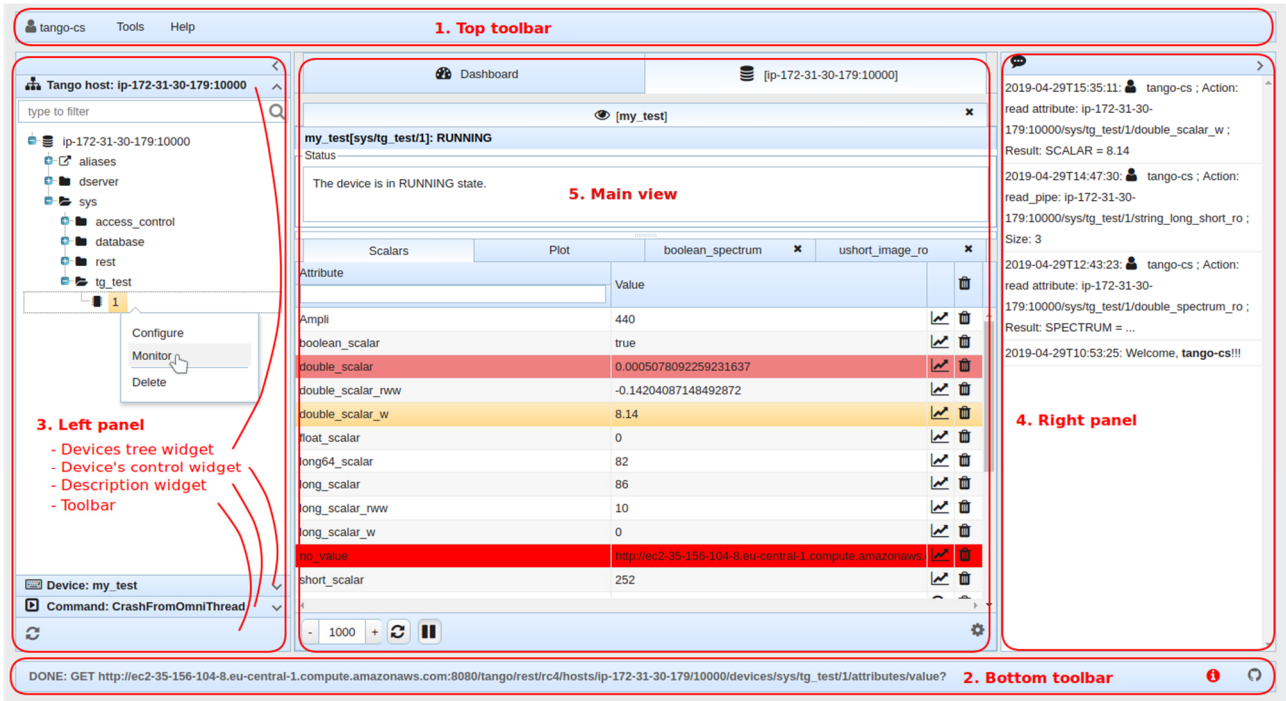

Figure 5: Waltz UI layout: 1 – top toolbar; 2 – bottom toolbar; 3 – left side panel; 4 – right side panel; 5 – main view.

Tango Device Model API is an abstraction of the Tango device model on top of the Tango REST API. It provides a high level client API familiar to Tango developers. Being a JavaScript API, it is based on Promise API [16], hence, allowing a convenient modern JavaScript way of programming. Using JavaScript 6 **async/await** [17] the client code becomes straightforward and clean. Here is an example of the Waltz platform Tango Device Model API in action:

```
const host = await
PlatformContext.rest.fetchHost
('localhost:10000');

const device = await host.fetchDe-
vice('sys/tg_test/1');

const attr = await device.fetchAttr('ampli');

const response = await attr.write(Math.ran-
dom());

return response.value;
```

Widget API allows developers to either create new widgets from scratch or re-use existing ones. Widget API is based on webix protoUI [18], thus being its extension. A code example of a widget definition is shown below:

```
const stateful_attrs_monitor = webix.protoUI
({"name": 'stateful_attrs_monitor'...},

TangoWebappPlatform.mixin.Stateful,
attrs_monitor_view);
```

A widget's functionality may be enriched using Waltz platform mixins. PlatformContext API is accessible from Waltz widgets.

DEPLOYMENT

Waltz is distributed as a standard Java Web Application Archive - .war file [19]. Therefore it can be deployed at any Java EE compliant application server. Apache Tomcat 9 server [20] is used as Waltz's production environment at DESY as it is very flexible in terms of security configuration and provides other useful features e.g., HTTP/2.0.

Standard components like .war files and Java EE compliant application servers enable implementation on a standard enterprise infrastructure which scales very well. For instance, it is possible to deploy several Waltz instances behind a load balancer [21], thus allowing for an enormous number of simultaneous users. It is also possible to deploy several instances of Waltz under different names at the same time or to deploy a Waltz instance from a specific branch for a specific beamline.

REAL LIFE EXAMPLES

Waltz itself is a powerful tool that can be used by end-users to operate Tango Controls based environments. In this section we will show real life Waltz features based on the components described in the section above and taken from release v0.7.3 [9].

Waltz leverages web applications principles making the whole application user specific. Therefore, users first must login into Waltz. This also allows secure deployments of Waltz e.g., at DESY Waltz is integrated with DESY internal security system.

After log in to the version v0.7.3 of Waltz users can see an application consisting of 5 parts (Fig. 5).

Toolbars

Toolbars provide some utility widgets: ***top toolbar*** – links to the documentation, scripting console, terminal, settings bar and sign out button; ***bottom toolbar*** – application logs, current Tango REST API request status and "report an issue" link.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH3003

Side Panels

Side panels are connected with application's management. The *right side panel* shows a user actions log. The *left side panel* consists of the following parts:

Tango hosts tree. This widget can show several Tango Controls hosts at once. It also features a filter for quick search of required Tango device.

Tango device control panel. It shows attributes, commands and pipes of a particular Tango device selected in Tango hosts tree. Three small control widgets provide attributes read/write/plot functionality; commands execution and pipes reading. This widget features *clever* filter where user can simultaneously search through attributes, commands and pipes or only one of them, e.g., "a:double" in the search field will search for an attribute that contains "double".

Information/properties panels. Tango Controls host, device, attribute, command, pipe information and properties can be found here. It is possible to change properties from this widget.

Main View

Main view consists of dynamic tabs. Tabs may be of any complexity varying from simple iframe [22] to a rich embedded applications integrated with other frameworks e.g. React, Angular etc [23]. Some of them are listed in this section.

Dashboard. This widget allows users to create different profiles. Each profile can be of type "table" or "plot". A user configures each profile by drag-n-drop'ing Tango attributes into Dashboard. Its configuration is persistent per user basis (Fig. 6).

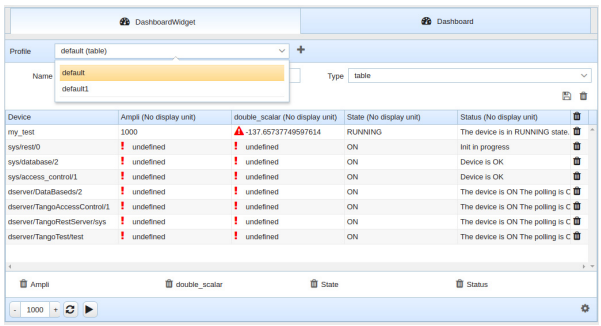

Figure 6: Dashboard widget.

Tango device configuration/monitoring. The whole Tango device can be reconfigured/monitored via this widget, e.g., you can set properties, define attribute alarm/warning ranges, set polling, view all attribute values etc.

Tango hosts manager. Having this widget user can monitor the state of the remote Tango hosts as well as restart particular remote Tango servers, navigate through Tango server's devices etc. Statuses of Tango servers are updated using native Tango event system exported via Server-Sent Events [7].

Tango attribute/command/pipe view. Main view of Tango attributes differs depending on the attribute's type. For scalars – it is a streaming plot view, for spectrum – plot and for images – heatmap (Fig. 7).

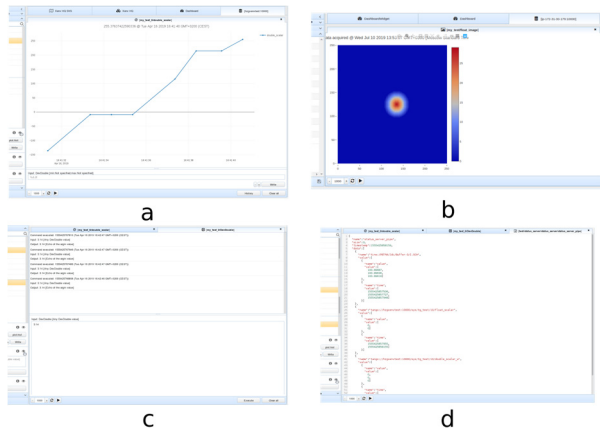

Figure 7: a – Tango scalar attribute main view; b – Tango image attribute main view; c – Tango command main view; d – Tango pipe main view.

In attribute view users can write and plot any attribute value and continuously update it. Command view allows users to execute Tango commands including those which require an argument. This command view also shows the history of the command execution. Pipe view displays Tango pipe value highlighted as JSON.

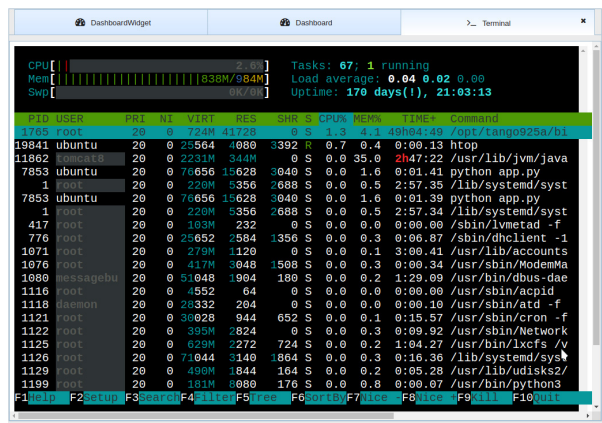

Figure 8: Terminal widget.

Teminal. It displays remote Linux terminal session. This widget provides fully functional remote Linux session so users may ssh to any Linux machine and perform required operations there. In Fig. 8 user executes *htop* program.

User settings. Here users can specify Tango REST API url, add/remove Tango hosts to the Tango hosts tree widget, define device filters and add new Tango devices to the Tango host. The device filter in this widget differs from other filters, e.g., device filter affects devices which will be actually loaded by Waltz, rather than filter in Tango hosts tree where widget performs on-fly filtering.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH3003

FURTHER DEVELOPMENTS

Waltz is already a production-ready development platform. Nonetheless, many new features and improvements are already on the raodmap.

We want to make the server-side of the application a rich application. That will allow integration with other control systems or even direct hardware communication. From the client perspective the platform will provide a well-defined API and communication protocol based on the reactive manifesto [24].

Also it is planned to improve SVG integration which will allow to define the visibility for particular layers as well as provide a real time feedback to users from upstream hardware by, for instance, displaying Tango device attribute values and updating them in real time either using client polling and/or the Tango event system.

User roles will be introduced. Users with an expert role will be able to create views using drag-n-drop visual constructors as well as simple macros.

Scripting capabilities will be extended with visual editor as well as integration with other scripting platforms e.g., Sardana integration [25] is planned.

A dedicated version of views for mobile devices will be added.

CONCLUSIONS

In this paper we have given an overview of Waltz as a platform for Tango Controls web/mobile applications development. Namely, we have covered a number of APIs, deployment and UI widgets that exist in Waltz.

In the previous sections we have looked at Waltz from different perspectives of what this platform offers to a developer: different views of architecture, APIs, deployment and how it can be implemented using real life examples.

It is clear that implementing features of enterprise web applications greatly improves user experience with SCADA systems. Among these features are: integrated UI; storing per-user basis data; deploying multiple instances of the application with specific for each use case widgets; having multiple views based on user role etc.

As per July 2019 when this paper was written about only 9 man/month efforts within 2 years were invested into Waltz [26]. We believe that in this limited time Waltz has become a very powerful and rich platform for development of custom Tango Controls web/mobile applications with great potential. Waltz has been in use for beamline commissioning at DESY for several months in which it proved to be mature and stable enough to replace other Tango Control software.

A number of workshops were organized to spread the knowledge on how Tango developers may use Waltz platform [15, 27-29] as well as a number of presentations were made to demonstrate Waltz as a product for end-users [30-32].

It is planned to evolve Waltz and make it more powerful in terms of end-user application (visual scripting editors, SVGs) and in terms of developer's platform. We foresee further developments to integrate other control systems into Waltz and to implement a direct hardware communication.

ACKNOWLEDGMENTS

The authors would like to thank Andrew Götz, Jean-Michel Chaize from ESRF and Tango Collaboration for believing in Waltz and funding 3 man/month of the Waltz platform development in 2018.

Developers team gives special thanks to Fabian Wilde and Vadim Murzin for their patience in testing Waltz and providing very useful feedbacks.

We thank: HZG and DESY colleagues, Christina Krywka, Jorg Hammel, Felix Beckmann for their support;

DESY, Solaris and ESRF which hosted several Waltz workshops;

Webix team for brilliant library that empowers Waltz greatly and for supporting open source projects.

REFERENCES

[1] A.Götz *et. al.*, "The TANGO Controls Collaboration in 2015", in *Proc. 15th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 585-588. doi:10.18429/JACoW-ICALEPCS2015-WEA3001

[2] A.Götz *et. al.*, "Tango Heads for Industry", in *Proc. 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 1195-1200. doi:10.18429/JACoW-ICALEPCS2017-THCPL05

[3] R. Bourtembourg *et. al.*, "TANGO Kernel Development Status", in *Proc. 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 27-33. doi:10.18429/JACoW-ICALEPCS2017-MOBPL02

[4] M. Canzari *et. al.*, "A GUI prototype for SKA1 TM Services: compliance with user-centered design approach", in Proc. *Software and Cyberinfrastructure for Astronomy V Conf. (Astronomical telescopes + Instrumentation SPIE'18)*, Austin, Texas, United States, Jun. 2018. doi:10.117/12.2313276

[5] OpenAjax, http://www.openajax.org/

[6] SVG, https://en.wikipedia.org/wiki/Scalable_Vector_Graphics

[7] SSE, https://www.w3.org/TR/eventsource/

[8] I. Khokhriakov, "Streaming Tango events to HTTP via SSE", 33rd Tango Collaboration meeting, DESY, Hamburg, Germany, Jun. 2019.

[9] Waltz releases, https://github.com/tango-controls/waltz/releases

[10] A. Rauschmayer, *Exploring ES6*. Leanpub, 2015.

[11] Webix, https://webix.com/

[12] JavaScriptMVC-1.5.x, https://github.com/jmvc-15x

[13] JS Mixin, http://javascript.info/mixins

[14] Waltz Developers guide, http://www.waltz-controls.space

[15] I. Khokhriakov, "TangoWebapp Insights", 32nd Tango Collaboration meeting, ELI-Beamlines, Dolni Brezany, Czech Republic, Jun. 2018.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH3003

[16] JS Promises, https://www.w3.org/2001/tag/doc/promises-guide

[17] JS async/await, https://developers.google.com/web/fundamentals/primers/async-functions

[18] webix-protoUI, https://docs.webix.com/api___protoui.html

[19] war file, https://en.wikipedia.org/wiki/WAR_(file_format)

[20] Apache Tomcat 9, https://tomcat.apache.org/download-90.cgi

[21] Load Balancer, https://en.wikipedia.org/wiki/Load_balancing_(computing)

[22] iframe, https://www.w3schools.com/html/html_iframe.asp

[23] Webix, Integration with Other Frameworks, https://docs.webix.com/desktop__third_party_integration.html

[24] The Reactive Manifesto, https://reactivemanifesto.org

[25] Sardana, https://sardana-controls.org/

[26] Waltz GitHub contributions, https://github.com/tango-controls/waltz/graphs/contributors

[27] I. Khokhriakov, O. Merkulova, "Waltz worshop@Solaris", Waltz workshop@Solaris, Solaris, Krakow, Poland, Jul. 2018.

[28] I. Khokhriakov, O. Merkulova, "Waltz architecture overview", Tango WebUI workshop, MAX-IV, Lund, Sweden, Oct. 2018.

[29] I. Khokhriakov, O. Merkulova, "Waltz worshop@ESRF", ESRF, Grenoble, France, Jan. 2019.

[30] I. Khokhriakov, "TangoWebapp status report", INAF, Firenze, Italy, Jun. 2017.

[31] I. Khokhriakov, "TangoWebapp workshop@DESY", DESY, Hamburg, Germany, May 2018.

[32] O. Merkulova, "TangoWebapp overview", 32nd Tango Collaboration meeting, ELI-Beamlines, Dolni Brezany, Czech Republic, Jun. 2018.